Fast event-driven incremental learning of hand symbols

Iulia Alexandra Lungu, Shih-Chii Liu, and Tobi Delbruck
Institute of Neuroinformatics, University of Zurich and ETH Zurich
Zurich, Switzerland
iulialexandra,shih,tobi@ini.uzh.ch

***Abstract*—This paper describes a hand symbol recognition system that can quickly be trained to incrementally learn to recognize new symbols using about 100 times less data and time than by using conventional training. It is driven by frames from a Dynamic Vision Sensor (DVS) event camera. Conventional cameras have very redundant output, especially at high frame rates. Dynamic vision sensors output sparse and asynchronous brightness change events that occur when an object or the camera is moving. Images consisting of a fixed number of events from a DVS drive recognition and incremental learning of new hand symbols in the context of a RoShamBo (rock-paper-scissors) demonstration. Conventional training on the original RoShamBo dataset requires about 12.5h compute time on a desktop GPU using the 2.5 million images in the base dataset. Novel symbols that a user shows for a few tens of seconds to the system can be learned on-the-fly using the iCaRL incremental learning algorithm with 3 minutes of training time on a desktop GPU, while preserving recognition accuracy of previously trained symbols. Our system runs a residual network with 32 layers and maintains 88.4% after 100 epochs or 77% after 5 epochs overall accuracy after 4 incremental training stages. Each stage adds an additional 2 novel symbols to the base 4 symbols. The paper also reports an inexpensive robot hand used for live demonstrations of the base RoShamBo game.**

***Index Terms*—neuromorphic, event camera, machine learning, data-driven, incremental learning, robotics, computer vision**

I. Introduction

Many state-of-the-art results in computer vision are based on convolutional neural networks (CNNs) processing images captured by conventional frame-based cameras. CNNs are trained using backpropagation on large labeled datasets. Learning new classes requires retraining on both old and new data. However, for platforms with limited resources, such an approach is not optimal, since computing and storage are limited. Neuromorphic cameras such as the Dynamic Vision Sensor (DVS) [1] [2] operate akin to the transient pathway in biological eyes to only report brightness changes in the scene, thereby obviating the need to process entire images. In this paper we present a system based on [3], [4] which uses an accumulation of a fixed number of DVS events to create sparse images of hand symbols for classification. Moreover, we also demonstrate how a recent incremental learning algorithm called iCaRL [5] is used to add new symbols that the system can recognize, without forgetting the old symbols, while using only 1% of the time needed to train the old symbols.

This work is partially funded by the Samsung Institute of Technology.

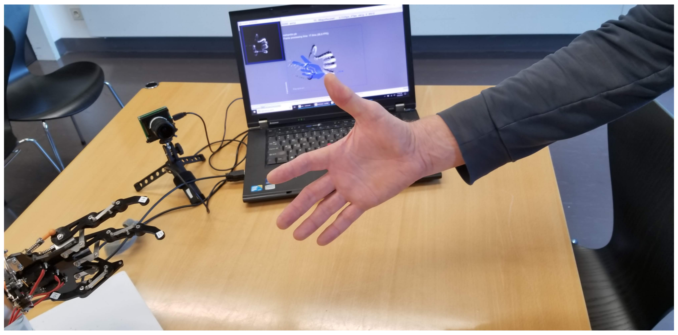
Fig. 1. "Dextra" robot hand driven by a convolutional network that can learn hand symbols on the fly.

In working towards our aim for continuous learning on low-power embedded hardware, this work reduces the resources required for training the system to recognize new symbols. This paper introduces a novel event-based hand symbol dataset and shows that event-based cameras can be successfully integrated in a CNN-driven pipeline for incremental object classification. Section II presents the related literature, while Section III introduces the methods used for incremental learning, described in detail in Section IV. The paper ends with a section discussing implications and future directions.

