Exploring Expansion and Sparseness for Efficient Spiking Neural Network Architectures: Insights from the Olfactory System

Authors:
Victor Petre (s1097736),
Brandon Chin-A-Lien (s1116012)
Andrei Lixandru (s1053555)
Laurens van Rongen (s1010314)

Faculty of Social Sciences
Radboud University
Nijmegen, 6525XZ

Abstract

The historical influence of hardware on Deep Learning architecture design led us to consider whether the olfactory nervous system offers insights into efficient architectures for Spiking Neural Networks (SNNs). The discrete, combinatorial nature of olfactory stimuli seems particularly compatible with the event-based, brain-like characteristics of SNN chips. In *Drosophila*, projection neurons connect sparsely to Kenyon cells, which outnumber the projection axons and are pivotal in olfactory learning and memory. It is hypothesized that this expansion offers computational advantages, enhancing odor classification. Our approach is to validate this hypothesis in SNNs by training them on an olfactory classification task with varying degrees of expansion. We discover that, particularly with local learning rules, expansion indeed seems to bolster accuracy in SNNs. This demonstrates how insights from neuroscience can still inform neuromorphic hardware development.

1 Introduction

1.1 The case for neuromorphic hardware, and current unknowns

The rise of deep learning has significantly increased the demand for computational resources. Historically, this growth was largely guided by Moore's law and Dennard scaling. However, the deceleration of Moore's law and the cessation of Dennard scaling have shifted the focus to domain-specific architectures, or accelerators, to meet the escalating computational demands. Google's Tensor Processing Units are a prime example of such accelerators, tailored specifically for deep learning applications (Hennessy and Patterson, 2019) Various other accelerators are in development, most of which employ the von Neumann Architecture, characterized by a central processing unit (CPU) and separate memory units. This architecture necessitates data transfer between the memory unit and CPU for processing, which has been optimized historically but now faces limitations, particularly in memory bandwidth and power consumption.

In response, neuromorphic hardware seeks to emulate brain-like principles in chip design, offering potential solutions to these limitations. A subset of neuromorphic architecture, incorporating spiking neural networks (SNNs) into the hardware, replaces traditional CPU and memory units with neuron-like components that perform both computation and memory tasks. This design, mimicking the brain's event-based and asynchronous communication, leads to highly parallel, energy-efficient

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

computation. However, SNNs are significantly more challenging to train than traditional artificial neural networks (ANNs). This is both due to incompatibilities between SNNs and gradient descent (e.g. non-differentiable activation functions), as well as SNNs' much higher training cost in general (Tavanaei et al., 2019). As a result, SNNs have yet to gain widespread adoption in deep learning, and there is limited literature on how to best design an SNN to solve a particular problem of interest. There is therefore significant value in formally exploring the impact of architecture and training decisions on SNN performance.

1.2 Finding the right inductive bias

A crucial choice in successful architecture design is the selection of the right inductive bias. Inductive biases are assumptions in model design which affect how well it learns a task, and the biases differ in the strength of their assumptions about the underlying data (Sutton, 2019; Richards et al., 2019). While models with extremely strong inductive biases were popular (e.g. Riesenhuber and Poggio (1999)'s HMAX, a manually-crafted vision model based on the visual cortex), many modern models employ neural network architectures with broader, less restrictive assumptions. Here, the assumptions stem from the network architecture. In practice, new architectures are implemented without clear understanding of which inductive biases improve model performance. Identifying these biases is important, as it enables deliberate incorporation of architectural components that increase performance. A notable example is how recognizing the importance of sequence-independent dependency modeling in sequence transduction led to replacing recurrent networks with attention mechanisms, i.e. invention of Transformers (Vaswani et al., 2023). Similarly, our goal is to identify effective biases suitable for implementation in SNN chips.

We limit our search space of inductive biases by careful consideration of the benefits of SNN hardware. While inductive biases themselves are hardware-agnostic, the architectures that embody them are not. In turn, an architecture's success depends on its ability to fully leverage computational capabilities of the underlying hardware. For instance, the success of Transformers is not only due to their attention mechanism, but also their parallelizability on modern architectures (Vaswani et al., 2023). In the context of SNNs, we find it most useful to look for architectures that capitalize on their discrete, sparse and asynchronous nature. We claim that olfaction, both in machine learning and biology, provides valuable insights into effective model architectures. Olfaction is characterized by spatially sparse, discrete and combinatorial signals, and much prior research has already been done on the modeling the olfactory nervous in both classic biology and machine learning. While basing architectures on neuroscientific findings is a debated topic, we argue that such inspiration can be successful when focusing on computational principles rather than explicitly imitating biology. An example being how convolutional neural networks were deeply inspired from early neuroscience research (Goodfellow et al., 2016).

1.3 Olfactory classification as a model for studying inductive biases in SNNs

Olfactory classification has already been studied using ANNs by Wang et al. (2021), allowing comparisons to be made between the effects of the same inductive biases in ANNs and SNNs. To understand this point, the structure of the olfactory system in *Drosophila* will briefly be discussed.

