EMPIRICAL ANALYSIS OF IEEE754, FIXED-POINT AND POSIT IN LOW PRECISION MACHINE LEARNING

Ștefan-Dan Ciocîrlan, Teodor-Andrei Neacșu, Răzvan-Victor Rughiniș[3]

Deep neural networks have changed the current algorithms' results in applications such as object classification, image segmentation or natural language processing. To increase their accuracy, they became more complex and more costly in terms of storage, computation time and energy consumption. This paper attacks the problem of storage and presents the advantages of using different number representations as fixed-point and posit numbers for deep neural network inference. The deep neural networks were trained using the proposed framework Low Precision Machine Learning (LPML) with 32-bit IEEE754. The storage was first optimized by the usage of knowledge distillation and then by modifying layer by layer the number representation together with the precision. The first significant results were made by modifying the number representation of the network but keeping the same precision per layer. For a 2-layer network (2LayerNet) using 16-bit Posit, the accuracy is 93.45%, close to 93.47%, the accuracy for using 32-bit IEEE754. Using the 8-bit Posit decreases the accuracy by 1.29%, but at the same time, it reduces the storage space by 75%. The usage of fixed point representation showed a small tolerance in the number of bits used for the fractional part. Using a 4-4 bit fixed point (4 bits for the integer part and 4 bits for the fractional part) reduces the storage used by 75% but decreases accuracy as low as 67.21%. When at least 8 bits are used for the fractional part, the results are similar to the 32-bit IEEE754. To increase accuracy before reducing precision, knowledge distillation was used. A ResNet18 network gained an 0.87% increase in accuracy by using a ResNet34 as a professor. By changing the number representation system and precision per layer, the storage was reduced by 43.47%, and the accuracy decreased by 0.26%. In conclusion, with the usage of knowledge distillation and change of number representation and precision per layer, the Resnet18 network had 66.75% smaller storage space than the ResNet34 professor network by losing only 1.38% in accuracy.

Keywords: Number representation systems, IEEE754, Machine Learning, Knowledge Distillation, Posit

1. Introduction

Artificial intelligence is identified as the solution to problems for which classical algorithmic produce primitive results. The subfield of artificial intelligence that has aroused major interest from the perspective of both research

[3]University POLITEHNICA of Bucharest, Dept. of Computer Science, Romania, e-mail stefan_dan.ciocirlan@upb.ro

and industry is represented by deep learning [6]. The core of this type of artificial intelligence is represented by deep neural networks. As the difficulty of solving problems using neural networks increases, they have become progressively more complex from an architectural point of view. This complexity in deep neural networks translates into the number of layers used in their architecture. If the number of layers increases, by default, the number of neural network parameters increases, which leads to an increase in the memory required for their storage. In addition, the running of such deep models must be performed on systems with high computational power. This paper aims to solve the problem of memory and running time of deep neural networks. The approach to solving this problem involves changing the classical numerical representation used by deep neural networks (32-bit IEEE754), applying the methods of knowledge distillation on complex neural networks and using a variable precision for each layer. We propose a framework (LPML-Low Precision Machine Learning Framework) for machine learning that can handle IEEE754 and other number representation systems such as Fixed-Point and Posit. In terms of precision, the framework can use 8-bit, 16-bit, and 32-bit Posit and Fixed-Point [13] with any bit configuration for the integer and fractional part. The experiment will be performed on two deep neural network architectures: a fully connected neural network and a ResNet18 [5] network. Training these networks using variable precision for each layer is an idea already tested in [8]. The resulting neural network architecture (including the precision obtained for each layer) will be used to create a new model that will be driven by a neural network teacher using the methods of knowledge distillation. In the end, we will get a reduced neural network from the perspective of the memory used, and its accuracy will be minimally degraded. The ideas tested from [8] and [9] will be combined to improve the solution. The improvement brought is that the architecture used for the knowledge distillation will contain a student with variable precision for each layer (in the article the student used a uniform precision). In addition, deep neural networks will use Fixed-Point and Posit number representation systems to represent data, gradients, and weights. The solution to this problem is divided into three stages: building a machine learning framework that allows using Fixed-Point and Posit, training the neural networks obtained using knowledge distillation and determining the optimal accuracy for each layer.

