

VPRTempo: A Fast Temporally Encoded Spiking Neural Network for Visual Place Recognition

Adam D Hines

Peter G Stratton

Michael Milford

Tobias Fischer

Abstract— Spiking Neural Networks (SNNs) are at the forefront of neuromorphic computing thanks to their potential energy-efficiency, low latencies, and capacity for continual learning. While these capabilities are well suited for robotics tasks, SNNs have seen limited adaptation in this field thus far. This work introduces a SNN for Visual Place Recognition (VPR) that is both trainable within minutes and queryable in milliseconds, making it well suited for deployment on compute-constrained robotic systems. Our proposed system, VPRTempo, overcomes slow training and inference times using an abstracted SNN that trades biological realism for efficiency. VPRTempo employs a temporal code that determines the *timing* of a *single* spike based on a pixel’s intensity, as opposed to prior SNNs relying on rate coding that determined the *number* of spikes; improving spike efficiency by over 100%. VPRTempo is trained using Spike-Timing Dependent Plasticity and a supervised delta learning rule enforcing that each output spiking neuron responds to just a single place. We evaluate our system on the Nordland and Oxford RobotCar benchmark localization datasets, which include up to 27k places. We found that VPRTempo’s accuracy is comparable to prior SNNs and the popular NetVLAD place recognition algorithm, while being several orders of magnitude faster and suitable for real-time deployment – with inference speeds over 50 Hz on CPU. VPRTempo could be integrated as a loop closure component for online SLAM on resource-constrained systems such as space and underwater robots.

I. INTRODUCTION

Spiking neural networks (SNNs) are computational tools that model how neurons in the brain send and receive information [1]. SNNs have attracted significant interest due to their energy efficiency, low-latency data processing, especially when deployed on neuromorphic hardware such as Intel’s Loihi or SynSense’s Speck [2], [3]. However, modeling the complexity of biological neurons limits the computational efficiency of SNNs, especially where resource-constrained systems with real-time applications are concerned. Using a SNN system that trades biological realism for overall system efficiency would be well suited for a variety of robotics tasks [4], [5], [6], [7].

One such task that could benefit from fast and efficient SNNs is visual place recognition (VPR), where incoming query images are matched to a potentially very large reference database, with a range of applications in robot localization

The authors are with the QUT Centre for Robotics, School of Electrical Engineering and Robotics, Queensland University of Technology, Brisbane, QLD 4000, Australia. Email: adam.hines@qut.edu.au

This work received funding from Intel Labs to TF and MM, AUS-MURIB000001 associated with ONR MURI grant N00014-19-1-2571 and an ARC Laureate Fellowship FL210100156 to MM, and the Commonwealth of Australia as represented by the Defence Science and Technology Group of the Department of Defence to PGS. The authors acknowledge continued support from the Queensland University of Technology (QUT) through the Centre for Robotics.

and navigation [8], [9], [10], [11], [12], [13], [14], [15], [16], including as a loop closure component in Simultaneous Localization and Mapping (SLAM) [10], [17], [18], [19], [20]. However, VPR remains challenging as query images often have significant appearance changes when compared to the reference images, with factors such as time of day, seasonal changes, and weather contributing to these differences [21], [22], [23]. SNNs represent a unique way to perform VPR for their low-latency and energy efficiency, especially where this is an important consideration for the overall design of a robot [24]. The flexible and adaptive learning that SNNs take advantage of make them ideal for VPR, as network connections and weights will learn unique place features from reference datasets.

SNNs employ a variety of encoding mechanisms, with the traditional rate encoding suffering from substantial computational costs due to its reliance on spike counts [25], [26], [27]. Our work instead adopts a temporal encoding strategy inspired by central information storage techniques used by the brain, reducing the overall content the network processes during learning [28], [29]. This takes a different approach to what information each spike is representing and transmitting when compared to previous studies like those by Hussaini et al. [8], [9]. We leverage a simplified SNN framework [29], modular networks [30], [31], [32], and one-hot encoding for supervised training to substantially decrease training and inference times. Compared with previous systems, our SNN achieves query speeds exceeding 50 Hz on CPU hardware for large reference databases (>27k places) and even higher inference speeds when deployed on GPUs (as high as 500 Hz), indicating its potential usefulness for real-time applications [8], [11], [12], [14], [33], [34], [35], [36].

Specifically, the key contributions of this work are:

- 1) We present VPRTempo, a novel SNN system for VPR that, to the best of our knowledge, for the first time encodes place information via a temporal spiking code (Figure 1), significantly increasing the information content of each spike.
- 2) We significantly lower the training time of spiking networks to under an hour *and* increase query speeds to real-time capability on both CPUs and GPUs, with the potential to represent tens of thousands of places even in resource-limited compute scenarios.
- 3) We demonstrate that our lightweight and highly compute efficient system results in comparable performance to popular place recognition systems like NetVLAD [11] on the Nordland [37] and Oxford RobotCar [38] benchmark datasets.

