Neuromorphic Soldier-Agent Teaming

MSML 612: Deep Learning Final Project

Walter Peregrim

Table Of Contents

1 Problem Setup

In military operations, soldier agent teaming refers to the integration of autonomous or semi-autonomous robotic agents with human soldiers. However, achieving seamless communication and coordination between humans and machines remains a significant challenge due to the disparity in sophistication. There are several forms of communication which could be of use on the battlefield but we will focus on one in particular, hand gestures. A gesture is a means of communication that involves physical body movements to convey thoughts and feelings, sign language is a typical example. Because hand gesturing is non-verbal and non-vocal skill, image processing techniques are needed for a machine learning model to learn to accurately classify different gestures. However, conventional computing frameworks struggle to emulate the energy efficiency, real-time processing, adaptability, and robustness of human cognitive processes.

2 Background

2.1 Neuromorphic Computing

Neuromorphic computing is an innovative approach that seeks to create computer systems inspired by the structure and function of the human brain. At its core are spiking neural networks (SNNs), artificial neural networks that use discrete-time spikes to represent and process information. This mimics the communication between neurons in the brain, where information is encoded in the timing of electrical spikes. Neuromorphic computing offers several advantages for computer vision tasks, particularly in hand gesture recognition.

Hand gesture recognition is a complex computer vision task that involves analyzing dynamic visual input and identifying specific gestures made by users. Neuromorphic computing is well-suited for this task due to its unique characteristics. SNNs excel at processing temporal information, making them ideal for handling the continuous motion involved in gestures. The timing of spikes in SNNs represents essential temporal dynamics, enabling the recognition of different gestures based on precise spike patterns. Additionally, neuromorphic computing can be coupled with event-based sensors, such as Dynamic Vision Sensors (DVS), which capture changes in the visual scene as events. These event-based sensors closely resemble the way neurons respond to spikes, providing a natural integration with SNNs. For hand gesture recognition, event-based sensors can efficiently capture the rapid changes in hand movements during gesture execution, enhancing the system's responsiveness and accuracy.

One of the key advantages of neuromorphic computing is its low power consumption. In hand gesture recognition applications, this feature becomes crucial, especially in portable devices and wearable technology. The event-driven nature of SNNs ensures that computations are performed only when spikes occur, significantly reducing energy consumption and extending battery life in resource-constrained devices. Moreover, SNNs exhibit robustness to noise, making them more resilient to environmental challenges like lighting changes or occlusions that may occur in

real-world scenarios. This robustness enables the system to accurately recognize hand gestures even in less-than-ideal conditions, enhancing its reliability and usability. Neuromorphic computing also leverages the massive parallelism found in the human brain. This parallel processing capability allows SNNs to simultaneously recognize multiple gestures or handle several instances of the same gesture, making it well-suited for crowded environments or multi-user scenarios. Furthermore, SNNs can adapt and learn from data using spike-timing-dependent plasticity (STDP) and other learning rules. This adaptability is particularly valuable in dynamic scenarios where gesture patterns may change over time or when the system needs to recognize new gestures. The continuous learning and adaptation of SNNs improve their accuracy and performance, enhancing the overall gesture recognition system.

2.2 Leaky Integrate-And-Fire Neuron Model

Spiking neural networks exhibit complex dynamics that are mathematically described by the Leaky Integrate-and-Fire (LIF) model. In this model, individual neurons integrate incoming electrical signals over time instead of passing the sum of weighted inputs directly to the activation function, as with the artificial neuron. Once the membrane potential reaches a threshold value, the neuron generates an output spike and resets. The different components of the LIF neuron can be modeled as a resistor-capacitor (RC) circuit as seen in the figure below.