2. **Background**

Deep neural networks are the best solution when it comes to computer vision. These neural networks produce impressive results in classifying, segmenting and detecting objects. In recent years, these neural networks, in addition to increasing the number of layers, may contain parallel layer structures such as the inception type in the article [14]. With this increasing complexity, systems need more and more computing power and storage memory.

The best-known number representation system in deep neural networks is the 32-bit IEEE754 floating-point. For constrained systems (embedded systems, systems used in IoT) in terms of memory and processing power, the use of the IEEE754 number representation system can lead to a significant increase in response time and costs. This impediment leads to the use of less complex neural networks (built using a small number of layers) to maintain a short response time. To solve this problem, a major interest has been directed towards the use of the fixed-point number representation system because the computational effort is significantly reduced and the hardware implementation is simpler. Another number representation system whose computational effort is similar to IEEE754 but has a better dynamic range and produces a better accuracy is posit. The posit number representation system introduced by John L. Gustafson and Isaac Yonemoto [4] has produced superior results to the IEEE754 floating-point in the field of deep neural networks, but the hardware support for it is still under development [2].

Reducing memory and increasing the inference speed in neural networks has been a topic of interest for researchers in both cloud computing and embedded systems. Thus, numerous articles have been written in this field. Next it will be presented the ideas in the articles that provide important results for the problem and represent the starting points for a solution.

In the article [10] it is presented the benefits of the quantization operation in deep neural networks for low-power embedded systems. This operation is performed on a trained neural network that uses 32-bit IEEE754 floating-point and involves optimum conversion to 8-bit or 16-bit integers of the weights so that accuracy is not significantly impaired. In general, the weights of the deep neural networks have a zero-centered weight distribution similar to a Gauss distribution, and the quantization operation maintains this property. The authors of this article have developed the MicroAI framework which, compared to the existing frameworks for embedded systems: MicroLM, emlearn, microTVM, offers quantization operation, easy modification of architectures that uses convolution operations and the possibility of launching on a larger family of embedded systems. The article [12] presents the trade-offs between the complexity of deep neural networks and the resulting accuracy. The memory used by deep neural networks can be reduced either by training directly using a low precision of the parameters or by using the quantization operation presented in the previous article. The number representation used by neural networks in this article is fixed-point and the main idea is to use mathematical analysis to determine the boundaries between fixed-point numbers in order to reduce the noise produced by quantization operations. It details the relationship between the cost of representing the parameters and the accuracy of the network: deep neural networks may be reduced in terms of the precision of the parameters, but after a certain precision, the accuracy will deteriorate significantly.

The previous articles had as a solution for the reduction of memory in deep neural networks the use of fixed-point representation of parameters. Another solution is addressed in the article [1] where the parameters will use the posit number representation system proposed in [4]. The posit is a new type of number representation that seeks to solve the shortcomings of IEEE 754 floating-point standard. The most impressive advantages of posit would be: they can represent a wider range using fewer bits, they produce superior arithmetic accuracy, they do not produce overflow, underflow or NaN type error and the hardware for the implementation of posit computers are less complex. The article [1] uses the posit number representation system to represent data, gradients, and weights in deep neural networks. The detailed comparison between IEEE754 floating-point, fixed-point and posit is based on the elements EMAC (Exact Multiply and Accumulated) because in neural networks each neuron performs a function that gathers the weights multiplied by the input. For the 3 types of number representation system the same deep neural network architecture is used but the EMACs will be different for each type. The results show that the posit produces important results for deep neural networks with low precision (less than 8 bits) compared to those produced using IEEE754 floating-point and fixed-point. In terms of energy consumed, posit and floating-point are much more expensive compared to fixed-point but fixed-point performance is the lowest.