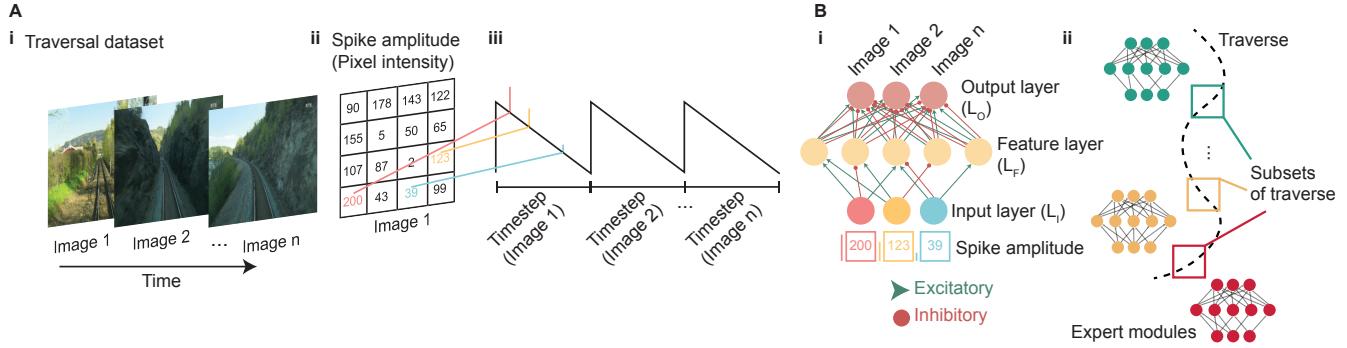


Fig. 1. **A-i** Image sequences from standard VPR datasets (Nordland, Oxford RobotCar) are filtered and processed to be converted into **A-ii** spikes where the pixel intensity determines amplitude. **A-iii** In order to temporally encode spikes to an abstracted theta oscillation, amplitudes determine the spike timing during a timestep. **B-i** Once spikes have been generated, they are passed into a SNN with 3 layers; an input, feature layer, and a one-hot encoded output layer where each output neuron represents one place. **B-ii** In order to scale the system for large datasets, we train individual expert module SNNs of up to 1000 places from subsets of an entire reference dataset.

To foster future research and development, we have made the code available at:

<https://github.com/QVPR/VPRTempo>

II. RELATED WORKS

In this section, we review previous works about VPR (Section II-A), how robotics have used and deployed SNNs (Section II-B), using SNNs to perform VPR tasks (Section II-C), and using temporal encoding in SNNs (Section II-D).

A. Visual place recognition

Visual place recognition (VPR) is an important task in robotic navigation and localization for an accurate, and ideally comprehensive, representation of their environment and surroundings. The basic principle of VPR is to match single or multiple query image sets to previously seen or traversed locations in a reference database [10], [21], [22], [23]. This remains challenging for numerous reasons, including but not limited to: changes in viewpoint, time of day, or severe weather fluctuations [39], [40].

Several systems have been deployed to tackle this problem, ranging from simple approaches such as Sum of Absolute Differences (SAD) [33], over handcrafted systems like DenseVLAD [13] to learned approaches like NetVLAD [11], [33] and many others including [12], [14], [15]. These systems are commonly used in localization tasks such as loop closure in Simultaneous Localization And Mapping (SLAM) [41]. Although these approaches have shown to be highly effective at accurate place recognition, they require computational and time intensive training regimes and learn generalized feature extraction, rather than actually learning the specific location.

In this work, we are interested in alternative spiking neural network solutions to approach place recognition, which have desirable characteristics such as being able to have online adaptation and energy efficiency, especially when deployed on specialized neuromorphic hardware [2], [3], [42], [43].

B. SNNs in robotics

Thanks to these characteristics, SNNs have been deployed to perform a variety of robotics tasks, including in the physical interaction domain for robotic limb control and manipulation [6], scene understanding [44], and object tracking [45]. Of significant interest is the potential to combine neuromorphic algorithms and hardware for these tasks [5], since this has the capacity to decrease compute time and conserve energy consumption where this is a concern (e.g. battery powered autonomous vehicles). However, it is important to note that the deployment and use cases of SNNs in robotics is limited and has been mostly constrained to simulations or indoor/small-scale environments [8], [46].

C. Spiking neural networks for robot localization

Neuromorphic computing involves the development of hardware, algorithms, and sensors inspired by neuroscience to solve a variety tasks [5], [12], [47], [48], [49]. Clearly, the brain is robustly capable of place recognition in a variety of contexts, providing a solid rationale to explore neuromorphic systems for VPR related challenges. A recent study exhibited prolonged training and querying times in an SNN, due to the reliance on the low spike efficiency of rate encoding and modeling complex neuronal dynamics [8]. Robotic localization tasks have also employed SNNs to perform SLAM [48], [50], encode a contextual cue driven multi-modal hybrid sensory system for place recognition [12], and navigation [51], [52]. Our approach mitigates traditional barriers to deploying SNNs for real-time compute scenarios, enhancing the learning speed substantially by simplifying network processes and incorporating temporal spike encoding.

D. Using temporal encoding in spiking networks

Temporal encoding has, to date, not been utilized for VPR tasks but is a common method used in SNNs [28], [53], [54], [55]. Sometimes referred to as a latency code, rather than using the number of spikes to encode information, the timing of the spike is taken into consideration. This is particularly an important concept when designing a system that updates

weights using spike timing dependent plasticity (STDP) [56], as the temporal information of a pre-synaptic spiking neuron can help determine which post-synaptic neurons to connect to.

Various temporal coding strategies have been used for image classification where it achieved very high accuracy [55], [57]. One way to achieve temporal coding is to define the pixel intensity of an image not by the number of spikes the system propagates, but the timing of a single spike in a time-step [29]. Another method to achieve a similar outcome is to model the oscillatory activity of the brain to modulate when spikes occur, relative to the phase of a constant, periodic shift in a neurons internal voltage [3], [58], [59].

