Neural Networks at the Edge

ABC
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

I. Introduction

Problem: Federated setting, different parties exist and different resources available with constraints on these resources. How to split a neural network to address these resource constraints?

Challenge in a federated setting and edge is that the edge has limited resources. For example, party A may provide access to party B for only limited amount of resources to ensure that its clients' needs are met. Neural networks have become quite prevelant, however are resource intensive. One way to improve the resource consumption is to reduce the footprint of the neural network (e.g., TFLite). This does not always solve the problem as depending on the resource availability, the network may not get executed at all.

We examine the problem of scoring neural networks in a federated setting with resource constraints. Idea: Splitting of the neural network layers across multiple nodes. Evaluate the initial part in party A and then use the output with confidence metric to determine partial information on the task at hand. This is based on the resource consumption for the portion of the task. Then, depending on the need and availability of resources, move to the next portion of the computation on a different node (possibly in one's own party, when it becomes available). More details of this will be in scenario section (TODO: Richard).

The splitting and conditional evaluation is explained in Splitting section (TODO: Kaushik)

Can this be extended to using Spiking (TODO: Kaushik)

Conclusions

II. Scenario and Problem Statement

Our research is motivated by the need to enable situational awareness and situational understanding in coalition military operations [1], [2]. Militaries are increasingly using technology to help achieve this goal, deploying a variety of intelligent sensors for collecting and analyzing data. This has led to the development of the Internet of Battlefield Things (IoBT) concept to describe future military operations. In the IoBT, the things (e.g. sensors, vehicles, robots, wearable devices) will be augmented with various levels of intelligence, and will communicate with each other to coordinate their operations, as well as with their human users to facilitate sensemaking [3], [4].

The battlefield setting presents a number of barriers to achieving this vision. Information flow constraints are a major challenge: devices will be distributed and move around over a wide geographical area, often in regions with poor or no communications infrastructure. Communications will need to be wireless — for example via satellite link or cellular networks — but these methods suffer from high latency, as well as being potentially costly. The current popular model of providing devices with intelligent behaviours by transferring their data to back-end cloud services for processing will not be possible. Devices will therefore require local data processing and intelligence capabilities [5].

However, edge devices are also locally resource-constrained in terms of compute capability and energy consumption. A particular device may be serving many different requests from different users, and with limited local compute capacity could quickly become overloaded. If battery powered, the device will also need to minimize its energy consumption by reducing the number of computations it performs, while still remaining useful to the coalition.

In this paper, we explore a method that addresses these issues in the case of edge devices running deep neural networks (DNNs). DNNs have proven extremely effective for a variety of machine learning tasks including data classification, time-series prediction, anomaly detection, and speech/language processing [6], [7]. The power of DNNs comes from their capacity to learn rich, hierarchical representations from data [8]. The term "deep" refers to the number of layers used to learn these representations. For complex data such as video, audio, or images, deeper models with more layers tend to perform better than shallower models [9]. However, as the depth of the model increases, so do the computational requirements, meaning that DNNs may not be practical for deployment on computationally-limited or energy-constrained edge devices.

We propose exploiting the hierarchical nature of the learned representations in a DNN to adaptively distribute the computation over the edge devices in the coalition network. Our method builds on work by Panda et al., which showed how the energy consumption of DNN computation could be reduced by considering the output of one network layer at a time, and only computing the output of the next layer if the confidence of the output based on the current layer was low [10], [11]. This minimizes computational requirements by cutting out unnecessary processing by deeper layers for "easier" inputs. Our approach proposes distributing the computation of different layers between different devices on the network. Using this scheme, the processing requirements on the first device are minimized, and if further processing by deeper layers is required, another device (or, if reachable, the cloud) can be called on to do this if the first device needs to reduce its computational load. This process can be chained, potentially involving as many devices as there are network layers.