The article [8] raises the issue of reducing the precision of parameters in layer-level neural networks. This article shows that each layer has a different tolerance for changing the precision of the parameters depending on their position in the neural network architecture. The result of this article shows that the choice of a certain precision for each layer produces a superior result in terms of accuracy compared to deep neural networks that use a constant precision for all layers. These results were obtained using already trained deep neural networks and the weights of these networks use the 32-bit IEEE754. Memory reduction using such an approach will be achieved by finding the minimum precision required for each layer. In order to obtain the above specifications, the precision of each layer of the deep neural network will be modified in turn. The concept of knowledge transfer has been detailed in the article [3] and is based on a teacher-student architecture: the teacher will be a deep neural network already trained and the student is represented by a model with another architecture that will "take over" knowledge from the teacher. An important benefit of this learning technique is that the learning process for the student neural network will take place in a significantly short time. The idea behind this article is that if we have a trained model but want to build another model with a different architecture, the new model can transfer knowledge from the trained one without the need for training from scratch. This teacher-student architecture was used in the ImageNet competition where less complex neural networks were teachers for the student deep neural network. In the article [9]

teacher-student architecture is used to transfer knowledge from a deep neural network to a low-layer network so the student neural network will be reduced. The teacher neural network will have a 32-bit IEEE754 number representation system and the student neural network will also use IEEE754 but the precision will be decreased uniformly. The article proposes three methods to distil knowledge between teacher and student:

(1) The teacher and the student are trained simultaneously: for each example of training, the result of the teacher network being taken into account when training the student.
(2) The teacher is already trained and the student is the neural network that will be trained which implies that the backward propagation will be carried out only on the student network.
(3) This method is similar to the previous one, except that the student neural network will initially have an accuracy of 32 bits and this will be reduced until the desired result is reached.

Compared to deep neural network training with low precision, the use of the methods outlined above can significantly improve accuracy. The disadvantages of this approach would be the use of an additional deep neural network and the computational cost that will be increased because forward propagation will also be performed by the teacher network (if method 1 is used it will be necessary to propagate back to the teacher network as well).

3. **Framework Architecture**

3.1. **Datasets**

The input data sets for LPML are MNIST and CIFAR10 were chosen. MNIST is a data set that contains handwritten images, which means that the number of classes is 10. Images use a single colour channel (grayscale) and are $28 \times 28 \times 1$. The MNIST data set is divided into 60000 images representing the training set and 10000 images to be used for testing. CIFAR10 is a set of images that contain various objects or animals. The number of possible classes is 10 and these are airplanes, cars, birds, cats, deer, dogs, frogs, horses, boats and trucks. The images use 3 colour channels (RGB) and their size is $32 \times 32 \times 3$. The CIFAR10 data set is divided into 50000 images for training and 10000 images that will be used for testing.

3.2. **LPML Framework**

Low-Precision Machine Learning (LPML) is a framework which offers the possibility of using three number representation systems (IEEE754, Posit and Fixed-Point) to produce deep neural networks. The networks can be trained classic or using the process of knowledge distillation. After training, LPML can optimise the precision for every layer with the option of using a different number representation system than the one used for training. To obtain these functionalities LPML implemented the components presented in Figure **??**:

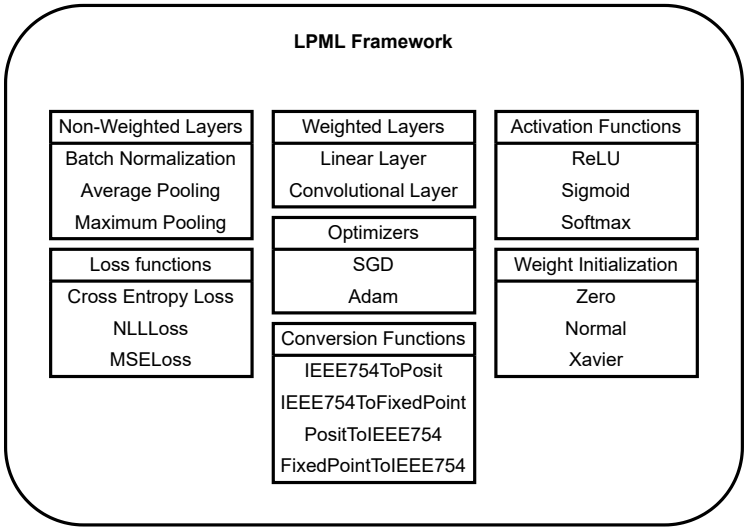

FIGURE 1. Architecture of LPML framework

non-weighted layers, weighted layers, activation functions, loss functions, optimizers, weight initialisation and conversion functions. The conversion functions are necessary for layer precision optimisation, while the other components are used for classical training and knowledge distillation training under uniform precision. With time the palette of options inside the components might increase to adapt to a higher number of deep neural network models, but for current analysis and scope, they are sufficient. The LPML framework uses the SoftPosit Python package for Posit and the fixedpoint Python package for Fixed-Point. The data, weights and gradients are stored as NumPy arrays. This has the advantage of adding future number representation systems as Object types but has the disadvantage of running single-threaded on CPU. The API for creating deep neural networks is similar to the PyTorch framework [11] (LPML copies the architecture of PyTorch). TensorFlow and PyTorch models can be converted to LPML framework easily. For the functions where the power/exponential or the logarithm operations are needed Posit and Fixed-Point are converted to IEEE754 the operation is done on IEEE754 and the result is converted back.

The classic training process of LPML is presented in Figure **??**. The input is shown in black, representing the data set, the desired deep neural network model and the number representation system used. The data set is divided into training samples (red) and test samples (blue). LPML use the training samples to produce a trained model which uses the specified number representation system. The model will go through benchmarks which evaluate the storage size, accuracy, training time, and inference time. The resulting model has a uniform precision.

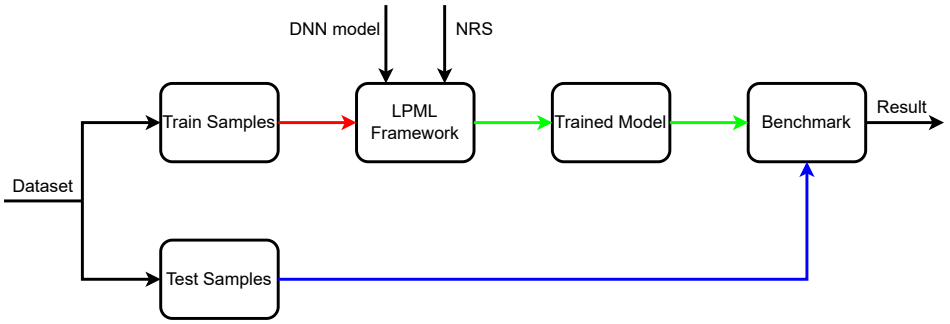

FIGURE 2. Training and evaluation using LPML (Low Precision Machine Learning).