III. METHODOLOGY

This work naturally expands previously published works on SNNs for VPR. Hussaini et al. [8] have introduced a scalable means of representing places, whereby independent networks represent geographically distinct regions of the traverse. However, as reviewed in Section II-C, their network used rate coding with complex neuronal dynamics, which resulted in significant training times and non-real-time deployment. Here, by leveraging the temporal coding in conjunction with spike forcing proposed in BliTNet [29], we overcome these limitations and demonstrate that BliTNet, combined with modularity, results in a high-performing VPR solution. We have completely re-implemented BliTNet in PyTorch [60], optimizing for computational efficiency and fast training times, thereby making it well-suited for deployment on compute-bound systems.

This section is structured as follows: We first briefly introduce the underlying BliTNet in Section III-A. This is followed by the modular organization of networks in Section III-B. We then describe our novel efficient parallelizable training and inference procedure in Section III-C.

A. Temporal coding for visual place recognition

Network architecture: Each network consists of 3 layers. The input layer L_I has as many neurons as the number of pixels in the input image, i.e. $W \cdot H$ neurons with W and H being the width and height of the input image, respectively. Neurons in L_I are sparsely connected to a feature layer L_F . The L_F layer is fully connected to the output layer L_O , which is one-hot encoded, such that each neuron $n_i \in L_O$ corresponds to one geographically distinct place from a set of reference images $\mathcal{D} = \{p_1, p_2, \dots, p_N\}$. The number of output neurons $|L_O|$ matches the number of distinct places N . For training that uses multiple traversals over different days/times, the same output neuron is used to encode that place so the network can learn changes in environments.

Network connections: Connections from $L_I \rightarrow L_F$ are sparse and determined by excitatory and inhibitory connection probabilities P_{exc} and P_{inh} respectively. Connections from $L_F \rightarrow L_O$ are fully connected, such that every neuron in L_O receives both excitatory and inhibitory input. Excitatory and inhibitory connections are individually defined since

inhibitory weights undergo additional normalization (refer to the Homeostasis subsection below).

Neuron dynamics: The neuron state x_j^n for neuron j in layer n evolves in the following way:

$$x_j^n = \sum_{m=1}^N \sum_i x_i^m(t)(W_{ji}^{+nm} - W_{ji}^{-nm}) + C - \theta_j^n, \quad (1)$$

where x_i^m is input neuron i in layer m , $\theta \in [0, \theta_{max}]$ is the neuron firing threshold, C is a constant input, and W_{ji}^{+nm} and W_{ji}^{-nm} are the excitatory and inhibitory weights, respectively. x_j^n is clipped in the range $[0, 1]$ to prevent spike vanishing or exploding, respectively.

Weight updates and learning rules: All weights between connected neurons are updated using spike timing dependent plasticity (STDP) rule [29], [56]. STDP strengthens a connection between two neurons when the pre-synaptic cell fires before the post-synaptic, and vice versa. As the spike amplitude represents the timing in an abstracted theta oscillation or time-step, we can determine whether a pre- or post-synaptic neuron fired first using:

$$\Delta W_{ji}^{nm}(t) = \frac{\eta_{STDP}(t)}{f_j^n} \cdot [\Theta(x_i^m(t-1)) \cdot \Theta(x_j^n(t)) \cdot (0.5 - x_j^n(t))], \quad (2)$$

where W_{ij}^{nm} refers to both positive and negative weights, η_{STDP} is the STDP learning rate, f_j^n is the target firing rate of neuron j in layer n , $\Theta(\cdot)$ is the Heaviside step function, x_i^m and x_j^n are the neuron states of pre- or post-synaptic neurons respectively, and t is the timestep. Effectively, a pre-synaptic neuron firing before a post-synaptic one will undergo a positive weight shift (synaptic potentiation). The inverse of this scenario will result in a negative weight shift (synaptic depression). To ensure that the weights persist as positive or negative connections, they cannot change sign during training. Therefore, if a sign change is detected then that connection will be reset to be $\epsilon = 10^{\pm 6}$, whereby the sign of ϵ matches the sign prior to the sign change.

The η_{STDP} is initialized to $\eta_{STDP}^{\text{init}}$ and annealed during training according to:

$$\eta_{STDP}(t) = \eta_{STDP}^{\text{init}}(1 - \frac{t}{T})^2, \quad (3)$$

where t is the current time step and T is the total number of training iterations.

Homeostasis: Inhibitory connections undergo additional normalization to balance and control excitatory connections. Whenever the sum input to a post-synaptic neuron is positive, the negative weights increase slightly. The inverse occurs when the net input is negative to post-synaptic neurons. Both cases are implemented by:

$$\hat{W}_{ji}^{-nm}(t) \leftarrow W_{ji}^{-nm}(t) \left(1 - \eta_{STDP}(t) \cdot \Theta \left(\sum_i x_i^m(t) \right) \right). \quad (4)$$

When the network is first initialized, neurons have randomly generated firing thresholds which may never be crossed depending on the input. To prevent neurons from being consistently inactive, an Intrinsic Threshold Plasticity (ITP) learning rate η_{ITP} is used to adjust the spiking threshold to an ideal value for post-synaptic neurons:

$$\Delta\theta_j^n(t) = \eta_{\text{ITP}}(t) \cdot (\Theta(x_j^n(t)) - f_j^n). \quad (5)$$

If a threshold value goes negative, it is reset to 0. η_{ITP} is initially set to $\eta_{\text{ITP}}^{\text{init}}$ and annealed as per Equation 3.

Spike forcing: Spike forcing [29] was used in the output layer L_O to create a supervised readout of the result that is represented in the feature layer L_F , similar to the delta learning rule [61]. For each learned place p_i , the assigned neuron n_i in the output layer L_O (refer to III-A subsection Network Architecture) was forced to spike with an amplitude of $x_{\text{force}} = 0.5$.