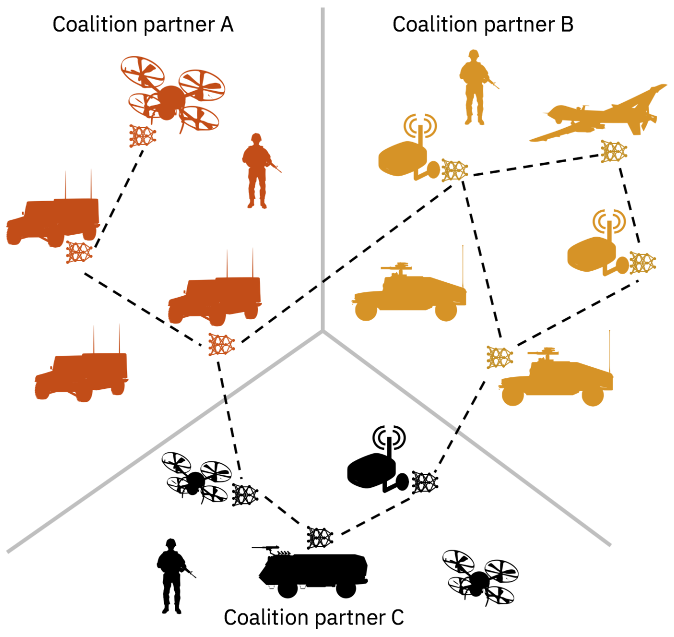

Fig. 1. Conceptual illustration of the IoBT, and the proposed distribution method for DNNs. In this example, three coalition partners are collaborating. Each partner has deployed intelligent devices to the battlefield. Several of these devices have the same DNN deployed on them, and so can distribute computation between them layer by layer as described in section III.

III. Conditional Deep Neural Networks

Conventional deep neural networks (DNNs) have scaled greatly in depth, from 8 layers in AlexNet [12] to 1000 layers in ResNet-1001 [13]. With the increase in size, the cost of implementing these very deep networks have increased as well. In order to determine the classification result, the input is processed through all the layers of the DNN, thereby consuming a lot of time and effort. However, in real world, not all inputs are made equal, and one can ideally design algorithms whose complexity is proportional to the difficulty of the input [14]. Such a design can provide both energy efficiency and speed up. Conditional Deep Learning [11], [10] is a technique of augmenting deep neural networks with linear classifiers at the output of intermediate layers, that gives the user the control to conditionally activate the deeper layers of the network.

In a Conditional Deep Learning network (CDLN), there are conditional exits added to the network after every few layers. As shown in Fig. 2, when an input is presented to the network, the first linear classifier outputs the classification probability of the image. If the classifier has a strong confidence, that is the maximum output is beyond a threshold, then the network assigns the corresponding label to the input. If the confidence value is lower than threshold, then the next layers are activated and we move to the next classifier. These linear classifiers can be trained in tandem while training the whole network, or can be trained separately after the network is fully trained.

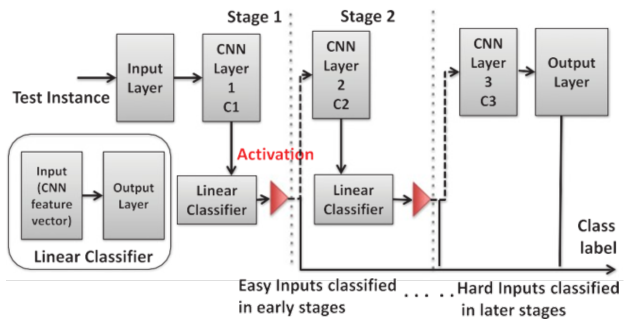

Fig. 2. Conditional Deep Learning Network (CDLN) with linear classifiers added at the convolutional layers whose output is monitored to decide if classification can be terminated at current stage or not.

CDLN [11] offers $1.84\times$ improvement in energy in comparison to traditional implementation for MNIST on LeNet-5 [15], which increases to $4.96\times$ for TinyImagenet implemented on ResNet-50 [16]. Along with energy improvements, CDLN also offers significant improvement in inference time. As the whole network doesn't need to be activated for all the examples, average inference time per example reduces. It is reported in [11], that CDLN implementation speed up the inference by $4.24\times$ for MNIST and $1.71\times$ for TinyImagenet.

For a resource-constrained environment, CDLN offers an unique solution, that has both energy and timing improvements. In [17], conditional deep learning is used to implement staged inference for fast real time applications. For applications involving distributed intelligence, CDLN offers a framework that can be split up and deployed across edge and cloud. A low power system with low compute capabilities can be used to execute the the initial layers of the network. The linear classifier at the end of the first stage can provide us with some information regarding the input with a certain degree of confidence. In situations where the confidence of the edge is low, the data can be transmitted to the cloud to be processed by the remaining layers of the network.

IV. Spiking extensions

Can spiking help reduce further energy consumption? Will the splitting/conditional evluation extend to spiking and what challenges are faced?