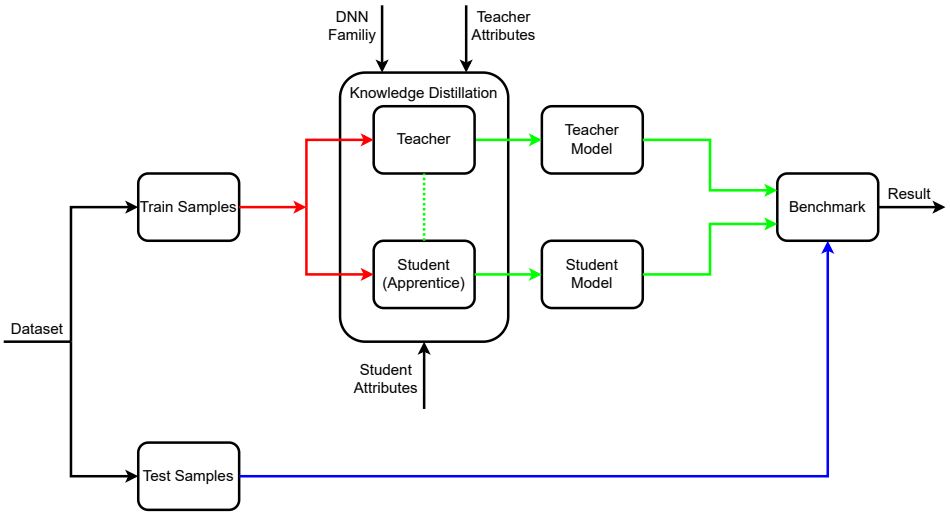

FIGURE 3. Training and evaluation using knowledge distillation

Knowledge distillation [7] is a technique used for training small deep neural networks with the help of larger networks. The LPML approach for knowledge distillation is presented in Figure **??**. The data set is similarly parted into train samples (red) and test samples (blue). The knowledge distillation process takes as input the train samples together with the desired deep neural network family (eg. Resnet) and attributes for the teacher and the student networks (number of layers, layer types, NRS). The teacher network is considered the larger network from which the smaller network (student) learns. The benchmarks applied to the resulting networks offer details about their storage size, training time, inference time, and accuracy. The interesting aspect of knowledge distillation training is that the accuracy of the small model is higher than if it was trained classically. There are multiple ways of using knowledge distillation, but for the experiments and evaluations proposed

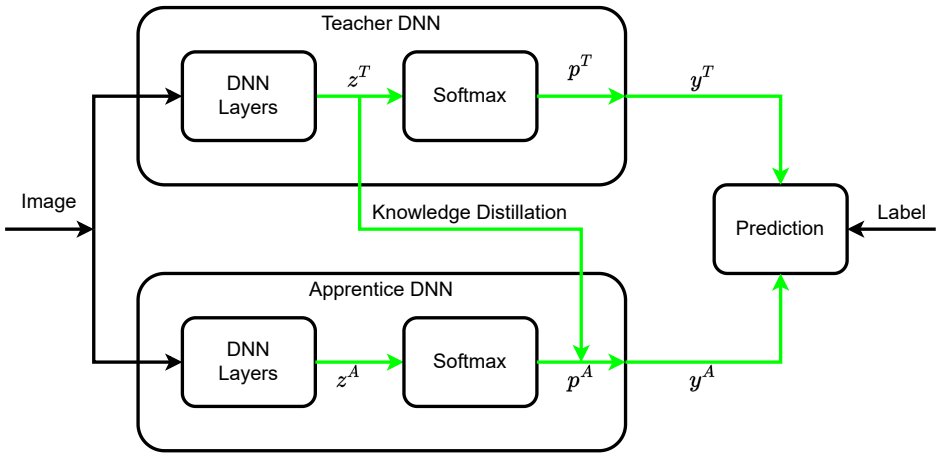

FIGURE 4. Knowledge distillation training method

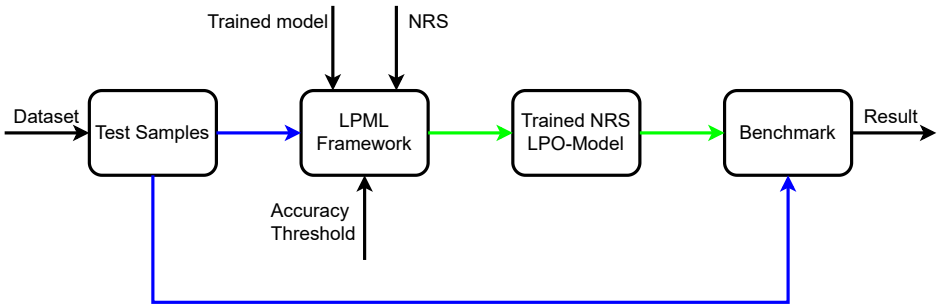

FIGURE 5. LPML Precision optimization architecture

the one where the teacher and student are training in the same was chosen. The method is presented in Figure **??**. The next loss function is used:

$$L(x; W_T, W_A) = \alpha H(y, p^T) + \beta H(y, p^A) + \gamma H(z^T, p^A) \tag{1}$$

The values for the hyper-parameters are $\alpha = 1, \beta = 0.5, \gamma = 0.5$. Knowledge distillation can be integrated with precision optimisation so the student network will use a uniform lower precision NRS. Given the results from [9] a more simple and modular process was chosen where knowledge distillation and precision optimisation are different stages of the pipeline.

