This paper has been accepted for publication at the
IEEE International Conference on Robotics and Automation (ICRA), Paris, 2020. ©IEEE

Event-Based Angular Velocity Regression with Spiking Networks

Mathias Gehrig[1], Sumit Bam Shrestha[2], Daniel Mouritzen[1], and Davide Scaramuzza[1]

***Abstract*— Spiking Neural Networks (SNNs) are bio-inspired networks that process information conveyed as temporal spikes rather than numeric values. An example of a sensor providing such data is the event-camera. It only produces an event when a pixel reports a significant brightness change. Similarly, the spiking neuron of an SNN only produces a spike whenever a significant number of spikes occur within a short period of time. Due to their spike-based computational model, SNNs can process output from event-based, asynchronous sensors without any pre-processing at extremely lower power unlike standard artificial neural networks. This is possible due to specialized neuromorphic hardware that implements the highly-parallelizable concept of SNNs in silicon. Yet, SNNs have not enjoyed the same rise of popularity as artificial neural networks. This not only stems from the fact that their input format is rather unconventional but also due to the challenges in training spiking networks. Despite their temporal nature and recent algorithmic advances, they have been mostly evaluated on classification problems. We propose, for the first time, a temporal regression problem of numerical values given events from an event-camera. We specifically investigate the prediction of the 3-DOF angular velocity of a rotating event-camera with an SNN. The difficulty of this problem arises from the prediction of angular velocities continuously in time directly from irregular, asynchronous event-based input. Directly utilising the output of event-cameras without any pre-processing ensures that we inherit all the benefits that they provide over conventional cameras. That is high-temporal resolution, high-dynamic range and no motion blur. To assess the performance of SNNs on this task, we introduce a synthetic event-camera dataset generated from real-world panoramic images and show that we can successfully train an SNN to perform angular velocity regression.**

Supplementary Matieral

Code is available at
https://tinyurl.com/snn-ang-vel

I. Introduction

A spiking neural network (SNN) is a bio-inspired model consisting of spiking neurons as the computational model. A spiking neuron is a mathematical abstraction of a biological neuron, which processes temporal events called spikes and also outputs spikes [1]. It has a one-dimensional internal state (potential), that is governed by first-order dynamics. Whenever a spike arrives, the potential gets excited but decays again if no other spikes are registered close in time. In case of the potential reaching a certain threshold, a spiking neuron emits a spike to connected neurons and resets its own potential. If we now link many neurons together we create a dynamical neural network that processes information with spikes rather than numeric values. This crucial difference is why SNNs and artificial neural networks (ANNs) are not necessarily competitors but rather models that are intrinsically suitable for a distinct set of problems. As an example, SNNs are able to process asynchronous, irregular data from event-based cameras directly [2], without pre-processing events [3] and at extremely low power [4]. We refer to the survey paper by Gallego et al. [5] for an introduction to event-based vision.

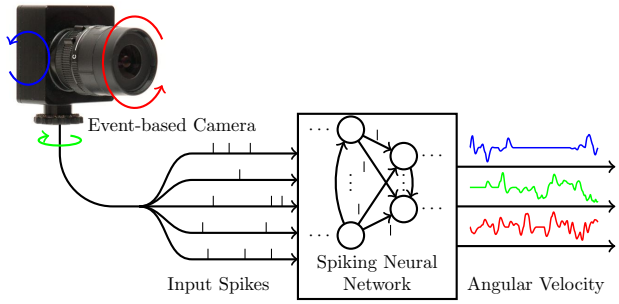


Fig. 1. Processing pipeline for event-based angular velocity regression using a spiking neural network.

Even training feedforward spiking neural networks is notoriously difficult. The main reason for this is that the spike-generation mechanism within a spiking neuron is non-differentiable. Furthermore, spikes have a temporal effect on the dynamics of the receiving neuron and introduce a temporal dimension to the error assignment problem. As a result, standard backpropagation [6] is not directly applicable to SNNs. Nonethelesss, the majority of research on supervised learning for SNNs has taken inspiration from backpropagation to solve the error assignment problem. However, some algorithms are only designed for a single neuron [7]–[9], ignore the temporal effects of spikes [10]–[13] or employ heuristics for successful learning [14]–[17] on small-scale problems. Although SNNs are a natural fit for spatio-temporal problems, they have largely been applied to classification problems [11]–[13], [18]–[23], except for a few demonstrations addressing learning of spike sequences (spike-trains) [13], [18], [24]. Therefore, it is unclear which algorithm can successfully train multi-layer architectures for tasks beyond classification.