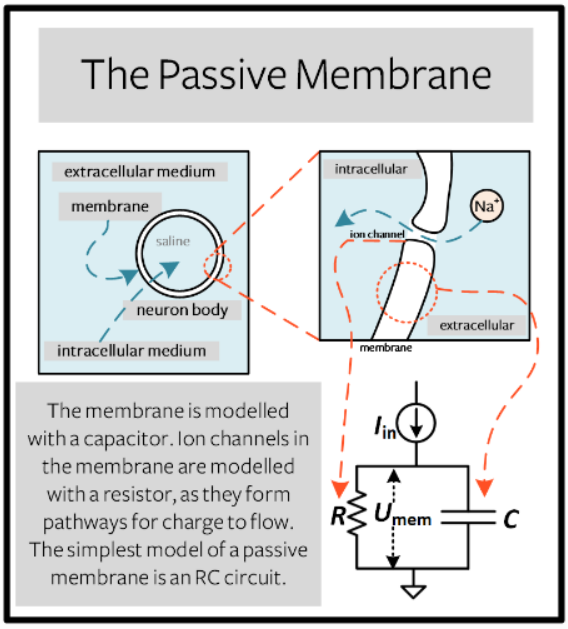

Figure 1: LIF Neuron Circuit

Neurons are surrounded by a thin lipid bilayer which insulates the conductive saline solution within the neuron from the extracellular medium. Electrically, this setup resembles a capacitor, with two conductive solutions on either side of the insulator. The membrane serves the vital function of regulating the movement of substances into and out of the cell, such as ions like Na.

Ordinarily, ions are blocked from freely entering or exiting the neuron's body due to the membrane's impermeability. However, certain channels in the membrane can be triggered to open when a current is injected into the neuron. This electrical charge movement can be likened to the behavior of a resistor.

The dynamics of this electric circuit are governed by a differential equation that is derived from Kirchhoff's current law, as seen in Figure 2 below, which states that the sum of all currents flowing into a node is equal to the sum of all currents flowing out of the node. This equation involves a time constant, tau, which characterizes how quickly the membrane potential responds to input currents. A smaller tau implies that the neuron's membrane potential changes very rapidly in response to input, while a larger tau implies the opposite[1].

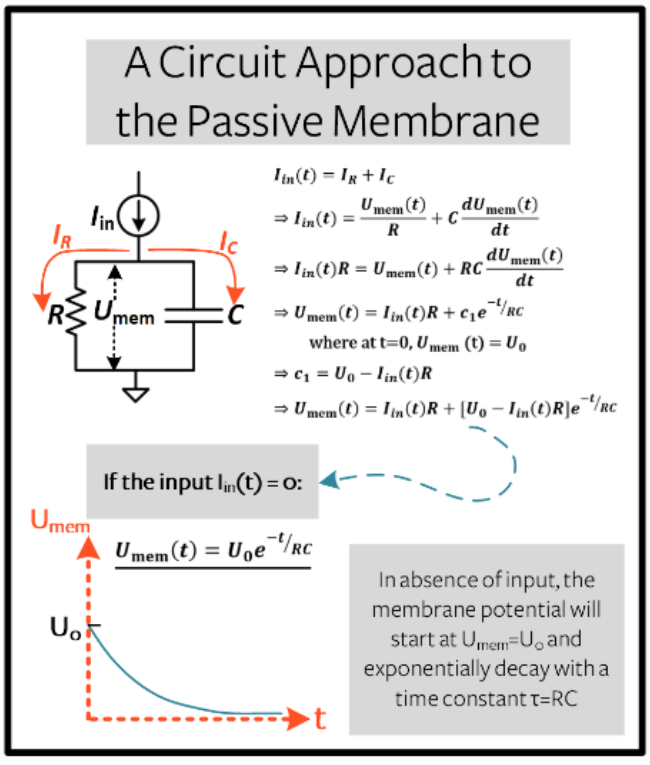

Figure 2: LIF Neuron Dynamics

Backpropagation, a foundational learning technique in conventional artificial neural networks, encounters significant challenges when applied to spiking neural networks due to the intrinsic nature of spiking behavior and the complexities of chain rule interactions. Spiking neurons communicate through discrete spikes, which are not differentiable. Consequently, the gradients needed for weight updates cannot be directly computed using conventional backpropagation. To address this, researchers have introduced surrogate gradients, which are continuous and differentiable approximations of the actual spike generation process. These surrogates allow gradient-based optimization to be applied to SNNs, which means they can be trained in the traditional manner.[2]

3 Data Description & Preprocessing

3.1 DVS128 Gesture Dataset

The dataset in question, spanning a period of six years, was captured employing the iniLabs DVS128 camera, a dynamic vision sensor boasting a resolution of 128 × 128 pixels. This sensor operates by detecting events arising from notable fluctuations in pixel intensity that surpass a predetermined threshold value. Each event registered by the camera is accompanied by its spatial coordinates within the dynamic pixel array, alongside a timestamp pinpointing the moment of the alteration.[3] Noteworthy is the camera's capacity to generate events at an average rate of 100,000 events per second, with a peak performance of 1 million events per second[4].