LPML's precision optimisation reduces the storage size of deep neural networks by using different number representation systems with configurable precision. It can change the data, the weights, and the gradients of a network independently for every layer. The palette of number representation systems is formed out of: single-precision IEEE754, 8 bits Posit with exponent size 0 ($es = 0$), 16 bits Posit with $es = 1$, 32 bits Posit with $es = 2$ and configurable Fixed-Point. The process of layer precision optimisation is presented in Figure **??**. The test samples are taken out of the data set. The LPML receives the

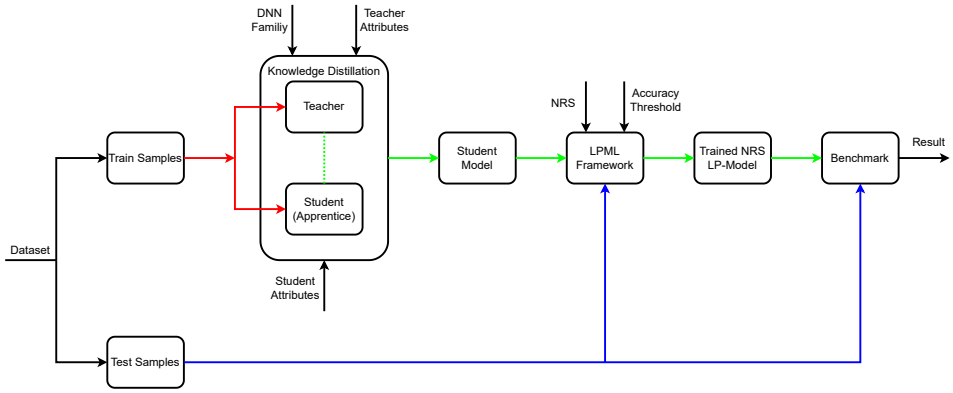

FIGURE 6. LPML pipeline architecture

trained model and the desired number representation system to be used for layer precision optimisation. The framework produces a network converted to the highest precision of the NRS. The layer index is initialised with the index of the second layer (the first and the last layer are kept at the highest precision possible) and current precision with the highest precision possible. The accuracy of the model is computed using the test samples. If the accuracy degradation (initial model accuracy minus current model accuracy) is lower than the accuracy threshold the precision decrease, the layer keeps its value and a new model is generated. Otherwise, the precision increase to the previous value for the current layer, the previous model is regenerated, then the layer index is increased and the precision resets to the highest possible value. This process runs until it arrives at the last layer, and then the resulting model passes through benchmarks for accuracy, storage size, and inference time.

Figure **??** presents the pipeline for knowledge distillation training and layer precision optimisation. The data set is divided into train samples (red) and test samples (blue). The Train sample is used together with the parameters for deep neural network family, teacher attributes and student attributes to produce the student model. The student network goes through the process of layer precision optimisation. The benchmarks use the resulting model for computing the storage size, accuracy and inference time.

4. **Results**

For the first LPML evaluation, the MNIST data set is used for training a neural network formed out of two fully connected layers followed by a logSoftmax layer. The learning rate of 0.01, the batch size of 64, and the number of epochs 3 were used for training. The time results for forward and backward propagation are presented in Table **??**. The total time for Posit and Fixed-Point was estimated. The software overhead is too high for doing feasible training using other number representation systems than IEEE754. From

TABLE 1. Forward and Backward Propagation Time

Number Representation system	Precision	Forward Propagation (one)	Forward Propagation (total)	Backward Propagation (one)	Backward Propagation (total)
IEEE754	32	0.0006s	13.1s	0.0006s	12.6s
Posit	8	43.2s	33.75h	74.5s	58.20h
Posit	16	42.6s	29.25h	73.9s	57.73h
Fixed-Point	4-4	455s	355h	878s	685h
Fixed-Point	8-8	460s	359h	899s	702h

TABLE 2. Accuracy and memory reduction for different number representation systems on simple network

Number representation system	Precision	Accuracy	Δ Acc	Memory reduction
IEEE754(8,23)	32	93.47%	+0.00%	0%
Posit(8,0)	8	92.18%	-1.29%	75%
Posit(16,1)	16	93.45%	-0.02%	50%
Posit(32,2)	32	93.62%	+0.15%	0%
Fixed-Point(4,4)	8	67.21%	-26.26%	75%
Fixed-Point(4,8)	12	93.28%	-0.19%	62.5%
Fixed-Point(8,8)	16	93.42%	-0.05%	50%

this point on Posit and Fixed-Point were used only for inference and precision optimisation. A consistent improvement of the software version of Posit or Fixed-Point or a hardware implementation can reopen the subject of training. The LPML framework is ready for this and can be used.

