HDC AS AN ALTERNATIVE FOR SPIKING NEURAL NETWORKS

Margarita Geleta & Chase Overcash
Department of Computer Science
University of California at Irvine
Irvine, CA 92697, USA
{mgeleta,covercas}@uci.edu

1 INTRODUCTION

Even though *Artificial Neural Networks* (ANNs) have been hisorically brain-inspired and tried to mimic the behavior of biological neurons, ANNs are not biologically accurate indeed (Tavanaei et al., 2019). Biological and artificial neurons are fundamentally different in:

1. **Structure**: in general, in ANNs all neurons between consecutive layers are interconnected, resulting in *dense* layers. This is not the case of biological neurons, which have specific connections (called *synapses*) and random connections between unrelated neurons (e.g., induced by psychedelic drugs, Ly et al. (2018)) can lead to unexpected results.
2. **Neural computations**: in general, the inputs for ANNs are continuous signals (e.g., image and audio data), which, in a biological system, it is the input of the sense organs. However, the sense organs process these continuous inputs and transform them into discrete signals called *spike trains* which are sent to the biological neurons. That is, the *hidden units* of an ANN are continuous, while the *action potential* of a biological network is a discrete signal.
3. **Learning rule**: addtionally, the activation functions used in ANNs are in general differentiable, whereas in a biological neuron a function closest to a threshold is used, which is non-differentiable, by definition. For this reason, the training of biological neurons is compromised and this fact has prevented to popularize biological networks.

The closest approximation of biological neural network are the *Spiking Neural Networks* (SNNs), which define biologically-realistic neural models, merging neuroscience and machine learning – an external stimuli is captured by a neuromorphic sensor which in its turn sends the captured stimuli to a neuromorphic encoder. The neuromorphic encoder transforms the input into spike trains and those serve as input to the neuromorphic processor, the SNN (Kaiser et al., 2020).

Neuromorphic data, which is data that mimics the brain's input encoding and coordinates data to time, can only be handled meaningfully by spiking machine learning algorithms like SNNs and it is not evident how neuromorphic data can be processed by a non-spiking approach (Hersche et al., 2020). However, as another brain inspired model, *Hyper-Dimensional Computing* (HDC) does lend itself to brain-inspired data and it can do so using digital synchronous accelerators unlike a SNN. Therefore, HDC is a feasible alternative to SNNs for neuromorphic data.

2 METHODS

The source code has been developed in python. Git, as a version control system, has been used to track the code base development, and the project itself is hosted on GitHub and is publicly accessible[1].

1. **Datasets**: Neuromorphic data differs from typical data as it is event based and therefore tracks relative time. For neuromorphic data to be handled by a non Spiking ML algorithm, we would need a paradigm that can transform asynchronous sparse events to a condensed temporal representation, high-dimensional computing fits these requirements. For this paper we used the multi-vehicle stereo event camera (MVSEC) dataset. This dataset consisted of 2 mounted DAVIS cameras, a pair of VI sensors, a couple of DAVIS IMUs and a Velodyne Lidar stereo based system. For this implementation we are mainly concerned with the events tracked by the DAVIS dynamic vision sensor (DVS) systems and the position tracking sensors that allow us to determine the velocity.

[1] https://github.com/Chase-Overcash-UCI/SpikingNeuralNetwork

2. **Preprocessing**: Given an 2D matrix of events that store the x position, y position, the timestamp of when event took place and the polarity of the place we can construct the event-count image, a histogram that represents the number of event occurrences at each pixel during a given time period, $T \in \mathbb{R}^{n_x \times n_y}$:

$$I(x, y) = \sum_{t_k \in T} \delta(x - x_k, y - y_k) \tag{1}$$

Where $\delta(.,.)$ is the 2D Kronecker delta function and event polarity is not taken into consideration. An example of the usage of the event-count image could be: $I(3, 5) = 4$ which would mean that for some time period, *T*, an event occurred at pixel location (3, 5) a total of 4 times. Using the event-count image we can then construct the time image, which is a $n_x \times n_y$ sized matrix that averages the timestamp of the events that occurred at each pixel over the time period *T*:

$$T(x, y) = \frac{1}{I(x, y)} \sum_{x_k = x, y_k = y} t_k \tag{2}$$

The time image is very useful as it will allow for gradient image computations to obtain G_x and G_y. These gradient images are integral for feature calculation. The first two features can be obtained by calculating the sum of all gradients in G_x and G_y. The next two features are the sums of element-wise inverses of gradients in G_x and G_y. The last two features require finding the sum of the gradients in both directions multiplied by their relative position to the center of the image. Finally, the sum of the ratios between the l_2-norm of G_x and G_y and the euclidean distance between each pixel's location and the center of the image's location. These features can be used for velocity calculation and will be used to define our model for learning.