DVS128 gesture encompasses more than a thousand instances of 11 distinct hand and arm gestures. Executed by 29 different subjects, these gestures were rigorously captured through 122 trials conducted under three distinct lighting conditions. During each trial, a subject would stand against a static background and sequentially perform all 11 gestures, with lighting conditions held constant throughout. The gestures, each lasting approximately six seconds, encompass actions such as bilateral hand waving, substantial clockwise and counterclockwise arm rotations, as well as forearms rolling in both forward and backward directions. Notably, unconventional gestures like air guitar and air drums are also part of the dataset. The three lighting scenarios, comprising natural light, fluorescent light, and LED light, were meticulously chosen to mitigate potential disruptions caused by shadows and fluorescent light flicker on the DVS128 camera.

The dataset has been partitioned into two distinct sets. The training set comprises video data gleaned from 23 subjects, while the testing set is composed of recordings obtained from the remaining six subjects. Each individual trial is documented across two chronologically stamped files. The first file, an .aedat datafile, encapsulates the spatial coordinates of pixels that undergo marginal intensity changes. Meanwhile, the second file, a .csv document, encapsulates both the labels representing the true hand gestures and the time intervals within which these gestures transpire. The comprehensive architecture of this dataset thus provides valuable resources for research and analysis in the realm of dynamic vision and gesture recognition.

3.2 Data Preprocessing

The discrete gesture events inherently contain spatial data through their pixel coordinates. However, in order to fully harness the capabilities of spiking neural networks, it is imperative to encode these gesture events temporally. Several distinct strategies for temporal encoding are available, each yielding differing quantities of training instances. Within the context of the DVS128 gesture dataset, the creators adopt a methodology involving the sequential passage of gesture event sequences through an array of temporal filters, a process designed to encode pertinent transient information. This specific approach culminates in the formulation of training and testing sets consisting of 1,077 and 289 samples, correspondingly, thus amalgamating to form a total of 1,366 instances[3].

In our pursuit, an alternative temporal encoding mechanism is embraced: frame stacking. This technique segments input data into predefined temporal slots. Employing frame stacking facilitates the categorization of the unprocessed .aedat datafile according to class and time interval, subsequently archiving the outcomes in the form of numpy datafiles (.npz). As a practical illustration, consider the file "user03_natural.aedat," which is differentially partitioned into multiple numpy datafiles, labeled as "user03_natural_*.npz." Each of these files corresponds to a distinct class and is endowed with an allocated time interval (T1, T2). Correspondingly, the associated ground truth file, "user03_natural_labels.csv," is succinctly summarized using the following notation: T11 denotes the inception point for the primary class, while TN2 designates the termination point for the Nth class. By employing this methodology, our outcome comprises training and testing assemblages, encompassing 1,176 and 288 samples, respectively. In summation, this amalgamates into a total of 1,464 instances.

To fortify the resilience of our model, we engage in cross-validation, a technique for model validation that involves cyclically resampling distinct segments of the data during each iteration. The process of subjecting the model to diverse data iterations during both training and testing serves to counteract overfitting—a phenomenon denoting an overly tailored fit to the training data, at the expense of generalizability to novel data. Subsequent to the preprocessing of the raw data, it becomes amenable for input into our convolutional spiking neural network.

3.3 Alternative Datasets

3.2.1 IPN Hand dataset

The IPN Hand dataset stands as a comparatively modern benchmark video dataset with a specific purpose: the training and evaluation of deep neural networks intended for the task of Continuous Hand Gesture Recognition (HGR). This dataset encompasses an impressive count of more than 4,000 instances of gestures, extracted from a total of 800,000 frames obtained from the contributions of 50 distinct subjects. Within its scope, the dataset comprises a repertoire of 13 gestures, characterized by a blend of static and dynamic manifestations, all orchestrated for seamless interaction with touchless screens. These gestures were meticulously captured by the subjects themselves, utilizing their personal computers, thereby yielding RGB videos that exhibit varying camera-to-subject distances. Notably, the video recordings boast a resolution of 640 × 480 pixels and a frame rate of 30 frames per second.