[1]Mathias Gehrig, Daniel Mouritzen and Davide Scaramuzza are with the Robotics and Perception Group, Dep. of Informatics, University of Zurich, and Dep. of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland— http://rpg.ifi.uzh.ch. Their work was supported by the SNSF-ERC Starting Grant and the Swiss National Science Foundation through the National Center of Competence in Research (NCCR) Robotics.

[2]Sumit Bam Shrestha is with Temasek Laboratories, National University of Singapore, Singapore. His work is partially supported by Programmatic grant no. A1687b0033 from the Singapore governments Research, Innovation and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain)

A. Contributions

In this work, we explore the utility of SNNs to perform regression of numeric values in the continuous-time domain from event-based data. To the best of our knowledge, this problem setting has not been explored in SNN literature at the time of the submission. The framework is illustrated in figure 1. The task of our choice is angular velocity (tilt, pan, roll rates) regression of a rotating event-camera. Successful attempts to this task require a training algorithm that is able to perform accurate spatio-temporal error assignment. This might not be necessary for performing classification on neuromorphic datasets and, thus, raises a challenge for SNNs.

Our problem setting offers the context to approach a number of unanswered questions:

- How do we formulate a continuous-time, numeric regression problem for SNNs?
- Can current state-of-the-art SNN-based learning approaches solve temporal problems beyond classification?
- What kind of architecture performs well on this task?
- Can SNNs match the performance of ANNs in numeric regression tasks?

As a first building block, we introduce a large-scale synthetic dataset from real-world scenes using a state-of-the-art event-camera simulator [25]. This dataset provides precise ground truth for angular velocity which is used both for training and evaluation of the SNN. We use this dataset to successfully train a feedforward convolutional SNN architecture that predicts tilt, pan, and roll rates at all times with a recently proposed supervised-learning approach [18]. In addition to that, we show that our network predicts accurately at the full range of angular velocities and extensively compare against ANN baselines designed to perform this task in discrete-time.

In summary, our contributions are:

- The introduction of a continuous-time regression problem for spiking neural networks along with a dataset for reproducibility.
- A novel convolutional SNN architecture designed for regression of numeric values.
- A detailed evaluation against state-of-the-art ANN models crafted for event-based vision problems.

II. Related Work

Currently, artificial neural networks are the de facto computational model of choice for a wide range problems, such as classification, time series prediction, regression analysis, sequential decision making etc. Spiking neural networks add additional biological relevance in these architectures with the use of a spiking neuron as the distributed computational unit. With the promise of increased computational ability [26], [27] and low power computation using neuromorphic hardware [28]–[31], SNNs show their potential as computational engines, especially for processing event-based data from neuromorphic sensors [32], [33].

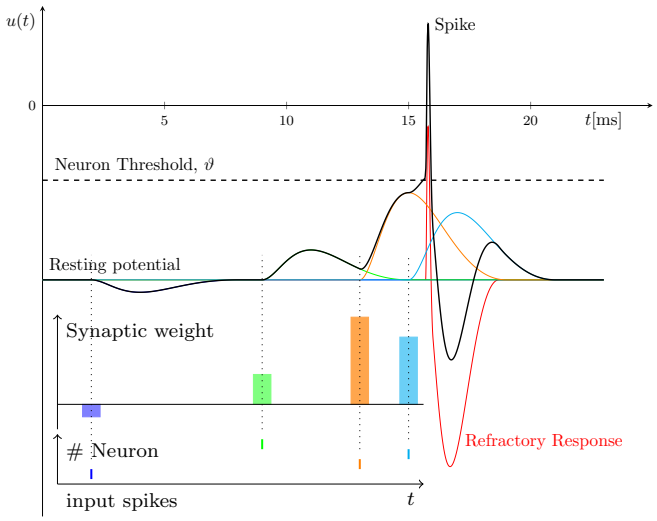


Fig. 2. Dynamics of a spiking neuron. A spiking neuron is exciting by incoming spikes according to the corresponding synaptic weights. If the potential reaches the neuron threshold, a spike is emitted and the potential of the neuron is reset by a refractory response.