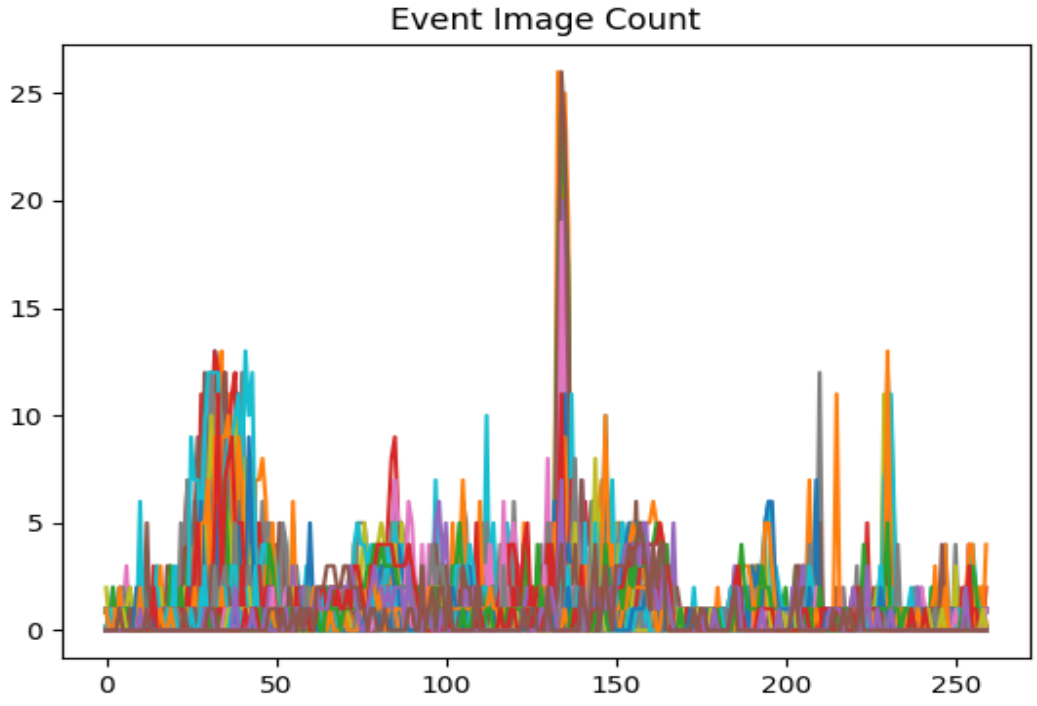
3. **Training**: Now that features have been found, the next step is to create hyper vectors that represent the time period of each event, that correspond to a certain velocity output. We permute these hyper vectors before bundling them based on those defining velocities. To do this we rounded the velocities to the nearest value in the array: [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]. Adaptive bundling was used for smarter learning. In this case when creating the hyper-vectors that define each velocity classifier, the percentage of how much of a vectors value was added to that hyper vector is determined so that the weight of the values added are more if they closely resemble the current feature class hyper vector. This similarity is determined using the cosine spatial distance which can be seen in formula 3.

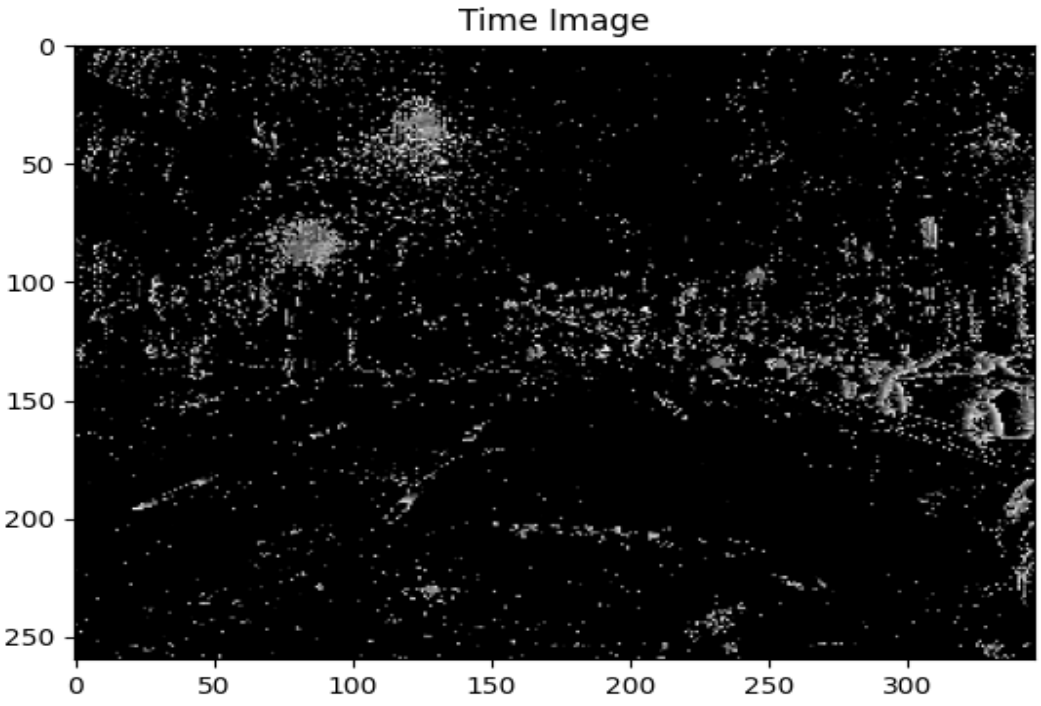
4. **Testing**: 10 percent of the data that is obtained through the gradient images are set assign as the testing data set. The algorithm checks the cosine spatial distance between the test hyper vectors and the bundled hyper vectors in order to estimate the velocity. The cosine spatial distance formula is defined as such:

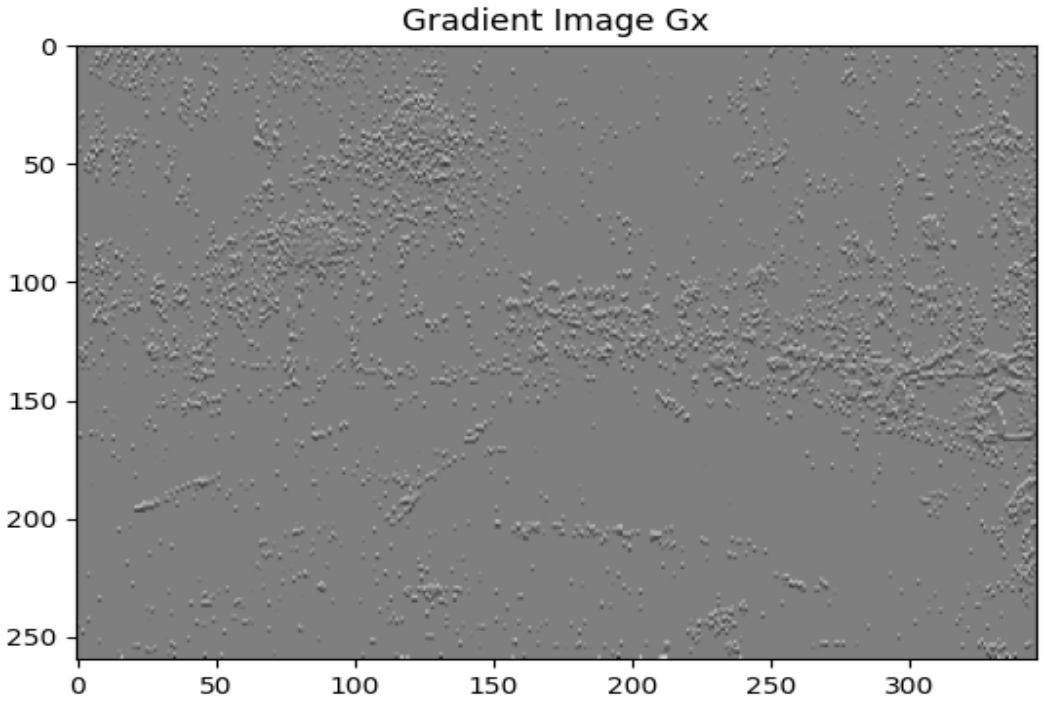
$$1 - \frac{u \cdot v}{||u||_2 ||v||_2} \tag{3}$$