The network uses the differences of the calculated spikes from $L_F \rightarrow L_O$ to the x_{force} to encourage the amplitudes to match. STDP learning rules strengthen connections between pre-synaptic neurons that cause output spikes and weakens those that do not:

$$\Delta W_{ji}^{nm}(t) = \eta_{\text{STDP}}(t)/f_j^n \cdot [x_i^m(t-1)(x_{\text{force}}^n(t) - x_j^n(t))]. \quad (6)$$

B. Modular place representation for temporal codes

The brain encodes scene memory with place cells into sparse circuitry using engrams, individual circuits of neurons that activate in response to external cues for recall [62]. To mimic the concept of an engram we train multiple expert networks, which provides several advantages: 1) Smaller networks train faster and are more accurate, 2) the heterogeneity of initial network seeds is uniformly random, thus a query image q trained in network N_i is unlikely to activate output neurons with a high enough amplitude in other networks to generate a false positive match, and 3) similar to previous work [8], modularizing improves system scalability to learn more places.

Choosing the number of expert modules to employ depends on the dataset and number of places being encoded. Accuracy is influenced by T and learning rate annealment therefore too few or too many places will affect the systems learning capability.

To that end, we employ a system that can simultaneously train non-overlapping subsets of images into separate networks N_i and then test query images q across all networks at once [8]. Formally the union of all networks U is described as:

$$U = \bigcup_{i=1}^{|N|} N_i \quad \text{with} \quad N_i \cap N_j = \emptyset \quad \forall i \neq j. \quad (7)$$

C. Efficient implementation

Our efficient implementation for VPRTempo involves training and querying a 3D weight tensor $\mathbf{T} \in \mathbb{R}^{|N| \times |L_i| \times |L_j|}$ with $|N|$ being the number of modules, $|L_i|$ the number of the neurons in the pre-synaptic layer (either L_I or L_F), and $|L_j|$ being the number of neurons in the post-synaptic layer (L_F or L_O). This setup capitalizes on parallel computing to boost efficiency and speed [60]. During training, the weight tensor \mathbf{T} is multiplied by an image tensor $\mathbf{I} \in \mathbb{R}^{|N| \times 1 \times |L_i|}$ that holds images associated with a particular module and their input spike rates. Single query images are fed to all modules concurrently, leveraging parallel processing to decrease the overall inference time.

IV. EXPERIMENTAL SETUP AND IMPLEMENTATION

Here we describe our implementation in Section IV-A, the datasets used in Section IV-B, evaluation metrics in Section IV-C, the baseline methods that we compare and contrast our network to in Section IV-D, and finally our strategy for optimizing system hyperparameters in Section IV-E.

A. Data processing

Our network was developed and implemented in Python3 and PyTorch [60], converting the original BLiTNet implementation from NumPy to efficiently use multi-dimensional tensors and GPU acceleration for improved network performance. The system is trained on non-overlapping and geographically distinct places from multiple standard VPR datasets (Nordland and Oxford RobotCar [37], [38]), as commonly used in the literature [8], [9], [10], [63]. All reference and query input images underwent pre-processing for gamma correction, resizing, and patch normalization [33]. Specifically, Gamma correction was used to normalize pixels ρ_i of the input images:

$$\rho_i^{\text{norm}} = \begin{cases} \rho_i^\gamma & \text{for } 0 \leq \rho_i^\gamma \leq 255 \\ 255 & \text{for } \rho_i^\gamma > 255, \end{cases} \quad (8)$$

where $\gamma = \frac{e^{\lambda \times 255}}{e^\mu}$ with $\lambda = 0.5$ and $\mu = \bar{\rho}_i$. Normalized images were then resized to $W \times H = 28 \times 28$ pixels and patch-normalized with patch sizes $W_P \times H_P = 7 \times 7$ pixels [33]. Neuron states x are defined as floating point amplitudes in the range $[0, 1]$, where $x = 1$ is a full spike and $x = 0$ is no spike. Initial input spikes are generated from pixel intensity values $i = [0, 255]$ from training or test images and converted to amplitudes by $x = \frac{i}{255}$.

As an abstraction of theta oscillations in the brain, the neuron state determines the timing of a spike within a timestep (Figure 1). The network hyperparameters after performing a grid search (Section IV-E) were set as to the values in Table I. Network modules were trained to learn 1100 places with 3 modules (totalling 3300 places) from the Nordland dataset for the comparisons presented in Figure 2 and Table II, Figure 3.

Training each module is done simultaneously such that for each epoch, multiple unique places can be learned at the same time. After the network trained for sufficient epochs, all modules were simultaneously queried with independent

TABLE I
HYPERPARAMETERS FOR NETWORK TRAINING

| Parameter | θ_{max} | η_{STDP}^{init} | η_{ITP}^{init} | f_{min}, f_{max} | P_{exc} | P_{inh} | C |
|-----------|----------------|----------------------|---------------------|--------------------|-----------|-----------|-----|
| Value | 0.5 | 0.005 | 0.15 | [0.2, 0.9] | 0.1 | 0.5 | 0.1 |

network activation states to perform place matching. The network was trained for a total of 4 epochs.

B. Datasets

We follow the suggestion by Berton et al. [64] to have training and reference datasets that cover the same geographic area, rather than the common VPR practice to train networks on a training set that is potentially disjoint from the deployment area (i.e. reference dataset). Specifically, our network was evaluated on two VPR datasets, Nordland [37] and Oxford RobotCar [38] with image subsets and training performed as previously described [8]. Briefly, the Nordland dataset covers an approximately 728km traversal of a train in Norway sampled over Summer, Winter, Fall, and Spring. As is standard in the field, filtered images from the Nordland dataset removed any segments containing tunnels or speeds < 15 km/hr [8], [63], [65].