One of the major bottlenecks in realizing the computational potential of SNNs has been the fact that backpropagation is not directly applicable to training SNNs. This is due to the non-differentiable nature of the spike generation mechanism in spiking neurons. Nevertheless, there have been some efforts in tailoring backpropagation for SNNs. Prominent examples are event-based backpropagation methods [14], [24], [34] that backpropagate error at spike-times. However, they have shown limited success. In recent times, the idea of using a continuous function as a proxy for spike function derivative has been used effectively [12], [13], [18], [19] for relatively deep feedforward SNNs. [18], [19] also take into account the temporal dynamics present in SNNs to assign error in time. Still, gradients can only be computed approximately with these methods. As a result, it is unclear whether there are better performing algorithms for supervised learning for feedforward SNNs.

Almost all of the reported use cases of SNNs in the aforementioned methods are classification problems, such as image classification [35], [36], neuromorphic classification [37], action recognition [38], etc. When it comes to regression problems, there are demonstrations of toy spike-to-spike translation problems [13], [18] for which a target spike-train is learned. To the best of our knowledge, there is currently no published work exploring the use of SNNs for predicting numeric values in continuous-time.

III. Methodology

A. Spiking Neurons

Spiking neurons model the dynamics of a biological neuron. They receive spikes, which are short pulses of voltage surge, and distribute their effects in time to form a post-synaptic potential (PSP). The magnitude and sign of the PSP is determined by the synaptic weight corresponding to the spike. Finally, the accumulation of all the PSPs in a neuron

constitutes the sub-threshold membrane potential $u(t)$. This process is illustrated in figure 2 via spikes from multiple synapses. When the sub-threshold membrane potential is strong enough to exceed the neuron threshold ϑ the spiking neuron responds with a spike. Immediately after the spike, the neuron tries to suppress its membrane potential so that the spiking activity is regulated. This self-suppression mechanism is called refractory response.

There are various mathematical models in neuroscience that describe the dynamics of a spiking neuron with varying degree of detail: from the complex Hodgkin-Huxley neuron [39] to the simple Leaky Integrate and Fire neuron [1], [40]. In this paper, we use the Spike Response Model (SRM) [41]. In SRM, the PSP response is a decoupled, normalized spike response kernel, $\varepsilon(t)$, scaled by the synaptic weight. Similarly, the refractory response is described by a refractory kernel, $\nu(t)$. The SRM is simple, yet versatile enough to represent various spiking neuron characteristics with appropriate spike response and refractory kernels.

B. Feedforward Spiking Neural Networks

In this section, we define the model of feedforward SNNs and describe how events and spikes are related.

One of the advantages of SNNs over ANNs is their ability to process event-data from event-cameras directly. Event-cameras have independent sensors at each pixel that respond asynchronously to brightness changes. An event can be described by a tuple (x, y, t, p), where x and y are the location of the pixel from which the event was triggered at time t. The polarity p is a binary variable that indicates whether the change in brightness is either positive or negative. The SNN model in this work has two inputs (i.e. two channels) for each pixel location to account for the polarity of the events. When an event is fed as an input to the network, we refer to it as *spike*. A sequence of spikes is called *spike train* and is defined as $s(t) = \sum_{t^{(f)} \in \mathcal{F}} \delta(t - t^{(f)})$, where $\mathcal{F}$ is the set of times of the individual spikes.

Our SNN model is a feedforward SNN with n_l layers. In the following definition, $\boldsymbol{W}^{(l)}$ are the synaptic weights corresponding to layer l and $\boldsymbol{s}_{\text{in}}(t)$ refers to the spikes of the input layer:

$$\boldsymbol{s}^{(0)}(t) = \boldsymbol{s}_{\text{in}}(t) \tag{1}$$

$$\boldsymbol{u}^{(l+1)}(t) = \boldsymbol{W}^{(l)}(\varepsilon * \boldsymbol{s}^{(l)})(t) + (\nu * \boldsymbol{s}^{(l+1)})(t) \tag{2}$$

$$\boldsymbol{s}^{(l)}(t) = \sum_{t^{(f)} \in \{t | \boldsymbol{u}^{(l)}(t) = \vartheta\}} \delta(t - t^{(f)}) \tag{3}$$

$$\boldsymbol{\omega}(t) = \boldsymbol{W}^{(n_l)}(\varepsilon * \boldsymbol{s}^{(n_l)})(t) \tag{4}$$

where $\boldsymbol{\omega}$ is the prediction of the angular velocity. We use the following form of spike response kernel and refractory kernel:

$$\varepsilon(t) = \frac{t}{\tau_s} e^{1 - \frac{t}{\tau_s}} \mathcal{H}(t) \tag{5}$$

$$\nu(t) = -2\,\vartheta e^{-\frac{t}{\tau_r}} \mathcal{H}(t) \tag{6}$$

$\mathcal{H}(\cdot)$ is the Heaviside step function; τ_s and τ_r are the time constants of spike response kernel and refractory kernel respectively.