Olfactory classification in *Drosophila* begins with olfactory receptors (ORs) on the surface of olfactory receptor neurons (ORNs) on the antennae, onto which chemical compounds bind. Each ORN expresses only one of 50 different OR types, and all ORNs expressing the same OR type innervate separate loci known as "glomeruli" within the fly brain (one glomerulus for each ORN type). Projection neurons (PNs) from each glomerulus then send axons to Kenyon cells (KCs) in the mushroom body, where the sensory information is used to form associative memories and influence behavior. PN-KC connectivity is generally unstructured and quite sparse, with one KC receiving input from $\sim$4-10 PNs (Wang et al., 2021). Wang et al. trained a fully-connected feedforward ANN to classify simulated combinations of OR activations into multiple odor classes, and found that the network connectivity after training mirrored the expected anatomical connectivity. The network contained 4 layers, each representing (in order) ORs, ORNs, PNs and KCs, with an additional 5th linear readout layer being used for classification. Layer sizes were 50, 500, 50, and 2500, respectively. Post training, the authors found the following:

1. OR-ORN connectivity corresponded to each ORN only "expressing" one OR type. This was only found when OR-ORN weights were constrained to be greater than 0.
2. ORN-PN connectivity was consistent with a glomerular structure. This was quantified using a "GloScore" (see Section 2), and was again found only when ORN-PN weights were constrained to be greater than 0.
3. PN-KC projections were sparse ($\sim$3-7 PN inputs per KC, with this value being given the name "K", see Section 2) regardless of PN-KC weight sign constraints.
4. Accuracy was unaffected by the aforementioned weight sign constraints.
5. Number of KC neurons was crucial for both performance and GloScore. Neither increased when the KC layer had more than 2500 neurons, but both accuracy and GloScore decreased significantly when fewer than 2500 neurons were present. Given the much smaller size of the PN layer (50 neurons, an "expansion ratio" of 1:50 for PN:KC), the authors suggest that the network's expansion of PN representations into the higher-dimensional space of the KC layer is crucial for performance.

The fact that constraining weights to be positive has no impact on performance suggests findings 1 and 2 to not be useful inductive biases towards solving the task in general. Rather, these properties appear to emerge in response to the specific training protocol (sign-constrained weights). However, finding 5 does represent a potential inductive bias to investigate, as it is not clear if dimensional expansion in the final layer will be equally beneficial to an SNN solving a comparable task.

Furthermore, the potential interplay between PN-KC expansion ratios and sparse PN-KC projections points towards another potentially useful inductive bias which is especially well-suited for neuromorphic hardware. Babadi and Sompolinsky (2014) examined the role of cortical sparsity and expansion ratios on classification accuracy. Cortical sparsity is inversely related to the fraction of active cortical neurons (KCs in the fly). Although cortical sparsity is not directly related to sparse PN-KC connectivity, a low connectivity with minimal overlap would produce high cortical sparsity. The authors found that high expansion ratios with high cortical sparsity increased classification accuracy in a linear model, but only if the cortical projections were established using Hebbian learning. When projections were initialized randomly, model performance got worse. According to the authors, this was likely because efficient sparse encoding requires projections to reflect the structure of the input ("structured projections"). Taken together, these findings suggest that Hebbian learning for the PN-KC weights might be a viable alternative to gradient-based methods. This is especially important considering that neuromorphic hardware is particularly well-suited for such local unsupervised learning (Tavanaei et al., 2019).

1.4 Research questions and design

It is possible that the low K-values ($\sim$3-7) in Wang et al.'s PN-KC projections reflect the model attempting to achieve appropriate cortical sparsity in KCs. It would be interesting to establish whether the interplay of structured projections, cortical sparsity and expansion ratios play a similar role in SNN models. In connecting these features to model classification accuracy, we may investigate the merit of these features as inductive biases in SNN models.

We first investigate whether the findings of Wang et al. can be replicated via SNNs, and whether PN-KC expansion ratios are relevant for performance. We achieve this by training SNNs with various PN-KC expansion ratios, using a backpropagation-based algorithm (DECOLLE, see Section 2).

We subsequently investigate the interplay of expansion ratios, cortical sparsity and structured projections. In line with the findings of Babadi and Sompolinsky (2014), we train additional SNNs with spike-timing-dependent plasticity (STDP, a form of Hebbian learning) between the PN-KC expansion layers, with DECOLLE training the rest of the network weights in a supervised manner. We track K to determine whether they remain sparse for SNNs. Assuming that the low K-values found by Wang et al. reflect the model attempting to learn the structured PN-KC projections which improved performance in Babadi and Sompolinsky (2014), it is possible that the use of STDP between the PN-KC layers might improve performance in this case. We train multiple networks in this manner, using the same PN-KC expansion ratios as the previous set of experiments, in order to investigate the interaction between expansion ratios and enforcing structured projections via STDP.

In all cases, we probe the networks' functional connectivity by computing GloScores and K-values.