The state-of-the-art DNNs are compute-intensive since they operate on real-valued inputs, leading to higher energy consumption in resource-constrained edge devices with limited power budget. In this regard, Spiking Neural Networks (SNNs) offer a promising solution for enabling energy-efficient neuromorphic computing in the edge nodes. SNNs process and encode input information temporally using sparse spiking events. The intrinsic sparse event-driven computing capability of SNNs can be exploited to achieve higher energy efficiency in hardware implementations as shown in [18], [19]. The inherent computational efficiency of SNNs can be attributed to the fact that every layer of an SNN needs to compute the weighted sum of the input spikes with the synaptic weights only in the event of a spike fired by the corresponding input neurons. Furthermore, computing the weighted input sum requires only adders since the inputs are encoded as spikes that are represented by logic states $\{0, 1\}$. On the contrary, DNNs require multipliers, which consume at least an order of magnitude higher energy than adders, for computing the weighted input sum since they operate on real-valued inputs.

SNNs are naturally suited for splitting and conditional evaluation among different edge nodes because of the following two-fold reasons. First, they improve the energy efficiency per edge node due to inherent sparse event-driven processing capability as mentioned in the previous paragraph. Furthermore, researchers in [18] demonstrated that sparsity in spiking activity increases substantially across successive layers of a deep SNN. Hence, energy efficiency would be further enhanced for edge nodes evaluating the deeper layers of an SNN. Second, SNNs minimize the communication overhead since only the sparse spiking events need to be transmitted between the edge nodes. Despite the computational and communication energy benefits offered by SNNs, the challenges relating to the training complexity of deep SNNs need to be addressed to champion their wide-spread adoption for real-world applications. Needless to say that efficient training strategies for deep SNNs is a current area of active research interest.

The training methodologies for deep SNNs can be broadly divided into the following three categories:

1) DNN-to-SNN conversion
2) Spiking backpropagation
3) Spike Timing Dependent Plasticity (STDP)

In the DNN-to-SNN conversion approaches, a DNN is trained using artificial rate-based neurons with state-of-the-art backpropagation algorithms and then mapped to deep SNN by substituting the artificial neurons with appropriate spiking neuron models and suitably normalizing the trained weights [20], [21], [22], [23], [18]. The conversion approaches have been shown to be almost loss-less for deep VGG and ResNet architectures for complex vision datasets. However, the conversion approaches typically incur larger inference latency for achieving the best accuracy. The inference latency can be minimized by using the proposed splitting and conditional evaluation approach. Alternatively, the inference latency can also be optimized by training deep SNNs using spike-based backpropagation algorithms that use differentiable approximations for the spiking neurons for error backpropagation [24], [25], [26], [27], [28], [29], [30]. However, the SNN backpropagation algorithms incur longer training time compared to the DNN-to-SNN conversion approaches while offering the potential to lower the inference latency under iso-accuracy conditions. The final approach uses bio-plausible layer-wise STDP-based local learning rules to self-learn hierarchical input representations in an unsupervised manner [31], [32], [33], [34], [35], [36], [37], [38], [39]. The STDP-based training rules are appealing for edge devices since they can be implemented with minimal hardware overhead compared to backpropagation algorithms. However, the STDP-trained SNNs proposed until now are only few (two to three) layers deep. It is not yet clear if the STDP-based learning rules are effective for the later layers of deeper SNNs.

Finally, we note that binary SNNs, which use binary weights $\{-1, 1\}$ and spiking activations $\{0, 1\}$, can be used to achieve both memory- and energy-efficiency in the edge nodes. Binary DNNs, on the contrary, use either $\{-1, 1\}$ [40] or $\{-\alpha, \alpha\}$ [41] where α is a layer-specific scaling factor for the weights and binary neuronal activations $\{-1, 1\}$. Hence, every layer of binary DNNs still needs to compute the weighted input sum (XNOR operation followed by population count) for all the input neurons and transmit the binary activations of all the output neurons. However, binary SNNs need to compute the weighted input sum (AND operation followed by population count) only in the event of a spike fired by the corresponding input neurons and transmit only the sparse spiking activations of the output neurons, leading to potentially much higher energy efficiency. Binary SNNs can be trained off-line using binarization algorithms proposed for DNNs that require the full-precision weights to be stored during training [40], [41], [42]. Alternatively, binary SNNs can also be trained on-chip using stochastic-STDP based learning rules that achieves plasticity by probabilistically switching the binary weights based on spike-timing [43], [44], [45], [46]. Stochastic-STDP trained binary SNNs, which eliminate the need for storing the full-precision weights, are attractive for memory- and energy-efficient learning as well as inference in the edge nodes.

V. Related Work

Prior works closely related to CDLNs and SNNs have been extensively referenced in the above sections. Previous works on addressing the broader resource challenges for DNNs fall into three main categories: (a) compressing the DNN models to reduce their footprint, (b) branching and early-exit to cut down the cost of inference, and (c) algorithmic optimizations in the compute-intensive parts of the training and inference steps. In the following, we constrast the proposed ideas of this paper with these works.