Each participant embarked on a sequence of 21 gestures, delivered in an uninterrupted fashion, punctuated by three random pauses, all within the span of a single video. These specified gestures were meticulously chosen to facilitate the control of a pointer and to enable actions specifically tailored for seamless interaction with touchless screens. In a noteworthy departure from the DVS128 dataset, the IPN Hand dataset showcases a notably larger number of continuous gestures contained within each video instance. The meticulous process of data

collection was thoughtfully devised to encompass challenges characteristic of real-world scenarios in continuous HGR. These challenges encompassed gestures lacking transitional phases, the integration of natural movements as non-gesture segments, the presence of intricate backgrounds, and extreme variations in lighting conditions.[5]

It is worth highlighting, however, that the utilization of this dataset in the immediate term was curtailed by certain practical constraints. Given its substantial size, attempting to execute computations on a local computer becomes infeasible. The resource-intensive nature of the dataset renders it unsuitable for processing within the confines of a typical personal computer, as my own system encounters limitations in terms of memory allocation and cache management when handling these voluminous video files.

3.2.2 Lava Swipe Detector

During the year 2021, Intel Labs pioneered the creation of a proprietary Python-based library designated for implementation on their neuromorphic computing hardware, a platform christened as "Lava" [6]. Subsequently, in the latter part of July 2023, an augmentative addition was introduced to one of the repositories encompassing the Lava framework. This augmentation took the form of a tutorial centered around hand gesture recognition—a concerted effort aimed at effectively categorizing input hand motions into one of four discrete labels: "swipe left," "swipe right," "swipe down," and "swipe up." It is worth acknowledging that the scope of this hand gesture dataset, while not reaching the expanse witnessed in datasets such as DVS128 or IPN Hand, serves as a testament to the alignment of your chosen problem with the pursuits of Intel's Neuromorphic Computing Lab. The resonance between my focal area and the endeavors of such a pioneering entity provides a degree of assurance and validation to my chosen course of exploration.

3.2.3 Electronic Skin

Artificial skins represent an emerging domain within sensing technology, showcasing the potential to undertake real-time monitoring of critical physiological parameters encompassing blood pressure, temperature, and oxygen levels. In a recent stride, the innovation of stretchable strain sensors has come to the fore. These sensors emulate the perceptual attributes of human skin, capturing intricate nuances such as pressure, tension, and weight. The information harvested from electronic skin exhibits a heightened level of sophistication and insightfulness, setting it apart from the data encapsulated within the DVS128 gesture dataset. Remarkably, a laboratory situated at the University of New South Wales in Sydney, Australia, is actively engaged in a pioneering pursuit of applying a neuromorphic paradigm to the analysis of e-skin data[7]. This progressive initiative underscores the marriage of cutting-edge research with the realm of neuromorphic principles to unlock deeper understanding and insights from this innovative field.

4 Convolutional Spiking Neural Networks

4.1 Intuition

In conventional neural networks, time is typically represented discreetly, with computations occurring at each time step using continuous activations. These networks, often employed for tasks like image classification, overlook the fine-grained temporal dynamics and emphasize gradual weight adjustments driven by optimization techniques such as backpropagation. In contrast, CSNNs delve into a more nuanced understanding of time by emulating the spiking behavior seen in biological neurons. Neurons in CSNNs communicate via discrete spikes, triggered upon their membrane potential surpassing a certain threshold. This precise timing of spikes serves as a carrier of temporal information, enabling the network to adeptly capture dynamic patterns and sequences, making CSNNs well-suited for tasks like video analysis and temporal data.