2 Methods

2.1 Network implementation

The chosen network structures mirrored that of Wang et al., each using 5 fully-connected feedforward layers. As in the original paper, each layer represented one stage of processing within the olfactory circuit. From input to output, layers each represented ORs, ORNs, PNs , KCs and the readout layer. The sizes of the OR, ORN, PN and readout layers were constant across all networks trained, with 50, 500, 50 and 20 neurons each, respectively. These were the same sizes that Wang et al. used in their main experiments. The size of the KC layer varied across the experiments conducted (see Section 2.3). The network structure was broken into "blocks", where one block contains two layers of neurons and the weights between them.

All neurons were implemented using the CUrrent BAsed Leaky Integrate and Fire (CUBA-LIF) model within the `slayer` module of the `lava-dl` librarylava-nc (2024) . Blocks were implemented using the `slayer.block.cuba.Dense` module. All implementation parameters were left at the default values, unless otherwise stated. Parameters were based on those in the official `lava-dl` DECOLLE classification tutorial [1], as they led to networks which behaved well during classification training. All neurons were initialized with a current decay of 25% per time step and voltage decay of 3% per time step. All neurons not in the input layer had their threshold set to 1.25 (the recommended value in the DECOLLE tutorial), while input neurons used a threshold of 1.0. This latter value was selected for easy interpretability of firing behavior in response to the input data (see Section 2.2 for more information on how the dataset was created). In addition, surrogate gradient function parameters had to be specified for the neurons where gradient-based updates were performed. To this end, the scale of the `slayer` gradient function derivative was set to 3.0, with a time constant of 0.03.

Additional linear readout layers of size 10 were required both for classification following the KC layer, as well as for training of blocks via DECOLLE (see Section 2.3 for more detail). All of these were implemented using `torch.nn.Linear` modules.

2.2 Dataset creation

The original dataset in Wang et al. used randomly- and independently-generated values bounded between 0 and 1 to represent the activation of each of the 50 ORs. The complete dataset consisted of 1,000,000 randomly-sampled combinations of OR activations. Assigning odor labels to each combination was done by randomly generating 200 odor "prototype" vectors of shape (1, 50) within the OR activation space, which correspond to a detected odor. These prototypes were then randomly paired up into 100 odor "classes", which acted as the labels for the OR activation dataset. To assign a label to a combination of OR activations, the Euclidean distance between the OR activation vector and all of the prototype odors was computed, with the datapoint being assigned the label of the odor class which contained the prototype with the smallest distance to the OR activation vector. This pairing of prototypes into classes was done for two reasons. Firstly, there is no clear "mapping" between OR activation patterns and behavioral classification outcomes, in that drastically different odors can have similar OR activation patterns and vice versa (Malnic et al., 1999). Secondly, the task was otherwise trivial to solve without requiring a deep architecture.

These same principles were used in the current study to create a neuromorphic-compatible version of the dataset. We sampled 6000 combinations of activations for the 50 ORs, sampling each OR activation from a uniform distribution bounded between 0 and 1. One "datapoint" in the neuromorphic version of the dataset consisted of one combination of OR activations repeated over 150 timepoints, generating a tensor of shape (50, 150). When treated as input current to the neurons of the input layer (i.e. where each input neuron received one of the 50 input signals across 150 timepoints), the neuron's memory combined with the firing thresholds of 1.0 resulted in the input neurons developing a firing rate proportional to the magnitude of OR activation.

The 6000 sampled OR activation combinations were partitioned into a training set of 5000 samples, and a testing set of 1000 samples. 20 prototype odor vectors were sampled within the OR activation space, using a uniform distribution bounded between 0 and 1 as before. These 20 prototypes were paired into 10 odor classes, with the classes being used as ground truth labels for each of the 6000

[1] https://github.com/kclip/lava-decolle/blob/main/tutorials

examples as described previously. This is much fewer training examples and classes than the authors used in the original paper, but was necessary due to the lack of computational power required for such large-scale SNN training to be feasible.

2.3 Experiments

The first research interest was how the expansion ratio across PN-KC layers would influence classification accuracy and resulting functional connectivity in an SNN. To test this, 4 separate SNNs were trained on the above-mentioned dataset, using the previously specified sizes for OR, ORN, and PN layers. Across the 4 networks, the size of the KC layer was varied, with values of 50, 500, 2500 and 5000 (PN-KC expansion ratios of 1:1, 1:10, 1:50, and 1:100, respectively). These networks were trained end-to-end using DECOLLE, an algorithm designed for performing backpropagation on SNNs using synthetic gradients defined via local information (Kaiser et al., 2020). In essence, the algorithm treats each network block as an independent entity. Assuming an arbitrary block with an input and output layer, DECOLLE takes a linear readout of the output layer, and uses this readout as well as the desired classification outcome to compute a loss. This loss is then used to perform surrogate-gradient-based updates to the weights of the block. The weights of the readout layers are not updated through this process. When applying these updates to all of the network blocks, this effectively results in the network bringing the input closer and closer to the desired output with each layer. These concepts have been illustrated in Figure 1.