II. Related work

The original rock-paper-scissors demonstration was briefly described in [4]. This paper reports an improvement of it by the inclusion of a robot hand described in Sec. IV-A. The demonstration plays the game of RoShamBo against a human opponent. Rather than trying to outguess the opponent, it turns RoShamBo into a physical sport by quickly recognizing the human's symbol and showing the winning symbol in response. The symbol recognition is performed by a small CNN that is driven by frames of accumulated DVS events. The frames have a constant number of brightness change events. Slow hand movements generate the frames at a low rate of about 2Hz, while rapid hand movements generate them at a higher rate of up to 500Hz. By using constant-event frames, the DVS frame is not blurred even for the fastest hand movements, as it would be for a constant-time frame. The small CNN capable of learning the three hand symbols plus the background no-

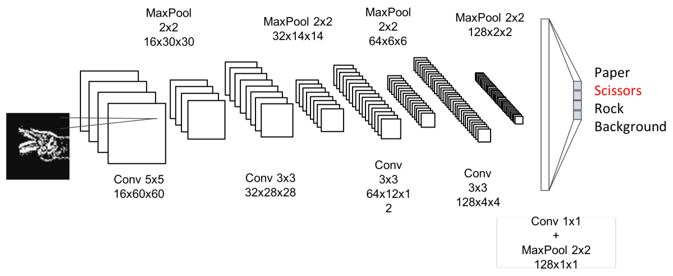

Fig. 2. The convolutional network used for the original rock-paper-scissors demonstration [4].

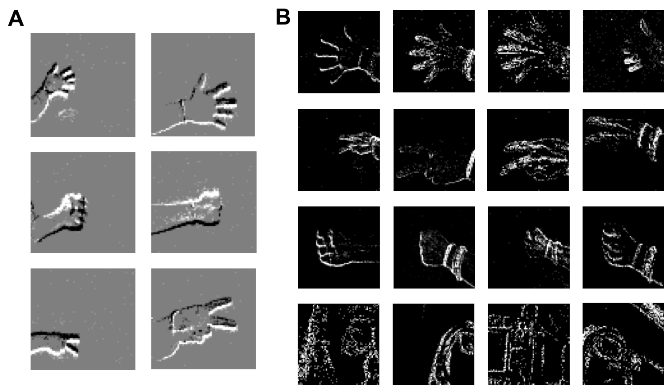

Fig. 3. A: DVS samples before normalization. B: DVS samples after normalization, including background class samples

symbol class and achieving real-time (<20ms on laptop CPU) inference is shown in Fig. 2 (see also Sec. IV-A).

In incremental learning, catastrophic forgetting occurs when training an existing network only with new data overwrites the old knowledge [6].

Forgetting can be offset to a certain extent by various methods, such as exemplar storage (iCaRL) and knowledge distillation [5], [7], elastic weight consolidation (EWC) [8], and autoencoder or GAN sample generation [9], [10], although the topic is still actively researched. EWC has been shown to not scale well [10], [11], while [9], [10] generate training samples from the model, which is much slower than reading them from memory.

III. Methods

For incremental symbol learning in this work, we chose iCaRL [5] because it has bounded memory, and automates the choice of exemplars. These features make it a good match to the data-driven frames from DVS and our live demonstration.

Symbols are classified by a residual network with 32 layers (ResNet-32) [12], because the capacity of the original 114k-parameter 18 MOp/frame RoShamBo CNN of [4] proved to be too small for incremental learning. The ResNet-32 has 460k parameters and takes 190 MOp/frame to compute. For live demonstration, the ResNet-32 requires about 130ms per frame in TensorFlow 1.10.0 in CPU mode on a linux PC compared with about 13ms for the original CNN.

The asynchronous brightness change events are accumulated into 64×64 2D pixel histograms of a constant number of events, referred to here as *constant-event* frames [3]. Since the frames are created using a fixed number of events rather than a fixed time window, the frame rate is proportional to the hand speed. The output decision consists of the class label of the symbol being shown by the human participant.

The base training data for this study uses the ROSHAMBO17 dataset[1] that consists of labeled continuous DAVIS240C recordings of people showing a single symbol, obtained using jAER [13], a software developed to process DAVIS data. The recordings are cut into frames using the jAER utility `dvs-slice-avi-writer` and compiled into Google TensorFlow tfrecords. Input normalization (Fig. 3) included the rectification of all DVS events to positive ON events with a 200-event maximum grayscale bin value and mapping the image pixel values to a 0-1 range by performing a 3-sigma normalization [3]. These methods are applied both during training and at inference time.