Images from both datasets were sub-sampled every 8 seconds, which is roughly 100 metres apart for Nordland, and 20 metres for Oxford RobotCar resulting in 3300 and 450 total places, respectively. It is important to note that while VPRSNN trained on these number of images, it reported accuracy performance on 2700 and 360 places respectively for the accuracy experiments due to the 20% of input images used for calibration [8]; note also that our proposed VPRTempo does not require a calibration dataset. For fairness of comparison, we present accuracy measurements that have omitted the first 20% of reference images.

C. Evaluation metrics

When a query image q is processed by the network, the matched place \hat{p} is the place i that is assigned to the neuron x_i in the L_O with the highest neuron state x_i :

$$\hat{p} = \arg \max_i x_i. \quad (9)$$

Network performance was measured with precision and recall [10]. Precision is the accuracy of identified places among all the predictions made, whereas recall is the coverage of true places successfully identified from all possible true places. Recall at $N = 1$ ($R@1$) measures the percent of correct matches when the network is forced to match every query image with a reference image. Recall at N ($R@N$) measures if the true place is within the top N matches. A match is only considered a true positive if it aligns with the exact ground truth, i.e. no ground truth tolerance was employed.

D. Baseline methods

We evaluated our system against standard VPR methods and state-of-the-art SNN networks. Sum of absolute differences (SAD) calculates the pixel-wise absolute difference

between query and reference database images, selecting a match based on the lowest sum [33]. For SAD, images were resized to 28 x 28 pixels. NetVLAD is trained to be highly viewpoint robust, and in this case images were kept at the original resolution [11]. We also compare with the current state-of-the-art Generalized Contrastive Loss (GCL) [14]. For Oxford RobotCar the dataset was resized to 320x240. Nordland and Oxford RobotCar are “on-the-rails” datasets with very limited viewpoint shift, which disadvantages GCL and NetVLAD. Our main comparison is VPRSNN, a recently reported SNN for performing VPR tasks based on spike rate coding by Hussaini et al. [8].

E. Hyperparameter search

To tune hyperparameters, we initially used a random search to identify the most influential parameters on match accuracy. Specifically, the hyperparameters we optimized were: firing threshold θ , initial STDP learning rate η_{STDP}^{init} , initial ITP learning rate η_{ITP}^{init} , firing rate range $f_{min} \leq f_j^n \leq f_{max}$, excitatory and inhibitory connection probabilities P_{exc} and P_{inh} , and the constant input C . An initial random sweep of 5,000 identified parameters for a refined grid search, specifically for f_{min} , f_{max} , P_{exc} , and P_{inh} . The resulting hyperparameters can be found in Table I and were used for both datasets. Hyperparameters were tuned on image subsets not used for any of the presented training and query performance results.

V. RESULTS

In this section, we establish the network training and query speeds when run on either CPU and GPU hardware (Section V-A). Then, we compare the performance of the network when compared to state-of-the-art and standard VPR systems (VPRSNN [8], GCL [14], NetVLAD [11], and SAD [33]) (Section V-B).

A. Training and querying speeds

To establish network performance against a previous state-of-the-art SNN for VPR (VPRSNN), we measured the time to train each network on 3300 places from Nordland [8]. We also tested query speeds of both networks, and the effect the number of places has on both (Figure 2). VPRSNN learned 3300 places in approximately 360 mins, compared to our system which required 17% (60 mins) and 0.28% (1 mins) of the time to train an equivalent number of images when run on CPU and GPU hardware, an Intel i7-9700K and Nvidia RTX 2080 respectively (Figure 2A). Whilst our network trained the fastest on GPU hardware, CPU training was still significantly faster than VPRSNN [8]. We attribute this predominately to increased spike efficiency, shifting away from modeling biological complexity in SNN simulators [66].

Capable of real-time query deployment, our system inferences on CPU and GPU hardware in the order of hundreds to thousands of Hz (Figure 2B), critical for resource limited compute scenarios. The time scaling of our system when training on reference datasets from 100 to 27,000 places was found to be $\mathcal{O}(n)$ for training and $\mathcal{O}(\log n)$ for query times,

TABLE II
COMPARISON OF NETWORK PERFORMANCE METRICS

| Method | Nordland (3300 places) | | | | | Oxford RobotCar (450 places) | | | | |
|-----------------|------------------------|-------------|------------|------------|-------------|------------------------------|-------------|-------------|------------|-------------|
| | R@1 (%) | Train (min) | CPU (Hz) | Query (Hz) | CPU (min) | GPU (Hz) | Train (min) | CPU (Hz) | Query (Hz) | GPU (Hz) |
| SAD [33] | 48 | - | 10 | - | 17 | 37 | - | 185 | - | 126 |
| NetVLAD [11] | 31 | 15120 | 0.5 | 15120 | 31 | 31 | 15120 | 0.5 | 15120 | 14 |
| GCL [14] | 32 | 360 | 2 | 360 | 77 | 33 | 360 | 0.4 | 360 | 75 |
| VPRSNN [8] | 53 | 360 | 2 | - | - | 40 | 49 | 2 | - | - |
| VPRTempo (ours) | 56 | 60 | 353 | 1 | 1634 | 37 | 3 | 1670 | 0.3 | 1955 |