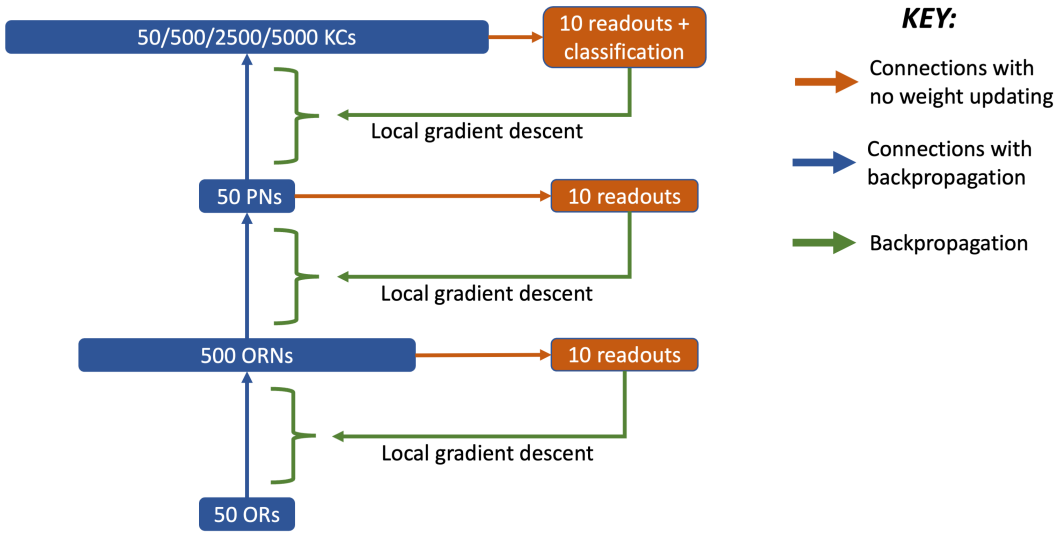

Figure 1: Visualization of the general network structure and training via DECOLLE. The number of neurons in the KC layer varied across the 4 training experiments. Throughout the training process, DECOLLE computes local gradient-based updates to the weights of each block, leaving the weights to the readout layers unchanged.

The second research interest was on the effect of implementing STDP for learning PN-KC projection weights on model accuracy and functional connectivity. As Babadi and Sompolinsky (2014) focus on Hebbian learning in the context of representational expansion layers (PN and KC, in our case), we chose to train 4 networks with the same 4 expansion ratios as above, using STDP to learn the weights of the PN-KC block. For the OR-ORN and ORN-PN blocks, DECOLLE was used again. As the learning within the PN-KC block was unsupervised, gradient-based updates were required for the final classification readout layer. For the rest of the readout layers (used by DECOLLE), weights were not updated. This hybrid training approach is referred to as DECOLLE+STDP from now on. For visual clarification, see Figure 2.

In comparing performance accuracy and functional connectivity across the 8 experiments outlined above, the interplay between representational expansion and STDP-based updating in expansion blocks could be studied, thereby elucidating their usefulness as general inductive biases.

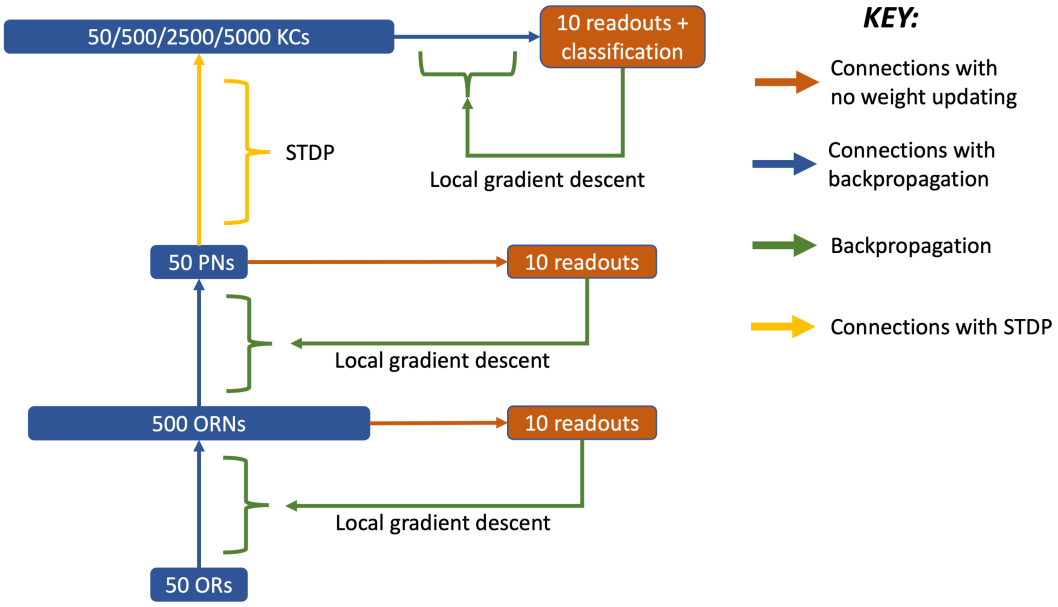

Figure 2: Visualization of the general network structure and training via DECOLLE + STDP. In contrast to Figure 1, the final readout layer requires training via backpropagation in order to use the STDP-learned representations for classification.