Training is divided in two phases: first we perform a *base training* over 100 epochs on the 2.56 million ROSHAMBO17 samples used for the original RoShamBo demonstration. ROSHAMBO17 consists of the three symbols used in the game, *rock*, *paper* and *scissors*, as well as a fourth *background* class which accounts for all other object types. Users showed each symbol with each hand for about a minute and were instructed to explore all possible orientations, positions and scales. The background class data was collected from camera output during no motion (noise) and by showing the camera as many other scenes as possible, e.g. body movement, waving the camera around the office, etc. Fig. 3 shows examples for the input classes before and after normalization. All the recordings for the base training were sampled using four different numbers of accumulated events: 500, 1000, 2000 and 4000. The resulting images were also mirrored to augment the data. This training phase was performed offline, using all the available samples, and it represents the core knowledge of the system. Training the ResNet-32 over 100 epochs on the 2.56 million samples of the base classes requires 12.5 hours on a 250W Nvidia GTX 1080 Ti GPU.

However, the goal is to use this algorithm on mobile devices to acquire new knowledge in the real world, which imposes constraints on training time and memory. Therefore, we wanted to avoid retraining on all previous data when new objects must be learned. iCaRL selects some of the most representative samples for each of the classes encountered and only stores those in memory. Here, we stored only 4000 exemplars for each class compared with the original 640k samples per base class (160 times fewer samples). These exemplars are chosen automatically, to closely approach the mean features of their respective classes. Each exemplar is chosen by maximizing the normalized dot product between the average feature vector over all the already chosen exemplars

[1]ROSHAMBO17: http://sensors.ini.uzh.ch/databases.html

and the average feature vector of all the samples. The feature vector consists of the output of the penultimate layer of the ResNet. In iCaRL, the symbol classification is based on this same metric, by comparing the feature vector relative angle between the input image feature vector and the mean feature vector of each set of class examplars, which stands in contrast to the usual softmax output used for conventional classifier CNNs.

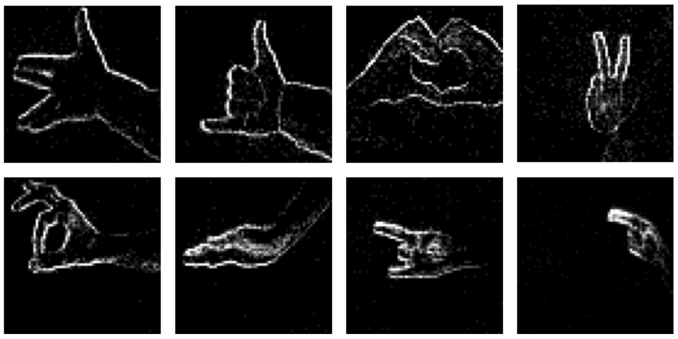
Fig. 4. Incremental symbols used in this paper

Whenever data for novel classes becomes available, a new re-training stage of the network takes place using the stored exemplars and the new data. After each incremental training stage, exemplars for the newly learned classes are stored. A distillation loss [7] is used to better preserve knowledge about the known classes. By using a distillation procedure, the network is encouraged to output the same non-softmaxed scores (logits) for the old classes as it did in the previous training phases. This loss also ensures that backpropagation is applied to all output units for the old classes, since these scores all have non-zero values, unlike the one-hot labels for the new classes. Novel class data (Fig. 4) is acquired in the same manner as the base data, i.e., by accumulating events and cutting videos into frames. However, for incremental learning of new symbols, we used only 2000-event DVS frames. Furthermore, the videos recorded for each new class are much shorter than the base training data, ranging from 15 seconds to 1 minute. To compensate for a lack of data, we augmented the new samples with additive uniform noise with an amplitude of 2 gray level units (total grayscale was 0-255) as many times as necessary to obtain 4000 images for each novel class.