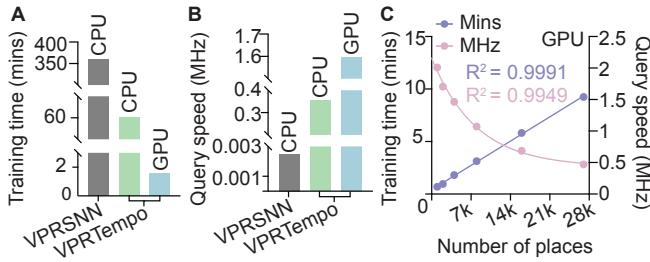


Fig. 2. A Comparison of training times for 3300 places from the Nordland dataset our system against state-of-the-art (VPRSNN [8]). VPRSNN trained in 360 mins, VPRTempo on a CPU trained in 60 mins, with our best performance of ≈ 1 min when VPRTempo runs on a GPU. B Querying speed 2700 places: VPRSNN 2 Hz, VPRTempo CPU at 353 Hz, and VPRTempo GPU queried at 1634 Hz. C Increasing the number of places scales the training time with a time complexity of $\mathcal{O}(n)$ and inference time increases with a time complexity of $\mathcal{O}(\log n)$.

respectively (Figure 2C). We next measured training and query speeds for the conventional SAD and NetVLAD and state-of-the-art GCL VPR methods, as summarized in Table II. VPRTempo trained and queried substantially faster than all of the other methods. We note that SAD has no training requirement to run and NetVLAD and GCL uses pre-trained models [11], [14]. Query speed for NetVLAD and GCL was measured as the time taken for query image feature extraction [11], [14]. Only CPU training or query speeds are listed for VPRSNN [8] since there is no GPU implementation of this network available.

B. Network accuracy

Now that we have established the beneficial training and inference speed, we next evaluated our system accuracy when compared to conventional and state-of-the-art VPR systems. For the Nordland dataset, our system achieved a $R@1$ of 56% (Table II). The conventional methods SAD and NetVLAD achieved an $R@1$ of 48% and 31% respectively, with the state-of-the-art GCL being 32%. Our main comparison competitor to VPRSNN achieved 53% (Table II).

By comparison, for Oxford RobotCar dataset we achieved a $R@1$ of 37% compared to 40% for VPRSNN (best performer) (Table II). SAD, NetVLAD, and GCL measured at 37%, 31%, and 33% respectively. In addition to this, we calculated the precision-recall curves and $R@N$ for both datasets to further validate our method, which are shown in Figure 3C and D. We speculate that VPRTempo performs better in Nordland

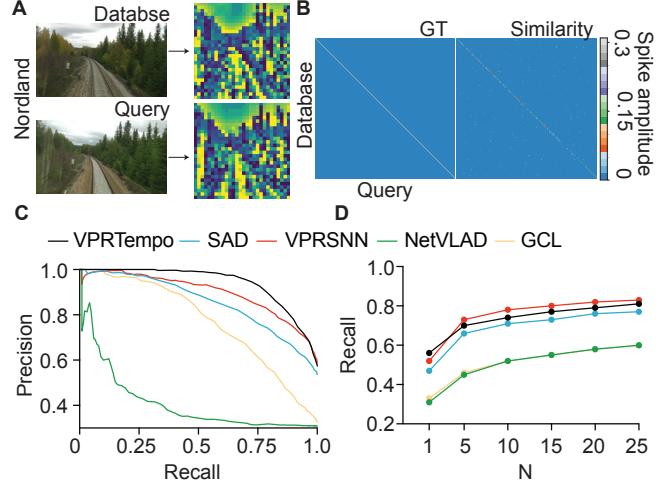


Fig. 3. A Example database and query image from the Nordland dataset (left) that are patch-normalized for network training (right). B Ground truth (GT, left) and descriptor similarity matrix from testing 3300 query images (right). C Precision-recall curves comparing our network with sum of absolute differences (SAD [33]), NetVLAD [11], Generalized Contrastive Loss (GCL [14]), and VPRSNN [8]. D Recall at N curves comparing methods as in C.

than Oxford RobotCar due to the lower viewpoint variation. As previously discussed (refer to Section IV-D), NetVLAD and GCL do not perform as well on static view-point datasets like Nordland and Oxford RobotCar.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented the first temporally encoded SNN to solve VPR tasks for robotic localization and navigation. We significantly improved the capacity of SNN systems to learn and query place datasets over state-of-the-art and conventional systems (SAD [33], NetVLAD [11], GCL [14], VPRSNN [8]). In addition to this, we observed comparable network precision to these systems in correctly matching query to database references.

There are multiple future directions for this work and its application in robotic localization and VPR methods: 1) We are working toward translating our method onto Intel's neuromorphic processor, Intel Loihi 2 [3], to deploy on energy efficient hardware. 2) For deployment on neuromorphic hardware, we are investigating the use of event streams from event-based cameras as the input to our network in order to further reduce latencies and improve energy-efficiency. 3) Given our system's fast training and inference times, we

will explore an ensemble fusing SNNs representing the same places for increased robustness. 4) Finally, we are exploring deployment of our network onto a robot for online and real-time learning of novel environments.