2.4 STDP implementation

As the `lava-dl` library had no support for STDP updating, this functionality had to be implemented from scratch. This implementation was based on the concept of an "update cycle", a property assigned to each weight in the PN-KC layer. An update cycle works in the following way. Upon network initialization, all update cycles are reset. Each weight then keeps track of the discrete timepoint when one of its two associated neurons (pre- or postsynaptic) first fires (the "start time"), thus initiating an update cycle for that weight. After this timepoint is logged, the weight logs when the *other* neuron (not the one which initiated the update cycle) fires (the "end time"). If the same neuron which initiated the update cycle fires again before the other neuron fires, the start time is changed to reflect this. When an end time is logged, a weight's update cycle is terminated. The number of discrete timepoints between the start and end times is calculated, with this value being positive if the presynaptic neuron fires before the postsynaptic neuron, zero if they fire at the same time, and negative otherwise. This value is then used in conjunction with the learning rule defined in Figure 3 below to compute a corresponding weight update, after which the update is applied, and a new update cycle for the just-updated weight begins. Although the entire PN-KC block is scanned for possible weight updates at each timepoint, weight updates are only calculated and applied for weights where an update cycle has just terminated.

Mathematically, the learning rule in Figure 3 is specified as follows:

$$\Delta w = \begin{cases} a^+ \exp(\frac{\Delta t}{\tau^+}), & \text{if } \Delta t \geq 0 \text{ (LTP)} \\ a^- \exp(-\frac{\Delta t}{\tau^-}), & \text{if } \Delta t < 0 \text{ (LTD)}, \end{cases} \tag{1}$$

with a^+, a^-, τ^+, τ^- taking values of $10^{-7}, -10^{-7}, -4, -10$, respectively. This specification was suggested by Gautam and Kohno (2021), and hyperparameter values were also selected following their recommendations.

2.5 Training

All networks were trained on the complete dataset for 30 epochs using a batch size of 256. For all DECOLLE-based gradient updates, a Smooth L1 loss was used, with an Adam optimizer using a learning rate of 0.0001. When STDP-based updates were required, the learning rule described

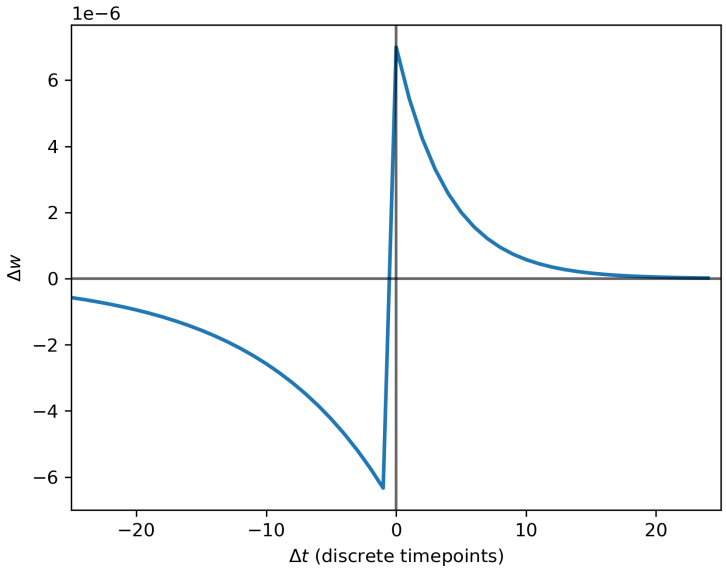

Figure 3: Learning rule used for STDP weight updating in the PN-KC block. Specified using Gautam and Kohno (2021).

in Section 2.4 was used for the PN-KC block as-is, with no additional superimposed learning rate. Before the forward pass of every training example, the voltages of all neurons were reset to 0, and a burn-in period was applied, where the first 50 timepoints of the current input were fed through the network with no weight updating. Following this burn-in period, weights were updated for all blocks at each of the remaining 100 timepoints of the input signal. Once all 150 timepoints of the current input had elapsed, the network was reset, and the next input was passed through.

Although Wang et al. found that constraining all network weights to be larger than 0 was necessary for post-training functional connectivity to reflect the fly neuroanatomy, the DECOLLE+STDP networks could not learn with this constraint. We were only able to achieve consistent learning via DECOLLE+STDP when only the weights of the ORN-PN block were enforced to be above 0. For consistency, all 8 networks across all conditions were trained with this constraint.

2.6 Quantifying functional connectivity

To draw parallels between our SNN implementations and the work of Wang et al., as well as to better understand the way the SNNs solve the task, measures for quantifying functional connectivity across layers were needed. In particular, we were interested in quantifying the extent to which ORN-PN connectivity reflected a glomerular structure, as well as the sparsity of PN-KC projections. The metrics Wang et al. used to quantify these properties were GloScore, and K-values, respectively.