Note how the spike response kernel distributes the effect of input spikes over time (eqs. (2) & (5)), peaking some time later and exponentially decaying after the peak. This temporal distribution allows interaction between two input spikes that are within the effective temporal range of the spike response kernel, thereby allowing short term memory mechanism in an SNN. It is pivotal in allowing the network to estimate the sensor's movement and enables prediction of angular velocity.

C. Network Architecture

Our network architecture is a convolutional spiking neural network losely inspired by state-of-the-art architectures for self-supervised ego-motion prediction [42]. It consists of five convolutional layers followed by a pooling and fully connected layer to predict angular velocities. The first 4 convolutional layers perform spatial downsampling with stride 2. At the same time, the number of channels is doubled with each layer starting with 16 channels in the first layer. Table I shows these layer-wise hyperparameters in more detail. It can be seen that there is another set of hyperparameters that are time constants concerned with the decay rate of the spike response and refractory kernels in equation (5) and (6). These time constants are increasing with network depth to account for both high event rate from the event-camera at the input and slower dynamics at the output for consistent predictions. Table I lists these layer-wise hyperparameters of the architecture in more detail.

1) Global Average Spike Pooling (GASP): So far, the discussed elements of our architecture are regularly encountered in literature of both spiking and artificial neural networks. In recent years, global average pooling [43] has become prevalent in modern network architectures [44], [45] due to their regularization effect. We adapt this line of work for spiking neural networks and introduce global average spike pooling after the last convolutional layer.

To describe GASP, we define a spatial spike-train $\boldsymbol{S}_i(t, x, y)$ resulting from the i-th channel of the previous layer as

$$\boldsymbol{S}_i(t, x, y) = \sum_{t^{(f)} \in \mathcal{F}_i(x,y)} \delta(t - t^{(f)}), \tag{7}$$

where $\mathcal{F}_i(x, y)$ is the set of spike times in the i-th channel at the spatial location (x, y). Let $g_i(t)$ be the i-th output of the pooling operation, then

$$g_i(t) = \sum_{\substack{x \in \{0,\dots,W-1\} \\ y \in \{0,\dots,H-1\}}} \boldsymbol{S}_i(t, x, y), \tag{8}$$

where W and H are width and height of the previous channel. Successive synapses connected to the spike-train $g_i(t)$ are then scaled by $1/W \cdot H$ to introduce invariance with respect to the spatial resolution.

After the spike-train pooling, a fully connected layer connects the spike-trains to three, non-spiking, output neurons

for regressing the angular velocity continuously in time. To summarize the computation after the pooling layer, we can reformulate the angular velocity prediction in equation (4) as

$$\boldsymbol{\omega}(t) = \frac{1}{N}\left(\varepsilon * \boldsymbol{W}\left[g_1, \ldots, g_C\right]^\top\right)(t) \tag{9}$$

where $N = W \cdot H$ is the number of neurons per channel in the last convolutional layer with C channels.

D. Synthetic Dataset Generation

Supervised learning of spiking neural networks requires a large amount of data. In our case, we seek a dataset that contains events from an event-camera with ground truth angular velocity. The three main criteria of our dataset are the following: First, there must be a large variety of scenes to avoid overfitting to specific visual patterns. Second, the dataset must be balanced with respect to the distribution of angular velocities. Third, precise ground truth at high temporal resolution is required. To the best of our knowledge, such a dataset is currently not available. As a consequence, we generate our own dataset.

To fulfill all three criteria, we generated a synthetic datasets using ESIM [25] as an event-camera simulator. ESIM renders images along a trajectory and interpolates a brightness signal to yield an approximation of the intensity per pixel at all times. This signal is then used to generate events with a user-chosen contrast threshold. We selected the contrast threshold to be normally distributed with mean 0.45 and standard deviation of 0.05. Furthermore, we set the resolution to 240×180 to match the resolution of the DAVIS240C event-camera [46].

As a next step, we selected a subset of 10000 panorama images of the Sun360 dataset [47]. From these images, ESIM simulated sequences with a temporal window of 500 milliseconds each. This amounts to approximately 1.4 hours of simulated data. The random rotational motion used to generate this data was designed to cover all axes equally such that, over the whole dataset, angular velocities are uncorrelated and their mean is zero.