In conventional convolutional neural networks (CNNs), feature maps are generated by convolving input data with learnable kernels. Kernels are small filters that slide across the input, extracting salient features via dot product operations. These feature maps undergo subsequent activation functions. However, these operations occur independently at each time step, with minimal temporal coherence. In CSNNs, the temporal interplay is more intricate. While kernels in CSNNs perform similar convolutions, the generated feature maps influence the membrane potentials of spiking neurons over time. Neurons respond not only to the presence of features but also to their temporal evolution[11]. As seen in Figure 4 below, neurons spike based on this temporal relationship with the evolving feature maps, enabling the network to encode not just static features but also dynamic changes in the input data.

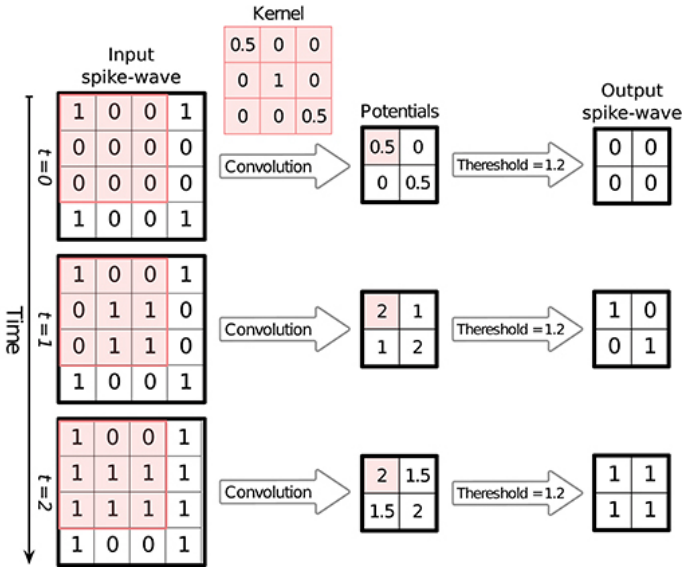

Figure 4: Feature Map Temporal Evolution

4.2 Our Model

Our model architecture is similar to the example one illustrated in Figure 5, but with a few key differences. I used five convolutional spiking layers instead of two, and each one consists of: (i) a 2D convolutional layer to extract features from images[8]; (ii) a 2D batch normalization layer which is responsible for regularizing its inputs via padding and scaling, which makes training faster and more stable[9]; and (iii) a spiking layer which is composed of a tensor of spiking neurons, with the same shape as the input to that layer; and (iv) a max pooling layer which is used to reduce the size of the feature map generated by the spiking-convolutional layer, this decreases the number of training parameters which subsequently reduces runtime[8]. The multidimensional output is flattened into a one-dimensional vector and then passed through not one, but two output layers, one of which comprises: (i) a dropout layer which is used as a mask to ignore the contributions of certain neurons as information is propagated to the next layer[10]; (ii) a fully connected layer which applies a linear transformation to its input vector; (iii) an output LIF spiking layer.