GloScore computation first required establishing the "type" of each ORN. In neuroanatomy, each ORN only expresses one type of OR, thus determining the ORN's type. In the implementation, the type of an ORN was determined based on which of the 50 ORs had the largest weight projecting to the ORN. Then, for each PN, the weights of all 500 ORNs projecting to the PN were averaged by type, and ranked in decreasing order. The largest two averages were labeled w_1, w_2, and were used to calculate the GloScore for that particular PN using the following formula: $\text{GloScore} = (w_1 - w_2)/(w_1 + w_2)$. GloScores were then averaged across all PNs, generating the network's overall GloScore. This generates a score bound between 0 and 1, where higher GloScore values are more indicative of glomerular-like structure. In neuroanatomy, each glomerulus only receives projections from ORNs of the same type, which would receive a GloScore of 1.0.

K is designed to reflect projection sparsity within the PN-KC block. To compute it, the average number of PNs projecting to each KC was calculated. A PN was labeled as "projecting to a KC" when its weight exceeded a threshold of 1/N, where N is the number of KC neurons. This thresholding was

necessary, as no positive PN-KC weight was exactly 0. The lower the K-value, the fewer the number of PNs projecting to each KC, and the sparser the PN-KC projections overall.

3 Results

3.1 SNNs perform worse than ANNs in classification

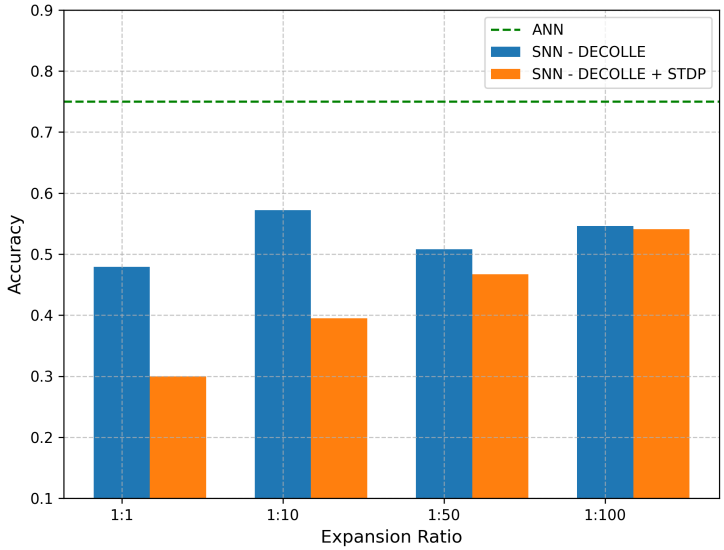

Figure 4: Classification accuracy of SNNs trained with DECOLLE and DECOLLE + STDP, across 4 PN-KC expansion ratios. The green line represents the accuracy of the ANN in Wang et al.

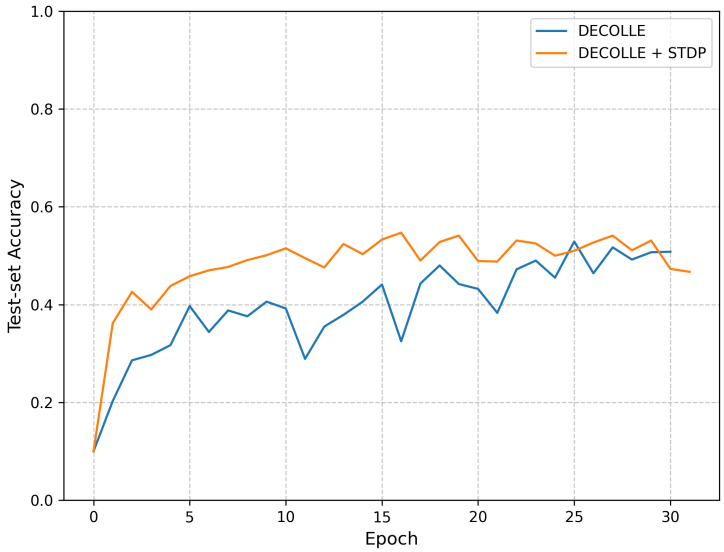

Figure 5: Example performance during training for DECOLLE and DECOLLE + STDP networks. These results were collected for the networks using a PN-KC expansion ratio of 1:50

When compared to the ANN used in Wang et al., our SNN shows inferior performance. The ANN achieves a testing accuracy of 75% on a 100-class olfaction classification task, while our versions plateau at 50% on a 10-class olfaction task. The distinction between the SNN trained only with DECOLLE and the one trained with DECOLLE + STDP is that the latter achieves 50% classification performance in half the time of the former. This can be seen in Figure 4 and 5. While convergence cannot be seen, performance of 50% is achieved more quickly and then drops for the DECOLLE + STDP model, while the curve for the DECOLLE-only model is still rising.