REFERENCES

- [1] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7), 2022.
- [2] Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Simon Davidson, Jeffrey Pepper, David Clark, Cameron Patterson, and Steve Furber. Spinnaker: A multi-core system-on-chip for massively-parallel neural net simulation. In *IEEE Custom Integrated Circuits Conference*, 2012.
- [3] Garrick Orchard, E. Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T. Sommer, and Mike Davies. Efficient neuromorphic signal processing with Loihi 2. *arXiv 2111.03746*, 2021.
- [4] Ignacio Abadía, Francisco Náveros, Eduardo Ros, Richard R. Carrillo, and Niceto R. Luque. A cerebellar-based solution to the nondeterministic time delay problem in robotic control. *Science Robotics*, 6(58), 2021.
- [5] Antonio Vitale, Alpha Renner, Celine Nauer, Davide Scaramuzza, and Yulia Sandamirskaya. Event-driven vision and control for UAVs on a neuromorphic chip. *arXiv 2108.03694*, 2021.
- [6] J. Camilo Vasquez Tieck, Heiko Donat, Jacques Kaiser, Igor Peric, Stefan Ulbrich, Arne Roennau, Marius Zöllner, and Rüdiger Dillmann. Towards grasping with spiking neural networks for anthropomorphic robot hands. *Artificial Neural Networks and Machine Learning*, 2017.
- [7] J. Camilo Vasquez Tieck, Lea Steffen, Jacques Kaiser, Arne Roennau, and Rüdiger Dillmann. Controlling a robot arm for target reaching without planning using spiking neurons. In *IEEE International Conference on Cognitive Informatics & Cognitive Computing*, 2018.
- [8] Somayeh Hussaini, Michael Milford, and Tobias Fischer. Ensembles of compact, region-specific & regularized spiking neural networks for scalable place recognition. In *IEEE International Conference on Robotics and Automation*, 2023.
- [9] Somayeh Hussaini, Michael J Milford, and Tobias Fischer. Spiking neural networks for visual place recognition via weighted neuronal assignments. *IEEE Robotics and Automation Letters*, 7(2), 2022.
- [10] Stefan Schubert, Peer Neubert, Sourav Garg, Michael Milford, and Tobias Fischer. Visual place recognition: A tutorial. *IEEE Robotics and Automation Magazine*, 2023.
- [11] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [12] Fangwen Yu, Yujie Wu, Songchen Ma, Mingkun Xu, Hongyi Li, Huanyu Qu, Chenhang Song, Taoyi Wang, Rong Zhao, and Luping Shi. Brain-inspired multimodal hybrid neural network for robot place recognition. *Science Robotics*, 8(78), 2023.
- [13] Akihiko Torii, Relja Arandjelović, Josef Sivic, Masatoshi Okutomi, and Tomáš Pajdla. 24/7 place recognition by view synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015.
- [14] María Leyva-Vallina, Nicola Strisciuglio, and Nicolai Petkov. Data-efficient large scale place recognition with graded similarity supervision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [15] Amar Ali-bey, Brahim Chaib-draa, and Philippe Gigueré. MixVPR: Feature mixing for visual place recognition. *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023.
- [16] Paul-Edouard Sarlin, Frédéric Debraine, Marcin Dymczyk, Roland Siegwart, and Cesar Cadena. Leveraging deep visual descriptors for hierarchical efficient localization. *arXiv 1809.01019*, 2018.
- [17] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2), 2006.
- [18] Rafiqul Islam and H. Habibullah. A semantically aware place recognition system for loop closure of a visual SLAM system. *International Conference on Mechatronics, Robotics and Automation*, 2021.
- [19] Zhe Xin, Xiaoguang Cui, Jixiang Zhang, Yiping Yang, and Yanqing Wang. Real-time visual place recognition based on analyzing distribution of multi-scale CNN landmarks. *Journal of Intelligent Robot Systems*, 94, 2019.
- [20] Han Wang, Chen Wang, and Lihua Xie. Online visual place recognition via saliency re-identification. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.
- [21] Mubariz Zaffar, Sourav Garg, Michael Milford, Julian Kooij, David Flynn, Klaus McDonald-Maier, and Shoaib Ehsan. VPR-Bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change. *International Journal of Computer Vision*, 129(7), 2021.
- [22] Carlo Masone and Barbara Caputo. A survey on deep visual place recognition. *IEEE Access*, 9, 2021.
- [23] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1), 2016.
- [24] Navvab Kashiri et al. An overview on principles for energy efficient robot locomotion. *Frontiers in Robotics and AI*, 5, 2018.
- [25] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9), 2023.
- [26] Wenzhe Guo, Mohammed E. Foufa, Ahmed M. Eltawil, and Khaled Nabil Salama. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. *Frontiers in Neuroscience*, 15, 2021.
- [27] Ingvars Birznieks Roland S Johansson. First spikes in ensembles of human tactile afferents code complex spatial fingertip events. *Nature Neuroscience*, 7, 2004.
- [28] Wolf Singer. Distributed processing and temporal codes in neuronal networks. *Cognitive Neurodynamics*, 3, 2009.
- [29] Peter G. Stratton, Andrew Wabnitz, Chip Essam, Allen Cheung, and Tara J. Hamilton. Making a spiking net work: Robust brain-like unsupervised machine learning. *arXiv 2208.01204*, 2022.
- [30] Gasser Auda and Mohamed Kamel. Modular neural networks: A survey. *International Journal of Neural Systems*, 09(02), 1999.
- [31] Bart L.M. Happel and Jacob M.J. Murre. Design and evolution of modular neural network architectures. *Neural Networks*, 7(6), 1994. Models of Neurodynamics and Behavior.
- [32] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1), 1991.
- [33] Michael J. Milford and Gordon F. Wyeth. SeqSLAM: visual route-based navigation for sunny summer days and stormy winter nights. In *IEEE International Conference on Robotics and Automation*, 2012.
- [34] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-NetVLAD: Multi-scale fusion of locally-global descriptors for place recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- [35] Liu Liu, Hongdong Li, and Yuchao Dai. Stochastic attraction-repulsion embedding for large scale image localization. In *IEEE International Conference on Computer Vision*, 2019.
- [36] Michał R. Nowicki, Jan Wietrzykowski, and Piotr Skrzypczyński. Real-time visual place recognition for personal localization on a mobile device. *Wireless Personal Communications*, 97(1), 2017.
- [37] Daniel Olid, José M. Fácil, and Javier Civera. Single-view place recognition under seasonal changes. *PPNIV Workshop at International Conference on Intelligent Robots and Systems*, 2018.
- [38] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1), 2017.
- [39] Sourav Garg, Tobias Fischer, and Michael Milford. Where is your place, visual place recognition? In *International Joint Conference on Artificial Intelligence*, 2021.
- [40] Stefan Schubert, Peer Neubert, and Peter Protzel. Unsupervised learning methods for visual place recognition in discretely and continuously changing environments. In *IEEE International Conference on Robotics and Automation*, 2020.
- [41] Konstantinos A. Tsintotas, Loukas Bampis, and Antonios Gasteratos. The revisiting problem in simultaneous localization and mapping: A survey on visual loop closure detection. *IEEE Transactions on Intelligent Transportation Systems*, 23(11), 2022.
- [42] Jing Pei et al. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 572(7767), 2019.
- [43] Filipp Akopyan et al. TrueNorth: design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10), 2015.