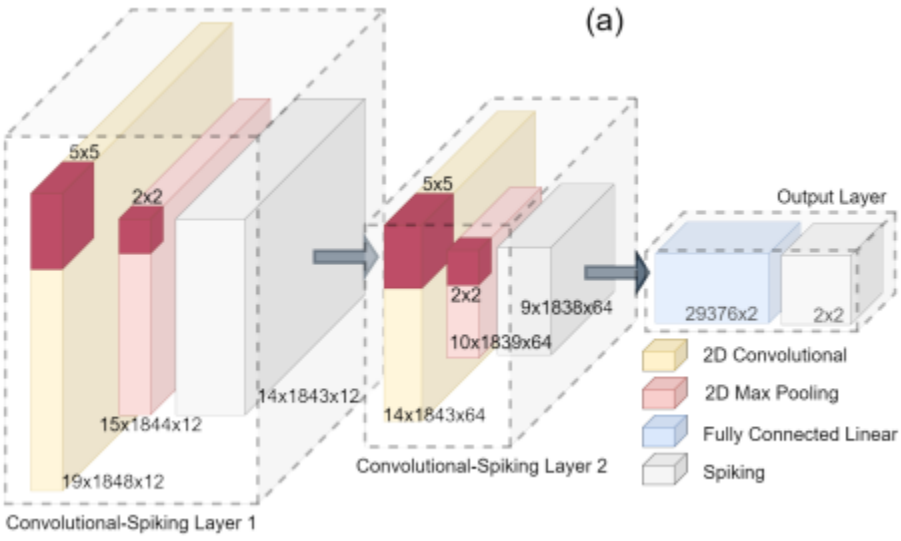

Figure 5: Example CSNN Architecture

Every convolutional sublayer had the same kernel size, 3, the same number of input and output channels, 64, and the same subsequent max pooling shape, 2x2. Each spiking sublayer is also uniform, where every LIF neuron has a voltage threshold of 0.9, a membrane time constant of 1.9, and an arc tangent surrogate activation function. The arc tangent (ATan) activation function can be looked at as a rescaled and shifted version of the sigmoid function which is symmetric about the origin. The ATan function is preferable over the sigmoid because it doesn't saturate as quickly at extreme input values, which helps mitigate the vanishing gradient problem.

4.3 Loss Function

We will use mean square error (MSE) loss to train the model and classification accuracy to measure its performance. Classification accuracy, which is given as a percentage, is just the number of correct predictions divided by the total number of predictions. Whereas mean square error is a common loss function used to measure the dissimilarity between the predicted output of a neural network and the actual ground truth labels. In a classification task, the goal is to assign one or more class labels to an input image. After processing an input image through the neural network, the model produces a probability distribution over all the possible classes using the equation below.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

This loss function penalizes larger errors more than smaller ones due to the squaring operation, which means that the model will try to reduce the magnitude of large prediction errors to improve its performance during training[12].

5 Results & Discussion

5.1 Results

Training the model can be inefficient and inaccurate depending on how it was configured, one would need to modify the command line arguments when executing the script in order to see a change in the model configuration. A decrease in runtime can be observed with a smaller batch size and a smaller number of epochs, but this obviously comes with a decrease in model performance. A large increase in model accuracy can be seen with a smaller learning rate, a larger batch size, and a larger number of epochs, but again this comes with a drop in computational efficiency. To balance the tradeoff between efficiency and accuracy I selected model parameter values roughly at the midpoint of these two extremes: a batch size of 8, 10 epochs, and a learning rate of 0.001. Training and testing the model with these moderately valued model parameters resulted in a test loss of ~0.028 and a test accuracy of 77.8%.

5.2 Benchmarking

5.2.1 Convolutional Artificial Neural Network

To put these metrics into context we need to benchmark our model against a convolutional artificial neural network trained on the same dataset. But due to time constraints we didn't train our own artificial model, instead we read about the work of others who had done this**.** They trained their network for over 100 epochs and achieved high accuracies between 90 to 100 percent. Our model's performance isn't really comparable to their network's due to the large difference in epochs, but it seems to be on the right track.

5.2.2 Von Neumann Architecture

It's important to note that pretty much all conventional computers use the Von Neumann architecture, including the laptop we used to train and test the CSNN. To exploit the benefits of SNNs with respect to computational efficiency, we would need to benchmark a model trained and tested on a specialized neuromorphic board against the one developed in this project .

5.3 Conclusion

The purpose of this project was to illuminate the massive gap in communication between man and machine for a specific real-world task, battle strategy. We addressed the issue by training a multi-class classifier on a hand gesture video dataset to predict the probabilities of each class. We chose a convolutional spiking neural network for our model to demonstrate the exciting potential of biologically plausible deep learning architectures and shine light on the field of neuromorphic computing as a whole.

To see an example of how a soldier-agent team can communicate via hand gesturing please see the appendix (section 7).

