N-MNIST object recognition with Spiking Neural Networks

Federico Pennino
Universität Bielefeld
federico.pennino@uni-bielefeld.de

Shamini Koravuna
Universität Bielefeld
skoravuna@techfak.uni-bielefeld.de

Christoph Ostrau
Universität Bielefeld
costrau@techfak.uni-bielefeld.de

Prof. Dr.-Ing. Ulrich Rückert
Universität Bielefeld
rueckert@techfak.uni-bielefeld.de

***Abstract*—In the modern world, energy efficiency is crucial, that is the motivation why optimization of energy use is so important. Event-based signaling and processing promises increased energy efficiency (human brain as paragon). Spiking Neural Networks (SNNs) are artificial neural networks that more closely mimic natural neural networks. Their use has shown interesting results in the field of energy efficiency. In this work, we explore an SNN approach for object recognition based on larger event frames from `N-MNIST`. We get a final accuracy of 99% on the object detection task and an accuracy of 85% on the classification task.**

I. Introduction

Artificial Neural Networks (ANNs) are computing systems inspired by the biological neural networks that constitute animal brains. Over the years they have been heavily studied but what is important for our use is their evolution, so called *Spiking Neural Networks* (SNNs) [3]. SNNs are artificial neural networks that more closely mimic natural neural networks. In addition to neuronal and synaptic state, SNNs incorporate the concept of time into their operating model. In this project, we use them combined with `N-MNIST` dataset [4], a version of the classic MNIST obtained via neuromorphic camera [1]. We do this because biological neural systems can well perform tasks with small energy consumption. At the end of our work, we present an *SNN* architecture - inspired by *YOLO* [6] - able to detect and classify digits in larger event-based frames. `Norse` [5] has been as simulator for *SNNs*.

II. Dataset

We extend the `N-MNIST` dataset putting the digits in a larger frame. A lazy approach has been chosen to do that. This means that meta-data of the neumorphic sequence is generated when the dataset is created, but the real sequence is rendered only when it's needed. We set a shape of 128×128 as default size, where the original shape was 34×34. The original `N-MNIST` data is not resized, it's only padded. However, it's even possible to reshape randomly the original sequence of frames, new shape will range in $[34;\ max]$.

III. Network

We tried to maintain the structure of the network as simple as possible. Our network architecture is inspired from *YOLO*, that is where the idea of the sub-networks came from. However, looking at it in detail, we can see a shared part between *classification* and *detection*. This common stream is simply composed of two convolutional layers. The first convolutional layer is composed of 30 filters of shape 5×5. The second one is composed of 70 filters of shape 5×5. Result of each convolution is max pooled (shape 3×3) and `LIFCell` is applied. So far we have a *flatten* layer and the network is splitted in two *streams*: one for *classification* and one for *detection*. These two parts differ in shape but not in structure. In both we have a first *linear* layer followed by a `LIFRecurrentCell` and as output we have a *linear* layer where `LICell` is applied. The network is iteratively called on each input frame and the result is stacked. At the end, max voltage is taken for each class. In Fig. 1 network structure is shown.

IV. Loss function

Our network is working on two different tasks at same time, then we decided to use an additive *loss function*. The resulting

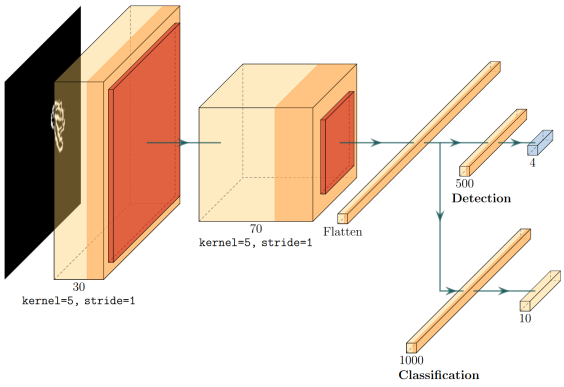

Fig. 1. A network overview. There's a common stream, and after it, the network splits in two sub-streams: one for classification task and the other for detection task.

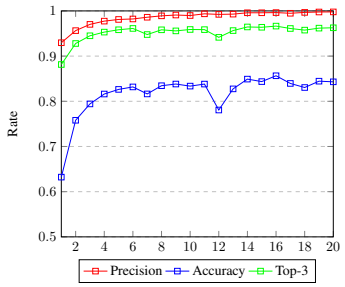

Fig. 2. Results on test and training set. The network has been trained for 20 epochs. The red line is the *bounding-box* prediction accuracy with a $threshold = 0.5$.