- [44] Raphaela Kreiser, Alpha Renner, Vanessa R. C. Leite, Baris Serhan, Chiara Bartolozzi, Arren Glover, and Yulia Sandamirskaya. An on-chip spiking neural network for estimation of the head pose of the icub robot. *Frontiers in Neuroscience*, 14, 2020.
- [45] Ashwin Lele, Yan Fang, Justin Ting, and Arijit Raychowdhury. An end-to-end spiking neural network platform for edge robotics: From event-cameras to central pattern generation. *IEEE Transactions on Cognitive and Developmental Systems*, 14(3), 2022.
- [46] Raphaela Kreiser, Alpha Renner, Yulia Sandamirskaya, and Panin Pienroj. Pose estimation and map formation with spiking neural networks: towards neuromorphic SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.
- [47] Guangzhi Tang, Neelesh Kumar, and Konstantinos P. Michmizos. Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.
- [48] Guangzhi Tang, Arpit Shah, and Konstantinos P. Michmizos. Spiking neural network on neuromorphic hardware for energy-efficient unidimensional SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.
- [49] Hermann Blum, Alexander Dietmüller, Moritz Milde, Jörg Conradt, Giacomo Indiveri, and Yulia Sandamirskaya. A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor. *Robotics Science and Systems*, 2017.
- [50] F. Galluppi, J. Conradt, T. Stewart, C. Eliasmith, T. Horiuchi, J. Tapson, B. Tripp, S. Furber, and R. Etienne-Cummings. Live demo: Spiking ratSLAM: Rat hippocampus cells in spiking neural hardware. *IEEE Biomedical Circuits and Systems Conference*, 2012.
- [51] Junxiu Liu, Hao Lu, Yuling Luo, and Su Yang. Spiking neural network-based multi-task autonomous learning for mobile robots. *Engineering Applications of Artificial Intelligence*, 104, 2021.
- [52] Nelson Santiago Giraldo, Sebastián Izaza, and Ricardo Andrés Velásquez. Sailboat navigation control system based on spiking neural networks. *Control Theory and Technology*, 2023.
- [53] Iulia-Maria Comsa, Luca Versari, Thomas Fischbacher, and Jyrki Alakuijala. Spiking autoencoders with temporal coding. *Frontiers in Neuroscience*, 15, 2021.
- [54] Daniel Auge, Julian Hille, Etienne Mueller, and Alois Knoll. A survey of encoding techniques for signal processing in spiking neural networks. *Neural Processing Letters*, 53(6), 2021.
- [55] Bodo Rueckauer and Shih-Chii Liu. Temporal pattern coding in deep spiking neural networks. In *International Joint Conference on Neural Networks*, 2021.
- [56] Guo qiang Bi and Mu ming Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18(24), 1998.
- [57] Yi Chen, Hong Qu, Malu Zhang, and Yuchen Wang. Deep spiking neural network with neural oscillation and spike-phase information. In *AAAI Conference on Artificial Intelligence*, volume 35, 2021.
- [58] Eugene M. Izhikevich. Resonate-and-fire neurons. *Neural Networks*, 14(6), 2001.
- [59] Badr AlKhamissi, Muhammad ElNokrashy, and David Bernal-Casas. Deep spiking neural networks with resonate-and-fire neurons. *arXiv* 2109.08234, 2021.
- [60] Adam Paszke et al. PyTorch: an imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32, 2019.
- [61] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088), 1986.
- [62] Omid Miry, Jie Li, and Lu Chen. The quest for the hippocampal memory engram: From theories to experimental evidence. *Frontiers in Behavioral Neuroscience*, 14, 2021.
- [63] Timothy L. Molloy, Tobias Fischer, Michael Milford, and Girish N. Nair. Intelligent reference curation for visual place recognition via bayesian selective fusion. *IEEE Robotics and Automation Letters*, 6(2), 2021.
- [64] Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
- [65] Stephen Hausler, Adam Jacobson, and Michael Milford. Multi-process fusion: Visual place recognition using multiple image processing methods. *IEEE Robotics and Automation Letters*, 4(2), 2019.
- [66] Marcel Stimberg, Romain Brette, and Dan FM Goodman. Brian 2, an intuitive and efficient neural simulator. *eLife*, 8, 2019.