Finally, the dataset is divided into 9000 sequences for training and 500 sequences each for validation and testing.

E. Loss Function

The loss function L is defined as the time-integral over the euclidean distance between the predicted angular velocity $\boldsymbol{\omega}(t)$ and ground truth angular velocity $\hat{\boldsymbol{\omega}}(t)$:

$$L = \frac{1}{T_1 - T_0}\int_{T_0}^{T_1} \sqrt{\boldsymbol{e}(t)^\top \boldsymbol{e}(t)}\,\mathrm{d}t \tag{10}$$

where $\boldsymbol{e}(t) = \boldsymbol{\omega}(t) - \hat{\boldsymbol{\omega}}(t)$. The error function is not immediately evaluated at the beginning of the simulation because the SNN has a certain settling time due to its dynamics. Note that this loss function is closely related to the van Rossum distance [48] which has been used for measuring distances between spike-trains.

F. Training Procedure

SNNs are continuous-time dynamical system and, as such, must be discretized for simulation on GPUs. In the ideal case, we choose the discretization time steps as small as possible for accurate simulation. In practice, however, the step size is a trade-off between accuracy of the simulation and availability of memory and computational resources. We chose to restrict the simulation time to 100 milliseconds with a time step of one millisecond. The loss is then evaluated from 50 milliseconds onwards to avoid punishing settling time with less than 50 milliseconds duration.

The training of our SNNs is based on first-order optimization methods. As a consequence, we must compute gradients of the loss function with respect to the parameters of the SNN. This is done with the publicly available[1] PyTorch implementation of SLAYER [18].

We augment the training data by performing random horizontal and vertical flipping and inversion of time to mitigate overfitting. The networks are then trained on the full resolution (240×180) of the dataset for 240,000 iterations and batch size of 16. The optimization method of choice is ADAM [49] with a learning rate of 0.0001 without weight decay.

IV. Experiments

In this section, we assess the performance of our method on the dataset described in section III-D to investigate the following questions:

- What is the relation between angular velocity and prediction accuracy?
- Are the predictions for tilt, pan and roll rates of comparable accuracy?
- Is our method competitive with respect to artificial neural networks?

A. Experimental Setup

For the purpose of evaluating the prediction accuracy of different methods, we split the test set into 6 subsets each containing a specific range of angular velocities. The test set itself is generated in identical fashion to the training and validation set. More importantly, the panorama images, from which the event data of the test set is generated, are unique to the test set. What makes this dataset especially challenging is the fact that the angular velocity is not initialized at zero but rather at a randomly generated initial velocity. The angular velocity slightly varies within the generated sequence but does not change drastically.

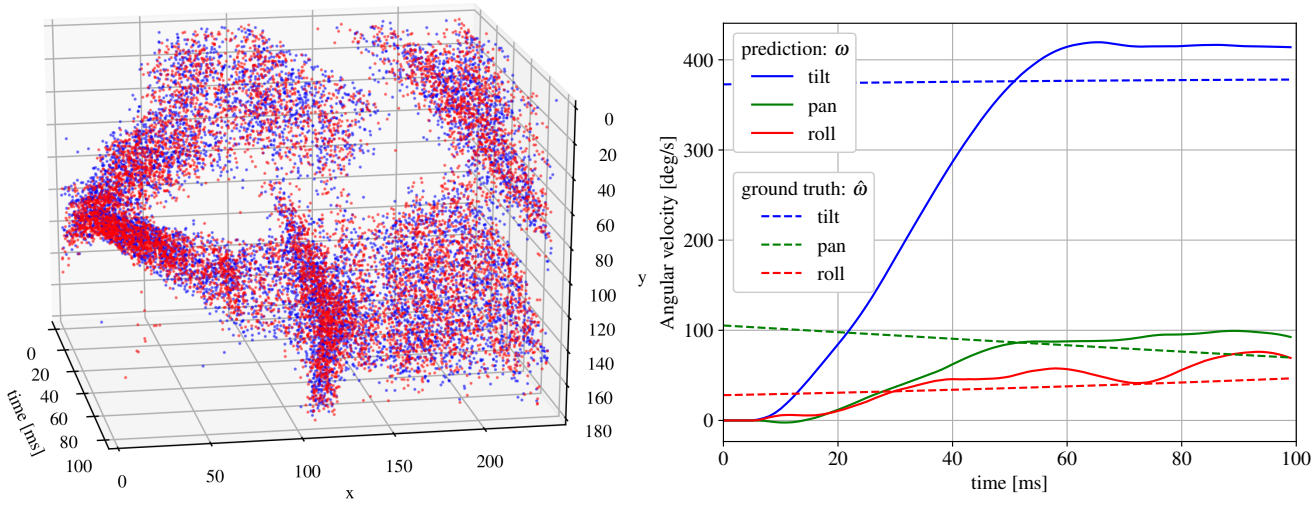
The SNN is simulated with a step size of 1 milliseconds but is only evaluated after 50 milliseconds to eliminate the influence of the settling time on the evaluation accuracy. This is in accordance to the training methodology discussed in section III-F. Figure 3b visualizes the prediction of the SNN over the 100ms sequence. As expected, the network only achieves good tracking after the settling time since it is not penalized for inaccuracies during settling time. Figure 3a also