Model compression Network compression schemes aim to reduce the the total number of model parameters of a deep network and thus reduce the resources required to perform inference. A pruning approach to remove network connections having small contributions is one of the common techniques [47]. SqueezeNet achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters and models smaller than 0.5MB using downsampling techniques [48]. Whereas compression has benefits in terms of resource consumption, thisa paper goes further in distributing network layers across devices with CDLN-based early exits on top of SNN-based architectures.

Early exit With the purpose of regularizing the main network, Szegedy et al. [49] introduced the concept of adding softmax branches in the middle layers of their inception module. BranchyNet [50] extends CDLN by jointly training early exit branch networks. Recently, thin vertical sub-networks, as opposed to shallow sub-networks as achieved with CDLN, are proposed to enable "anytime" prediction according to resource availability [51]. We leverage the key ideas of CDLN in this paper and introduce horizontal slicing across multiple edge nodes and leverage SNNs to reduce footprint and resource requirements.

Algorithmic optimizations FFTs have been widely used for efficient convolution on large convolutional filters [52]. Building on this, faster algorithms specifically for smaller 3x3 convolutional filters (which are used extensively in VGGNet and ResNet), and smaller batch sizes specifically important in an edge scenario have been proposed [53]. These improvements are relevant and orthogonal to the CDLN paradigm with SNN-based architectures.

VI. Discussion

The proposed distribution method for DNN inference poses a number of challenges. A crucial consideration is the balance between computation and data transmission. Military coalition operations will likely be limited in both respects, but the optimal balance for deciding what to compute locally versus what to send to other nodes will vary over the coalition network and depend on the following factors:

- Local compute availability: the computational resources, and how available those resources are given any other local computing tasks;
- Local energy availability: what level of computation and data transmission the device's energy supply supports, and how long this will last given the projected device usage; and
- Network transmission capability: the speed and reliability of the network to transmit data between devices.

As discussed above, CDLNs can reduce both local computation and local energy consumption over conventional DNNs. The proposed distribution method can further reduce local computation by offloading the processing of higher CDLN layers either to the cloud or to other edge devices. However, this requires the device to transmit data over a potentially slow and unreliable network. A balance must be reached between the cost (both in terms of energy and computation) of running the network locally, and the cost of sending data to other nodes.

A naive implementation of the distributed evaluation method might transmit the full output of a CDLN layer, which for large networks dealing with high dimensional inputs could amount to several hundred kilobytes or more. This could prove unfeasible in a coalition network. A simple approach for reducing data transfer in conventional (non-spiking) CDLN implementations is simply to reduce the bit-depth of the network. This approach has been found to maintain accuracy with networks implemented using 8-bit integers, immediately quartering the required data transfer compared with standard floating point representations [54]. Further data reduction methods could also be employed, such as only transmitting layer outputs larger than a certain threshold. However, as discussed in section IV, an even more efficient method would be to implement the CDLN as an SNN and transmit binary spikes between nodes. This not only reduces data transmission requirements, but also improves the robustness of the transmission to unreliable networks: SNNs are inherently robust to stochastic variation in spike trains, meaning that they would cope well with noisy data transmission between nodes.

Another limitation of the proposed method is that it requires copies of the CDLN to be available either in the cloud or on other edge devices. This is unlikely to be the case when the coalition is rapidly formed and dynamic. This limitation may be addressed by researching how to make different DNNs interpretable to each other, such that the output of a layer in one network could be interpreted appropriately by a layer in a different network. In this way, layer outputs from a CDLN could be processed by CDLNs on other devices provided that these CDLNs were *similar enough* to the originating CDLN. The concept of inter-model interpretability has previously been formalized by Dhurandar et al. under the framework of δ-interpretability [55], [56], but research remains to be done on how to achieve it in practice.

VII. Conlusions

References

[1] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[2] A. Preece, F. Cerutti, D. Braines, S. Chakraborty, and M. Srivastava, "Cognitive computing for coalition situational understanding," in *2017 IEEE SmartWorld*. IEEE, 2017, pp. 1–6.

[3] A. Kott, A. Swami, and B. J. West, "The internet of battle things," *Computer*, vol. 49, no. 12, pp. 70–75, 2016.

[4] N. Suri, M. Tortonesi, J. Michaelis, P. Budulas, G. Benincasa, S. Russell, C. Stefanelli, and R. Winkler, "Analyzing the applicability of internet of things to the battlefield environment," in *Military Communications and Information Systems (ICMCIS), 2016 International Conference on*. IEEE, 2016, pp. 1–8.