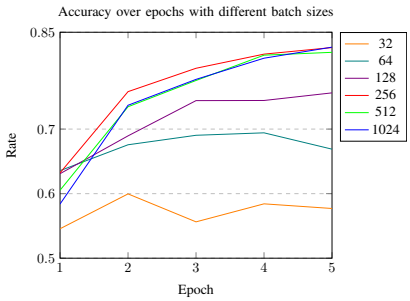

Fig. 4. Classification results based on `batch_size`. We can see how batch size is influencing the training. Bigger batch sizes are the ones with best accuracy values.

loss function, used for training and evaluation, is:

$$\ell_{total} = L_1(\theta_1) + L_2(\theta_2)$$

where L_1 denotes the *Mean Squared Error* and is applied to the detection stream and L_2 denotes the *label smoothing loss* and is applied to the classification stream. *Adam* has been used as an optimizer.

V. Results

The network has been trained over 20 epochs with a $batch\_size$ of 256. It is able to reach a score up to 99% in detection task and it is stable during the training. Furthermore, the results over classification are satisfactory. We obtained an accuracy of 85% in classification task and 99% on detection task. It is even interesting watching the 3-top accuracy score: correct result stands on the top 3 predicted values 95% of the time. This means even under uncertainty conditions network is well generalized. It is worth of attention how it seems that loss value is more related to accuracy than to precision. This can be caused by the fact that when precision start to be high, *classification stream* training start to be dominant. In Fig. 2, results are shown.

We went further to the basic implementation modifying the input: the original digit has been reshaped from the shape of 34×34 to a squared shape in range $[34 - 58]$. So far, we observed that the results are almost comparable with the original version. The best result we get is around 82% on classification, but contrary to not reshaped data - on classification task - network tends to overfit. Another alternative version of our network has been tested, this time we add noise at the end of the classification stream. We basically sum a value sampled from a normal distribution $\mathcal{N}(0,\ 1)$ to each output's value. We did that to improve the quality of our results. At the beginning of the training the noise is visible, the training is definitely slower. However, after some epochs, we converge to same results as original network. We can't see a real upgrade to the network, even though, contrary to what we see at epoch 11 in Fig. 2 there is no accuracy drop during the training. In Fig. 3 we summarize obtained results. In conclusion, we tried to understand which *hyperparameters* are more important for our network. The parameter α in `label_smoothing` has shown a good impact on the performance of the classifier. Setting $\alpha = 0$ can cause a drop of accuracy around a 1-2%. However, what is really impacting is `batch_size`. As we can see in Fig. 4 at the increasing of *batch_size* value correspond an increasing of the classification task accuracy.

	Precision	Accuracy	Top-3 accuracy
Original	99,78%	84,91%	96,45%
Resized input	99,78%	82,32%	95,30%
Random noise	99,78%	85,24%	96,245%
[2]	-	98.74%	-

Fig. 3. Experimental results. We report in this table the results over the various experiments. Furthermore, we add the SNNs state-of-art in `N-MNIST` classification.

VI. Conclusion

We showed how *Spiking Neural Networks* can be used for *object recognition* task. We provided an architecture that is able to predict a *bounding-box* containing a digit from `N-MNIST`, a spike-version of classical *MNIST* dataset. We present an approach that is able to reach an accuracy of 85% on classification task and 99% on detection task. We investigated even the possibilty of modifying the network adding some noise and applying it to different formatted input. Our results can be considered as a good starting point for future implementations of *Spiking Neural Networks* in *object recognition* field.

References

[1] Guang Chen, Hu Cao, Jorg Conradt, Huajin Tang, Florian Rohrbein, and Alois Knoll. Event-based neuromorphic vision for autonomous driving: A paradigm shift for bio-inspired visual sensing and perception. *IEEE Signal Processing Magazine*, 37(4):34–49, 2020.

[2] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10, 2016.

[3] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[4] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[5] Christian Pehle and Jens Egholm Pedersen. Norse - A deep learning library for spiking neural networks, January 2021. Documentation: https://norse.ai/docs/.

[6] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.