The incremental symbol learning demonstration is driven by a custom jAER class, RoShamBoIncremental[2] that communicates over a UDP socket to a python process. RoShamBoIncremental provides a GUI used to inform the system that a new symbol is being presented. Raw data is streamed to a file. When the demonstrator gauges that a sufficient variety of data has been collected, they push a button that provides a dialog to enter the new class label. Next, RoShamBoIncremental sends a message to the python client to start incremental training on the new data. The python client produces the training samples from the recorded data and starts training. When progress is complete, RoShamBoIncremental loads the new CNN from disk.

[2]https://github.com/SensorsINI/jaer/blob/master/src/ch/unizh/ini/jaer/projects/npp/RoShamBoIncremental.java

IV. Results

To quantify incremental training capability, we recorded a new dataset of 8 novel symbols, examples of which are shown in Fig. 4. Table I shows recognition accuracy results for base and incremental training after incremental learning of the first two novel symbols. Accuracies are calculated on test set data from samples not used in training, selected by recording a separate video for each novel symbol. The iCaRL accuracy is compared to a regular network, which does not retrain on previous classes whenever new symbols are available. Both the regular and iCaRL networks can be trained over 100 epochs for 97% accuracy on the 4 base symbols (“BA after BT”). However, if the regular network is retrained (starting from the base weights) on only the samples for two novel symbol classes, it achieves 99.56% accuracy on these two novel symbols (“IA after IT”), but the accuracy on the original base classes drops to almost 0% (“BA after IT”). This extreme illustration of catastrophic forgetting shows that the network never selects the base classes, because it mistakes all samples as belonging to the new classes. Even after only a few epochs of training the base class accuracy drops to less than 10%. However, using the iCaRL method, which uses exemplars of all classes for the incremental training phase, the network maintains a high accuracy of 92.9% on the base classes while achieving 95.6% accuracy on the two new classes. All the Table I experiments used 100 epochs of training.

TABLE I
Regular versus incremental training

Network type	BA after BT	BA after IT	IA after IT
Regular	97%	0.00002%	99.56%
iCaRL	97%	92.9%	95.6%

BA = accuracy on base classes
BT = training on base classes
IA = accuracy on incremental classes
IT = training incrementally on new classes

To reduce training time during the live demonstration, the incremental training of each two new symbols only used 5 epochs, which takes about 3 minutes on the NVIDIA GTX 1080 Ti GPU, as opposed to 15 minutes for 100 epochs. Fig. 5 shows the impact of reducing the number of epochs on accuracy. The x axis shows how the overall accuracy (on old and new classes combined) evolves as more classes are added incrementally. The curves are an average of 30 runs with each run using different choices of the novel classes. The left-most accuracy value corresponds to the 4 base classes alone. Its standard deviation is zero, because the same base network was used for all the 30 runs. At each incremental training stage we add 2 novel classes. In total we learn 8 new symbols on top of the 4 base ones. Adding the first 2 new

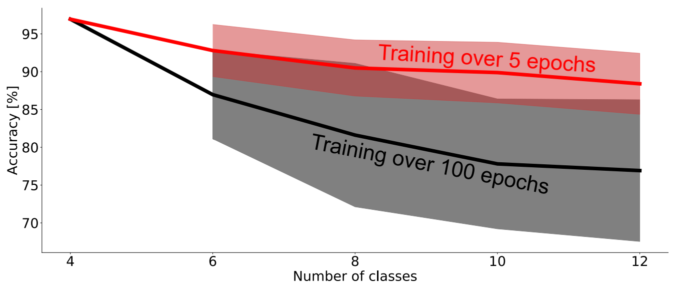

Fig. 5. Overall recognition accuracy of the incremental learning algorithm for different numbers of iCaRL training epochs. The solid curve represents the mean accuracy over 30 runs. The shading corresponds to the 1-sigma bounds. The novel symbols added are shuffled at each run.

symbols requires only 5 epochs training to reach above 85% accuracy. Although the drop in accuracy compared to training the network over 100 epochs is significant (around 8 percent points), when tested in a live demonstration, the level of accuracy obtained after only 5 epochs of training is sufficient to convince participants that the system can recognize their hand gestures. Accuracy variability results from a combination of random initial weights for the newly added output units, as well as the random choices of novel symbols; some are more difficult to distinguish than others. Only about half of the variability in accuracy is due to incomplete training. A video[3] shows the live demonstration.