3.2 SNNs do not develop glomerulus-like connectivity

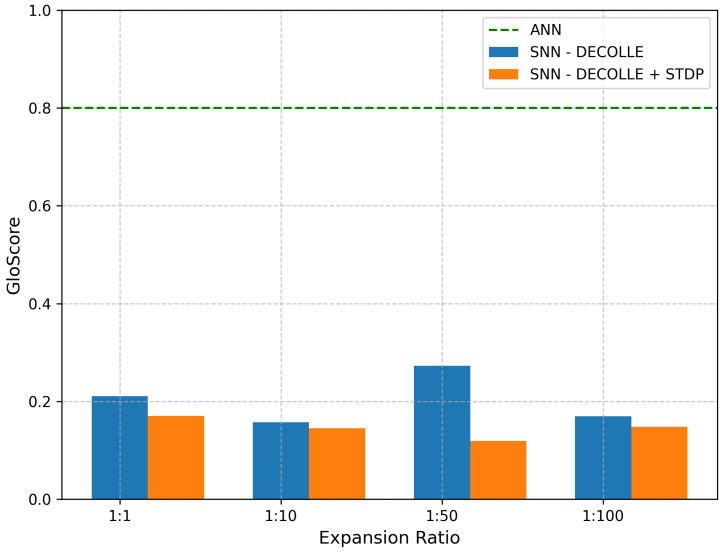

Figure 6: GloScores for SNNs trained with DECOLLE and DECOLLE + STDP, across 4 PN-KC expansion ratios. The green line indicates GloScores obtained for the ANN in Wang et al.

In Figure 6, we examine the emergent connectivity of our SNNs by extracting the GloScore of the ORN-PN connectivity for our range of expansion ratios. It can be observed that the GloScore of the ORN-PN connectome is somewhat dynamic across expansion ratios - ranging from 0.17 to 0.12, with a maximal value at 1:1 expansion. When compared to the GloScore obtained with an ANN for a 1:50 expansion ratio, our values are much lower, with an average discrepancy of around 0.6.

3.3 DECOLLE + STDP is sensitive to PN-KC expansion rates

For DECOLLE + STDP, accuracy was measured for an expansion ratio of 1:1, 1:10, 1:50 and 1:100. Respectively returning classification rates of 0.3, 0.4, 0.48 and 0.54. For just DECOLLE, classification rates were 0.48, 0.58, 0.51 and 0.54 (Figure x). Chance-accuracy lies at 10% classification rate.

Figure 7 shows the sparsity of PN-KC projections. It can be observed that the K-values of all SNNs are much higher (K $\approx$ 28-29) than that in a fly olfactory circuit (K = 6) and that of Wang et al.'s ANN (K = 3-7). It can also be observed that K is not significantly affected by a change in PN-KC expansion ratio for DECOLLE and STDP, whereas a noticeable change can be seen for the SNNs only using DECOLLE.

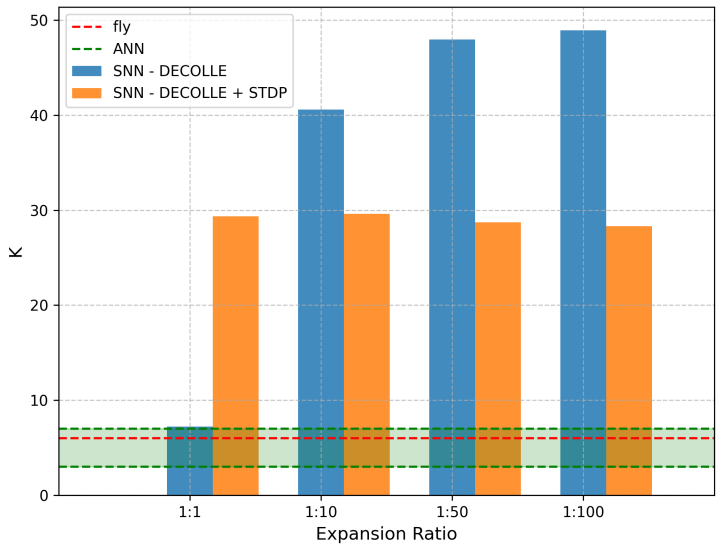

Figure 7: K-values for PN-KC connectivity for SNNs trained using DECOLLE and DECOLLE + STDP, across 4 PN-KC expansion ratios. Green area represents the range of K-values found in the ANN trained by Wang et al., with the dashed red line indicating the K-value of the fly neuroanatomy.

4 Discussion

This study aimed to evaluate the performance of SNNs in olfactory classification tasks. Additionally examining whether activity in KCs remains sparse and how expansion rate influences the performance of the DECOLLE+STDP model. Our results show that both DECOLLE+STDP and standard DECOLLE were less effective that Wang et al.'s model in terms of GloScore and accuracy. However, both models significantly outperformed chance-level accuracy. Consequently, we conclude that SNNs utilizing DECOLLE and DECOLLE+STDP are capable of odor classification, although not as effectively as ANNs. Additionally, the higher K-value in our SNNs compared to both the fly model and the ANN suggests less sparsity present in cortical activity for SNNs. Furthermore, the correlation of the expansion rate with accuracy in the DECOLLE+STDP model, despite a higher K-value, implies that structured projections, sparse cortical activity, and expansion can indeed enhance performance in deep SNNs, provided that this still constitutes sparse cortical activity.