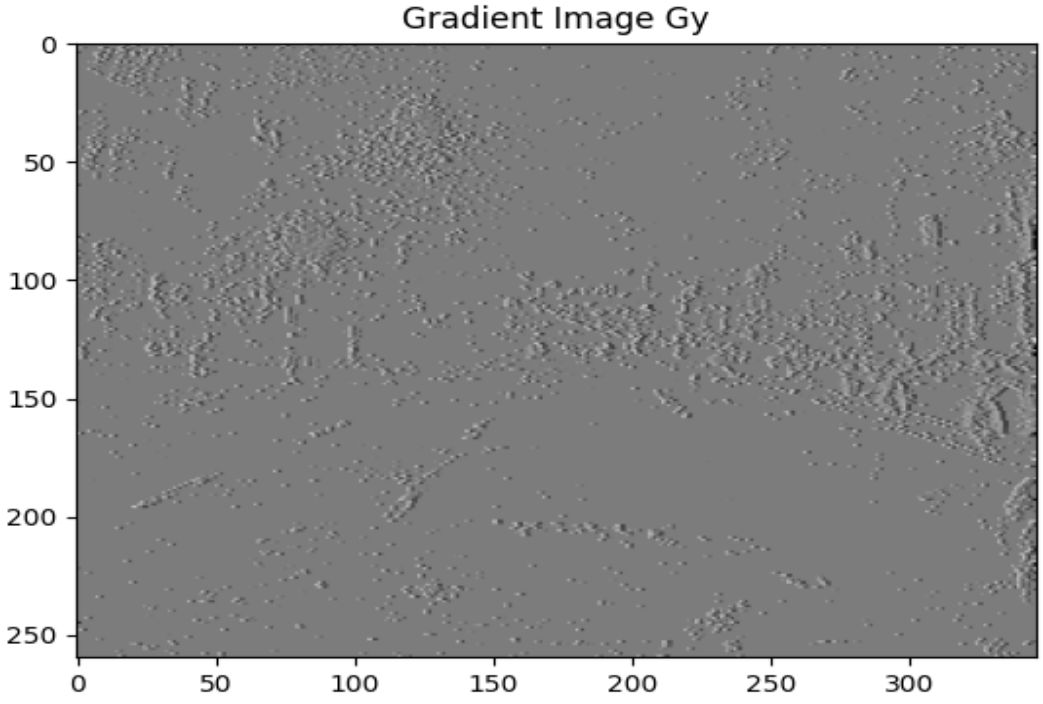
3 EXPERIMENTS

3.1 PRE-PROCESSING RESULTS






Gradient Image Gx
0
50
100
150
200
250
0
50
100
150
200
250
300


Gradient Image Gy
0
50
100
150
200
250
0
50
100
150
200
250
300

3.2 TESTING

While pre-processing would demonstrate a promising understanding, testing results came up less than satisfactory. Only maintaining approximately a 12% accuracy using single-pass adaptive training and closer to 10% with purely single-pass. Given that there is 10 potential classifiers, this does only a little better than guessing arbitrarily. While this is clearly not ideal, there is reason to believe that given a few tweaks to our feature calculation as we continue to work to improve the implementation down the line, that the accuracy has the capacity to be improved significantly.

4 CONCLUSION

In this project we unraveled the power of HDC for neuromorphic data. The classification results were unfortunately lack luster. We had limited computational resources and time, we are sure that if we had larger allowances, we could push far closer to the successes of related works. We believe that we were on the cusp of making a break through and intend to continue to pursue our original goals after the date of this paper's submission. We believe that the lack of success came from incorrect feature calculation, despite a strong pre-processing performance. Given additional time and research, readjusting our feature calculations further could result in a jump of over 90% accuracy, which would more closely resemble the results of similar, published works. But even with these constraints, there was a lot that was learned from this project, most specifically in regards to neuromorphic data, how to handle it, why handling it requires a different, more intricate approach than non-neuromorphic data, which compliments our now improved insight on what neuromorphic data, spiking machine learning, and high-dimension computing are capable of. Regardless of the end results, the ability for HDC to handle neuromorphic data does show promise that is worth investigating further, especially given that the future of spiking neural networks remains unclear, and having a potential suitable alternative is therefore abundant in importance in order to expand the applications of neuromorphic data, given that most works on SNNs are theoretical or lacking in results that can rival a simple fully-connected 2nd generation network. Both party members put in an equivalent amount of work.

REFERENCES

Michael Hersche, Edoardo Mello Rella, Alfio Di Mauro, Luca Benini, and Abbas Rahimi. Integrating event-based dynamic vision sensors with sparse hyperdimensional computing: A low-power accelerator with online learning capability. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '20, pp. 169–174, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370530. doi: 10.1145/3370748.3406560. URL https://doi.org/10.1145/3370748.3406560.

Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14, 2020. ISSN 1662-453X. doi: 10.3389/fnins.2020.00424. URL https://www.frontiersin.org/article/10.3389/fnins.2020.00424.

Calvin Ly, Alexandra C Greb, Lindsay P Cameron, Jonathan M Wong, Eden V Barragan, Paige C Wilson, Kyle F Burbach, Sina Soltanzadeh Zarandi, Alexander Sood, Michael R Paddy, et al. Psychedelics promote structural and functional neural plasticity. *Cell reports*, 23(11):3170–3182, 2018.

Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.