[5] D. Verma, G. Bent, and I. Taylor, "Towards a distributed federated brain architecture using cognitive IoT devices," in *The Ninth International Conference on Advanced Cognitive Technologies and Applications*, 2017.

[6] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 5 2015.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[8] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, 2013.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[10] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition." IEEE, 2016, pp. 475–480.

[11] ——, "Energy-efficient and improved image recognition with conditional deep learning," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, p. 33, 2017.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[14] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib, "Scalable-effort classifiers for energy-efficient machine learning," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 67.

[15] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] 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.

[17] M. Parsa, P. Panda, S. Sen, and K. Roy, "Staged inference using conditional deep learning for energy efficient real-time smart diagnosis," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2017, pp. 78–81.

[18] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in Neuroscience*, vol. 13, p. 95, 2019.

[19] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," *arXiv preprint arXiv:1812.01739*, 2018.

[20] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.

[21] E. Hunsberger and C. Eliasmith, "Spiking deep networks with lif neurons," *arXiv preprint arXiv:1510.08829*, 2015.

[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)*. IEEE, 2015, pp. 1–8.

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

[25] P. Panda and K. Roy, "Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 299–306.

[26] 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, 2018.

[27] C. Lee, P. Panda, G. Srinivasan, and K. Roy, "Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning," *Frontiers in neuroscience*, vol. 12, 2018.

[28] 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*, 2018, pp. 7005–7015.

[29] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, 2018, pp. 1419–1428.

[30] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," *arXiv preprint arXiv:1901.09948*, 2019.

[31] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.

[32] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS computational biology*, vol. 3, no. 2, p. e31, 2007.

[33] G. Srinivasan, P. Panda, and K. Roy, "Stdp-based unsupervised feature learning using convolution-over-time in spiking neural networks for energy-efficient neuromorphic computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 4, p. 44, 2018.

[34] A. Tavanaei, Z. Kirby, and A. S. Maida, "Training spiking convnets by stdp and gradient descent," in *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil, July 2018, pp. 1–8.

[35] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "Stdp-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608017302903

[36] P. Ferré, F. Mamalet, and S. J. Thorpe, "Unsupervised feature learning with winner-takes-all based stdp," *Frontiers in computational neuroscience*, vol. 12, p. 24, 2018.

[37] J. C. Thiele, O. Bichler, and A. Dupret, "Event-based, timescale invariant unsupervised online deep learning with stdp," *Frontiers in Computational Neuroscience*, vol. 12, p. 46, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2018.00046

[38] C. Lee, G. Srinivasan, P. Panda, and K. Roy, "Deep spiking convolutional neural network trained with unsupervised spike timing dependent plasticity," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2018.

[39] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Combining stdp and reward-modulated stdp in deep convolutional spiking neural networks for digit recognition," *arXiv preprint arXiv:1804.00227*, 2018.

[40] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, Montréal, Canada, 2015, pp. 3123–3131.

[41] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Amsterdam, The Netherlands: Springer, 2016, pp. 525–542.

[42] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[43] M. Suri, D. Querlioz, O. Bichler, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Bio-inspired stochastic computing using binary cbram synapses," *IEEE Transactions on Electron Devices*, vol. 60, no. 7, pp. 2402–2409, 2013.

[44] D. Querlioz, O. Bichler, A. F. Vincent, and C. Gamrat, "Bioinspired programming of memory devices for implementing an inference engine," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1398–1416, 2015.

[45] G. Srinivasan, A. Sengupta, and K. Roy, "Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip stdp learning," *Scientific reports*, vol. 6, p. 29545, 2016.

[46] G. Srinivasan and K. Roy, "Restocnet: Residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing," *Frontiers in Neuroscience*, vol. 13, p. 189, 2019.

[47] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015. [Online]. Available: http://arxiv.org/abs/1510.00149

[48] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[50] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, 2016.

[51] H. Lee and J. Shin, "Anytime neural prediction via slicing networks vertically," *CoRR*, vol. abs/1807.02609, 2018.

[52] M. Mathieu, M. Henaff, and Y. LeCun, “Fast training of convolutional networks through ffts,” *CoRR*, vol. abs/1312.5851, 2014.

[53] A. Lavin, “Fast algorithms for convolutional neural networks,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4013–4021, 2016.

[54] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[55] A. Dhurandhar, V. Iyengar, R. Luss, and K. Shanmugam, “A formal framework to characterize interpretability of procedures,” *arXiv:1707.03886*, 2017.

[56] ——, “TIP: Typifying the interpretability of procedures,” *arXiv:1706.02952*, 2017.