[1] https://github.com/bamsumit/slayerPytorch

TABLE I

HYPERPARAMETERS OF THE SPIKING NEURAL NETWORK ARCHITECTURE.

Layer-type	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	Fully connected
Kernel size	3×3	3×3	3×3	3×3	3×3	-
Channels	16	32	64	128	256	-
Stride	2	2	2	2	1	-
τ_s, τ_r [ms]	2, 1	2, 1	4, 4	4, 4	4, 4	8, -



(a) (b)

Fig. 3. (a): Events over the 100ms test sequence. Positive events in red and negative events in blue. (b) Continuous-time angular velocity predictions by the SNN and the corresponding ground truth. The SNN requires a settling time of around 50 milliseconds which is exactly when the loss function is applied while training the network. 'Pred' refers to prediction and 'gt' refers to ground truth.

shows the space-time volume of events that are fed to the SNN for the same sequence.

We also compare our method against three feedforward artificial neural networks. Architecture ANN-6 is based on the same architecture as the 6-layer SNN (SNN-6) specified in table I (with ReLU activation functions). To examine the importance of deeper networks we train two ResNet-50 architectures [44]. The only difference between them is that one is trained with inputs consisting of two-channel frames computed by accumulating events [50], denoted by (a), while the other is trained with inputs computed by drawing events into a voxel-grid [3], denoted by (V). ANN-6 is only trained with the voxel-grid representation.

Feedforward ANNs cannot continuously predict angular velocity[2]. As a consequence, the training of the ANNs is based on minimizing the error of the mean angular velocity within a time window of 20 ms. Subsequently, the time window is shifted to the next non-overlapping sequence of events. In a similar fashion, ANN predictions are evaluated every 20 ms for comparison with the SNN.

B. Quantitative Evaluation

Figure 4 reports the median of the relative error over the range of angular velocities in the test set for all trained models. All models tend to have high relative error at slow angular velocity. Note, however, that achieving low relative error at low absolute speed is difficult in general due to the fact that the relative error is infinite in the limit of zero angular velocity. Overall, the 6-layer SNN performs comparably to the ANN-6 and the ResNet-50 (A) baseline while ResNet-50 (V) with the voxel-based representation achieves the lowest error in general. These findings are condensed in table II which additionally provides the RMSE and median of relative errors of the naive mean[3] prediction baseline.

Next, we investigate the impact of angular velocities on tilt, pan and roll rates separately. Figure 5 shows the box plots of the relative errors[4] with respect to different angular velocities. Across the whole range of angular velocities, predictions for tilt are slightly more accurate than those for pan while the error for roll is in general higher than compared to the other axes.

C. Discussion

In summary, the SNN is able to regress angular velocity with reasonable accuracy across different rates. The 6-layer ANN achieves only slightly lower error than the SNN. From this result we conclude that it is possible to train SSNs to ANN-competitive accuracy on this continuous-time regression task. The slightly lower performance could originate

[2]It is theoretically possible to shift the time window for very small increments at the expense of computational costs

[3]Arithmetic mean of the training dataset which is close to zero

[4]defined as $\frac{\omega_i - \hat{\omega}_i}{|\hat{\omega}_i|}$, with i for either the tilt, pan or roll axis