A. Robot hand

Fig. 1 shows a robot hand that we added to the base RoShambo game as part of this work. This "Dextra" hand proved popular with the public, so we report it for easier replication. The robot hand shown is a no-brand device available from ebay and aliexpress, sometimes labeled as "Bionic Robot Hand". It costs about $100 unassembled. We replaced the low quality servos with better analog servos costing about $20 each. To control the fingers, we built a controller using an Arduino Nano. It receives single character commands over the USB serial port to show each symbol. Because the fingers tend to jam, to prevent burnout, the Arduino firmware ensures that servos are only powered on for less than 500 ms. The analog servos turn off motor current when pulses are not sent to them and do not generate the annoying buzzing sounds from more modern digital servos. The 3 RoShamBo symbols use only two servo channels. The robot hand symbol changes in about 150 ms (a human can change symbols in about 75 ms). Videos[4] show Dextra playing RoShamBo.

V. Conclusion

This paper introduced an event-driven hand symbol recognition system which can be incrementally trained to recognize new symbols without forgetting old symbols in a few minutes, a factor of more than 100X faster than retraining the network from scratch on all the data. The training set memory size is also reduced by a factor of 160X. It means that a smartphone GPU with about 1% of the speed of a desktop GPU could compute the incremental training using about 64MB of training-set memory in about 5h training time. This is still a significant amount of time but could take place during battery charging. Despite iCaRL contributions to incremental learning, this algorithm still requires thousands of samples to learn new classes. To further reduce training time, we are looking into Siamese networks for one shot learning [15], that are able to recognize new symbols after having seen only a few examples for each novel class.

[3]Incremental RoShamBo: https://youtu.be/aWK572MMa1E

[4]Dextra driven by NullHop [14]: https://youtu.be/nTUjROa5f18
Dextra on laptop CPU with naive subject: https://youtu.be/95GsOQbwNLU

References

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb 2008.

[2] R. Berner, C. Brandli, M. Yang, S.-C. Liu, and T. Delbruck, "A 240× 180 10mW 12us latency sparse-output vision sensor for mobile applications," in *2013 Symposium on VLSI Circuits (VLSIC)*, 2013, pp. C186–C187.

[3] D. P. Moeys, F. Corradi, E. Kerr, P. Vance, G. Das, D. Neil, D. Kerr, and T. Delbrck, "Steering a predator robot using a mixed frame/event-driven convolutional neural network," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, June 2016, pp. 1–8.

[4] I. Lungu, F. Corradi, and T. Delbruck, "Live demonstration: Convolutional neural network driven by Dynamic Vision Sensor playing RoShamBo," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–1.

[5] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[6] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks : The sequential learning problem," in *Psychology of Learning and Motivation*, G. H. Bower, Ed. Academic Press, 1989, vol. 24, pp. 109–165. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0079742108605368

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," Mar 2015. [Online]. Available: https://arxiv.org/abs/1503.02531

[8] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: http://www.pnas.org/content/114/13/3521

[9] R. Kemker and C. Kanan, "FearNet: Brain-inspired model for incremental learning." [Online]. Available: http://arxiv.org/abs/1711.10563

[10] A. Rios and L. Itti, "Closed-loop GAN for continual learning." [Online]. Available: http://arxiv.org/abs/1811.01146

[11] R. Kemker, A. Abitino, M. McClure, and C. Kanan, "Measuring catastrophic forgetting in neural networks." [Online]. Available: http://arxiv.org/abs/1708.02072

[12] 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*, 2016, pp. 770–778.

[13] "jAER." [Online]. Available: http://jaerproject.org

[14] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu, and T. Delbruck, "NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2018.

[15] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proceedings of the 32nd International Conference on Machine Learning, Lille, France*, vol. 2, 2015.