6 References

1. Gerstner, Wulfram, et al. Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition. Cambridge University Press, 2014. Accessed 15 August 2023.
2. E. O. Neftci, H. Mostafa and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks," in IEEE Signal Processing Magazine, vol. 36, no. 6, pp. 51-63, Nov. 2019, doi: 10.1109/MSP.2019.2931595.
3. A. Amir et al., "A Low Power, Fully Event-Based Gesture Recognition System," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 7388-7397, doi: 10.1109/CVPR.2017.781.
4. Lichtsteiner, Patrick & Posch, Christoph & Delbruck, Tobi. (2008). A 128× 128 120 dB 15 µs Latency Asynchronous Temporal Contrast Vision Sensor. Solid-State Circuits, IEEE Journal of. 43. 566 - 576. 10.1109/JSSC.2007.914337.
5. Benitez-Garcia, Gibran -, et al. "[2005.02134] IPN Hand: A Video Dataset and Benchmark for Real-Time Continuous Hand Gesture Recognition." arXiv, 20 April 2020, https://arxiv.org/abs/2005.02134. Accessed 15 August 2023.
6. By. "Intel Advances Neuromorphic with Loihi 2, New Lava Software Framework..." *Intel*, 30 Sept. 2021, https://www.intel.com/content/www/us/en/newsroom/news/intel-unveils-neuromorphic-loihi-2-lava-software.html#gs.306whr.
7. Laboratory, Los Alamos National. "Los Alamos National Laboratory Researchers Design New Artificial Synapses for Neuromorphic Computing." *LANL Discover RSS*, 1 June 2023, discover.lanl.gov/news/0601-artificial-synapses/.
8. Jeong, Jiwon. "The Most Intuitive and Easiest Guide for CNN." *Medium*, 17 July 2019, towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480#:~:text=It%20needs%20to%20be%20in,it%20to%20the%20next%20layer.
9. Olu-Ipinlaye, Oreolorun. "Batch Normalization in Convolutional Neural Networks." *Paperspace Blog*, 20 Oct. 2022, blog.paperspace.com/batch-normalization-in-convolutional-neural-networks/.
10. Brownlee, Jason. "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks." *MachineLearningMastery.Com*, 6 Aug. 2019, machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.
11. Mozafari, Milad, et al. "SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron." Frontiers, 31 May 2019, https://www.frontiersin.org/articles/10.3389/fnins.2019.00625/full.

7 Appendix

I actually spent a good portion of the semester trying to get the IPN Hand dataset working before switching to DVS128 Gesture due to memory issues. During that time I created something I call a *gesture map* to relate the hand gesture class labels to actions in my warzone environment. A soldier can use the *gesture map* to create a *gesture expression*, or a sequence of instructions to be executed by the robotic agent. There is a 1-1 mapping from IPN Hand labels to warzone actions, but there are 3 types of actions: expression formulation, task configuration, or tasks. I also define an *identity latent space*, or *identity interface*, which allows a soldier to configure the personality of a robotic agent. The *gesture map* and *identity space* are detailed in tables 1-4 below. The following is an example of how a soldier could instruct an agent in a warzone environment:

start gesture expression —> open identity interface —> walk northwest in identity latent space —> close identity interface —> toggle action interface —> define squadron of military vehicles as attention entity —> contract entity radius to focus only on one vehicle —> provide cover fire for entity —> toggle action interface —> end gesture expression

IPN Hand Class #	IPN Hand Class Label	Warzone Action
3	Single click with 1 finger	Start gesture expression
4	Single click with 2 fingers	Open identity interface
9	Open twice	Toggle action interface
10	Double click with 1 finger	End gesture expression
11	Double click with 2 fingers	Close identity interface

Table 1: Action Type - Expression Formulation

IPN Hand Class #	IPN Hand Class Label	Warzone Action
1	pointing with 1 finger	define attention entity
2	pointing with 2 fingers	Traverse identity space in direction
12	Zoom in	finer task granularity (contract attention radius)
13	Zoom out	broader task granularity (expand attention radius)

Table 2: Action Type - Task Configuration

IPN Hand Class #	IPN Hand Class Label	Warzone Action
5	throw up	spy // inform : gather sensory info about env (terrain, enemies)
6	throw down	support // aid: provide rt cover fire, vehicles, weapons, ammo, medicine to troops
7	throw left	storm the beach // aggress : attack & capture a base (boldness, trigger happiness)
8	throw right	hold the line // patrol: maintain a base (peruse perimeter less aggressively)

Table 3: Action Type - Tasks

Cardinal Direction	Personality Traits
North	very explorative (periphery robots equipped with various types of sensors),
East	very aggressive (weaponized, attacking robots),
South	very defensive (well-armored, crowd/enemy dispersal robots that transport cargo),
West	very covert (stealthy/agile/quick robots),
Northwest	stealthy, exploratory agent useful for gathering information from behind enemy lines

Table 4: Identity Latent Space