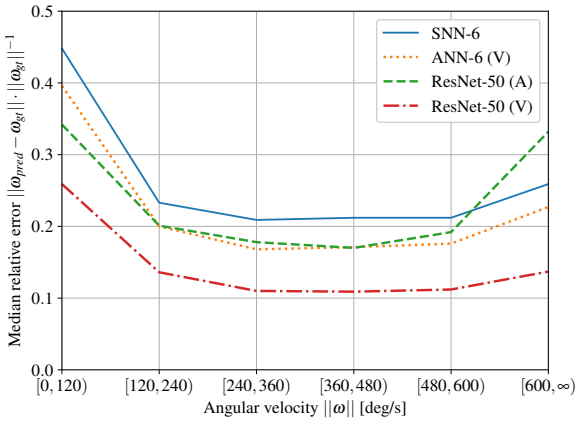


Fig. 4. Median relative errors on the test set for different angular velocities for all trained models. $[\omega_a, \omega_b)$ indicates that angular velocities in the range of ω_a and ω_b are considered. The SNN achieves comparable accuracy to its ANN counterpart with 6 layers. Both are outperformed by ResNet-50 with the voxel-based input representation (V). In contrast, the same network with accumulation-based input (A) achieves errors on the order of ANN-6 and SNN-6. This highlights that the lack of accurate input representation cannot be compensated with increasing the number of layers in the network.

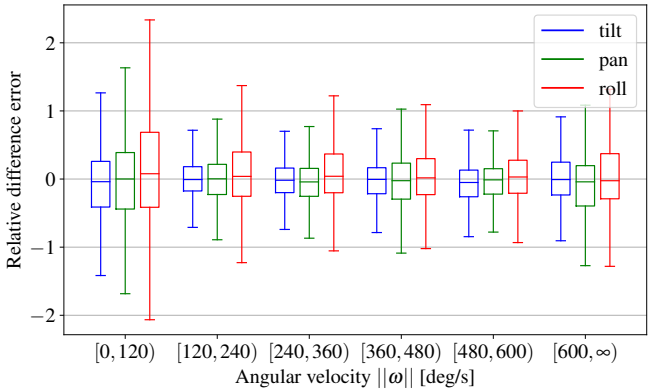


Fig. 5. Quartiles of the relative difference errors of SNN predictions on the test set. The difference between prediction and groundtruth is normalized with respect to the absolute value of the ground truth tilt, pan or roll rates respectively. Evidently, the SNN is performing better at moderate to high angular rates while the roll predictions are in general less accurate than tilt and pan.

from potentially suboptimal hyperparameters for spike and refractory response (τ_s and τ_r in table I). These parameters could potentially be learned as well but this is left to future work.

The large discrepancy between the error achieved by the two ResNet architectures are due to their difference in the input representation. Unlike the voxel-based representation, the accumulation-based representation completely discards timings of the events. This appears to be problematic for regression of angular velocity. On the other hand, the significant jump in accuracy from ANN-6 to ResNet-50, both with voxel-based input, suggests that the SNN could also benefit from increasing the number of layers. Nevertheless, we expect that optimizing deeper SNNs might uncover new challenges for currently popular training methods [12], [18].

Our axis-isolating experiments suggest that predicting roll rate is more challenging for the SNN than predicting tilt and pan rates. Similar observations were made for an optimization-based approach to angular rate tracking [51]. When the camera is being rolled, events are typically triggered at the periphery. The resulting spatial-temporal patterns are spread over the whole frame, which poses difficulties for our architecture.

TABLE II

BASELINE COMPARISONS ON THE TEST SET: THE SNN IS COMPARED AGAINST THE ANN MODELS AND THE NAIVE MEAN PREDICTION BASELINE. INPUT REPRESENTATIONS ARE EITHER EVENT-BASED (E), ACCUMULATION-BASED (A) [50] OR VOXEL-BASED (V) [3].

	mean	SNN-6	ANN-6	ResNet-50	
Relative error	1.00	0.26	0.22	0.22	0.15
RMSE (deg/s)	226.9	66.3	59.0	66.8	36.8
Input type	-	E	V	A	V

V. CONCLUSION

In this work, we investigated the applicability of feed-forward SNNs to regress angular velocities in continuous-time. We showed that it is possible to train a spiking neural network to perform this task on par with artificial neural networks. Thus, we can confirm that state-of-the-art SNN training procedures accurately address the temporal error assignment problem for SNNs of the size as presented in this work. Experimental results further suggest that deeper SNNs might perform significantly better, but there are a number of obstacles ahead. Backpropagation-based approaches require that we unroll the SNN in time at high-resolution. This requirement poses serious challenges for optimization on GPUs both in terms of memory consumption and FLOPS. It has been a long-standing research goal to address these issues and we believe it to be crucial to unlock the full potential of SNNs.