The low GloScores of the model are not surprising, given that it was established the emergence of glomeruli is dependent on the weights being sign-constrained. Notably, the accuracy of both SNNs underperformed relative to the ANN. While DECOLLE is known to generally have lower performance than classic backpropagation, the inclusion of STDP seems to worsen this issue. In most scenarios, the combined DECOLLE+STDP model performed worse than DECOLLE alone. Furthermore, the use of STDP introduced additional hyperparameters which complicated training. It was challenging to find the values of a^+, a^-, τ_+, τ_- (see Section 2) which allowed the DECOLLE + STDP models to train at all. Due to the way STDP was implemented, certain weights would be potentiated indefinitely, leading to complete training failure when a^+ and a^- were too large. The reduced training time likely also contributed to low accuracy. Although our task was simplified, the original paper featured 200 times more training data and had five times as many epochs. We do believe that all these issues are easily addressed given more computational resources, and a systematic grid-search based approach for hyperparameter optimization.

Another important question is why K is so high for DECOLLE+STDP. It is possible that the cortical sparsity necessary to gain benefits from expansion is associated with a different K value in SNNs. This is somewhat supported by the fact that K for DECOLLE+STDP does not scale with expansion rate. In theory, K being constant means that as expansion rate grows higher, cortical sparsity actually increases because a smaller percentage of neurons are innervated. Comparing this with DECOLLE,

where K increases with expansion rate, implies the model has not attempted to maintain cortical sparsity.

In future studies, next to increasing training time, there are several improvements to consider. First, we currently use the K-value as a proxy for cortical sparsity. A more direct approach should be adopted, such as applying a threshold to activity within the KC layer. Additionally, to speed up the research process, exploring the role of expansion in ANNs could be advantageous. While the ultimate goal is to apply these findings to SNNs, beginning with ANNs might reveal robust effects that can inform subsequent studies in SNNs. Should future research confirm that expansion enhances ANN performance, it is crucial to understand why this occurs. Prior research proposes that expansion in the brain facilitates better feature discrimination by projecting stimuli into higher dimensions Babadi and Sompolinsky (2014). This implies that stimuli entering the "cortical layer" in models should undergo a similar increase in dimensionality. This leads to pivotal questions regarding the representation of the cortex in Deep Neural Networks and the practical methods usable to motivate models to apply this higher-dimensional expansion. Our current research underscores the ongoing relevance of Neuroscience in AI development, and we encourage future researchers to continue integrating these insights into their work.

5 Reflection

In our proposal we noted the advantages of the use of SNNs in processing visual information presented in Gallego et al. (2022). The intent was to recreate this transformation for processing olfactory information. However, the scope of this project was initially quite large, with the idea being uttered of simulating the olfactory system whole in SNNs. This idea was quickly abandoned due to time constraints. It was discovered soon after that Wang et al. (2021) had implemented an ANN for simulating the evolution of olfactory neurons in the brain. We thought it fit to recreate this using SNNs and track the difference in performance using different expansion ratios. This way an aspect of the olfactory system was simulated using SNNs synonymous to parts of the visual system that have already been simulated using SNNs.

Wang et al. (2021) seemed like the perfect paper to introduce a SNN to, especially given the biological nature of the principle of olfactory neuron evolution. An SNN translation of the methods of Wang et al. (2021) was made, and through multiple rounds of improvements the results section of this report was filled. These improvements were not easy as it was difficult to scale down the network enough to fit the devices used to train the network. Additionally troubles included finding correct hyperparameters for the STDP-rules and errors in the code that caused the need to rerun the entire training.

Initially there was an intent to use the model for real-life purposes if it proved to be superior to the ANN model. Although we do not believe the model to be superior to that of a deep feedforward network, there is still more work to be done as we are not using the full values for the network as presented in Wang et al. (2021). Due to this we cannot fully estimate superiority unless the original values for training were used. Another method could have been to use the values used in the SNN for another run using the ANN, however due to time-constraints we did not perform these tests.

In the end we are pleased with the fact that we managed to create a working spiking neural network for the simulation of olfactory classification.

References

Babadi, B. and Sompolinsky, H. (2014). Sparseness and expansion in sensory representations. *Neuron*, 83(5):1213–1226.

Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K., and Scaramuzza, D. (2022). Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180.

Gautam, A. and Kohno, T. (2021). An adaptive stdp learning rule for neuromorphic systems. *Frontiers in Neuroscience*, 15.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hennessy, J. L. and Patterson, D. A. (2019). A new golden age for computer architecture. *Commun. ACM*, 62(2):48–60.

Kaiser, J., Mostafa, H., and Neftci, E. (2020). Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14.

lava-nc (2024). Lava-DL: A Library for Deep Learning with Spiking Neural Networks in Lava.

Malnic, B., Hirono, J., Sato, T., and Buck, L. B. (1999). Combinatorial receptor codes for odors. *Cell*, 96(5):713–723.

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., de Berker, A., Ganguli, S., and et al. (2019). A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025.

Sutton, R. S. (2019). The bitter lesson. Accessed: 25-Jan-2923.

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks*, 111:47–63.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

Wang, P. Y., Sun, Y., Axel, R., Abbott, L., and Yang, G. R. (2021). Evolving the olfactory system with machine learning. *Neuron*, 109(23).