REFERENCES

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

[2] G. K. Cohen, G. Orchard, S.-H. Leng, J. Tapson, R. B. Benosman, and A. Van Schaik, "Skimming digits: neuromorphic classification of spike-encoded images," *Frontiers in neuroscience*, vol. 10, p. 184, 2016.

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

[4] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 106–122, 2018.

[5] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *arXiv e-prints*, vol. abs/1904.08405v2, 2019.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[7] F. Ponulak, "ReSuMe-new supervised learning method for spiking neural networks," *Institute of Control and Information Engineering, Poznan University of Technology.*, 2005.

[8] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "Span: Spike pattern association neuron for learning spatio-temporal spike patterns," *International Journal of Neural Systems*, vol. 22, no. 04, p. 1250012, 2012. PMID: 22830962.

[9] R. Gütig and H. Sompolinsky, "The tempotron: a neuron that learns spike timing–based decisions," *Nature neuroscience*, vol. 9, no. 3, pp. 420–428, 2006.

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

[11] Y. Jin, W. Zhang, and P. Li, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 7005–7015, Curran Associates, Inc., 2018.

[12] 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," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[13] F. Zenke and S. Ganguli, "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks," *Neural Computation*, vol. 30, pp. 1514–1541, Apr. 2018.

[14] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1, pp. 17–37, 2002.

[15] O. Booij and H. tat Nguyen, "A gradient descent rule for spiking neurons emitting multiple spikes," *Information Processing Letters*, vol. 95, no. 6, pp. 552 – 558, 2005. Applications of Spiking Neural Networks.

[16] S. B. Shrestha and Q. Song, "Robust learning in SpikeProp," *Neural Networks*, vol. 86, pp. 54 – 68, 2017.

[17] S. B. Shrestha and Q. Song, "Robustness to training disturbances in SpikeProp learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–14, July 2017.

[18] S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 1419–1428, Curran Associates, Inc., 2018.

[19] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in Neuroscience*, vol. 12, p. 331, 2018.

[20] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, vol. 111, pp. 47 – 63, 2019.

[21] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the National Academy of Sciences*, vol. 113, no. 41, pp. 11441–11446, 2016.

[22] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2015.

[23] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, p. 682, 2017.

[24] S. B. Shrestha and Q. Song, "Event based weight update for learning infinite spike train," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 333–338, Dec 2016.

[25] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an open event camera simulator," in *Conf. on Robotics Learning (CoRL)*, 2018.

[26] W. Maass, "Lower bounds for the computational power of networks of spiking neurons," *Neural Computation*, vol. 8, pp. 1–40, Jan. 1996.

[27] W. Maass, "Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons," in *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996* (M. Mozer, M. I. Jordan, and T. Petsche, eds.), pp. 211–217, MIT Press, 1996.

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

[29] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[30] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proceedings of the IEEE*, vol. 107, pp. 144–164, Jan 2019.

[31] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[32] P. Lichtsteiner and T. Delbruck, "A 64x64 AER logarithmic temporal derivative silicon retina," in *Research in Microelectronics and Electronics, PhD*, vol. 2, pp. 202–205, 2005.

[33] V. Chan, S.-C. Liu, and A. van Schaik, "Aer ear: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, 2007.

[34] B. Schrauwen and J. Van Campenhout, "Improving spikeprop enhancements to an error-backpropagation rule for spiking neural networks," in *Proceedings of the 15th prorisc workshop*, vol. 11, 2004.

[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[36] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[37] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, p. 437, 2015.

[38] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha, "A low power, fully event-based gesture recognition system," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[39] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[40] H. Paugam-Moisy and S. M. Bohte, *Handbook of Natural Computing*, vol. 1, ch. Computing with Spiking Neuron Networks, pp. 335–376. Springer Berlin Heidelberg, 1st ed., 2011.

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

[42] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8977–8986, 2019.

[43] M. Lin, Q. Chen, and S. Yan, "Network in network," in *ICLR*, 2014.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[46] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240$\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[47] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, "Recognizing scene viewpoint using panoramic place representation," in *cvpr*, pp. 2695–2702, IEEE, 2012.

[48] M. v. Rossum, "A novel spike distance," *Neural computation*, vol. 13, no. 4, pp. 751–763, 2001.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2014.

[50] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 5419–5427, 2018.

[51] G. Gallego and D. Scaramuzza, "Accurate angular velocity estimation with an event camera," *IEEE Robot. Autom. Lett.*, vol. 2, pp. 632–639, 2017.