The IEEE754 resulting network has an accuracy of 93.47%. Uniform precision conversion was used for the fully connected layers and the results are presented in Table **??**. The Fixed-Point number representation system can show the boundaries of neural networks. The decrease to 4 bits for the fraction part produces an accuracy degradation of 26.26%, incomparable with the decrease of the integer part to 4 bits (0.19% accuracy degradation). In the tests going to 2 bits for integer parts degrades the accuracy below 50%. This validates that Fixed-Point is more sensitive to fraction size than integer size in neural networks. Posit has better results than Fixed-Point even on this small network with the same precision. The interesting result is that Posit(32,2) increases the accuracy of the network by 0.15%. Posit(16,1) reduce the memory used for the fully connected layers by 50% with an accuracy degradation as small as 0.02%. Posit(8,0) offer a solution for a high reduction of memory (75%) with a cost on accuracy of 1.29%. Given the bad performance of Fixed-Point for LPML even on a small network, but also in the next experiments their result was omitted. The next LPML experiments present the usage of Posit and IEEE754.

TABLE 3. Accuracy for ResNet18 on different training methods

Method	Layer Precision	Accuracy	Memory reduction
Classic	32-32-32-32-32-32-32-32-32-32	94.11%	0%
KD	32-32-32-32-32-32-32-32-32-32	94.98%	0%
KD+PO	32-32-32-16-16-16-16-8-8-32	93.85%	43.47%

To test LPML knowledge distillation training and layer precision optimisation the CIFAR-10 data set, the ADAM optimizer, the learning rate of 0.005 and the accuracy threshold of 2% were used. The deep neural network family is Resnet, Resnet34 is the teacher network and Resnet18 is the student network. The increase of accuracy for knowledge distillation training versus classic training is validated in Table **??** with a value of 0.87%. The usage of layer precision optimisation on the knowledge distillation trained model has an accuracy degradation of 0.26% compared to the classic training a reduction of memory for fully connected layers and convolutional layers of 43.47%. Resnet34 has an accuracy of 95.23% classic trained. The Resnet18 trained with knowledge distillation and layer precision optimisation reduces the storage size for fully connected layers and convolutional layers with 66.75% for a degradation in accuracy of 1.38%. The Resnet18 trained with knowledge distillation without layer precision optimisation reduces the storage for the same layers by 41.19% with degradation in accuracy as small as 0.25%. The usage of knowledge distillation and layer precision optimisation can reduce the storage space of a deep neural network by half with a degradation close to 1%.

5. **Conclusion**

LPML framework offers a way of training deep neural networks with Posit and Fixed-Point. Given the current software and hardware implementations, this type of training is inefficient regarding computation time. Posit is five times, and Fixed-Point is six times orders of magnitude slower. Fixed-Point might be used for inference and storage on energy-constraint devices. The results on small networks show an accuracy degradation of 0.19% for a storage size reduction of 62.5% for fully connected layers. Posit offers a better trade-off for accuracy. On a small network with two fully connected layers followed by one LogSoftmax layer, 16-bit Posit ($es = 1$) reduces the fully connected size by 50% with a loss in accuracy of 0.02%.

For the deep neural networks, the LPML knowledge distillation training combined with layer precision optimisation reduces the size of a Resnet34 network by 66.75% with a loss of 1.38% in accuracy or without layer precision optimisation, a size reduction of 41.19% with a loss of only 0.25%.

The accuracy increase for knowledge distillation training was validated by 0.87% on a Resnet18 trained with a Resnet34 versus a classic trained Resnet18.

The use of layer precision optimisation reduces the size of fully connected and convolutional layers by 43.47% with a loss of 1.13%.

Acknowledgment

This work was partly supported by Bitdefender's University PhD Grants Program 2019-2022 and by the Google IoT/Wearables Student Grants 2022.

REFERENCES

[1] *Carmichael, Zachariah and Langroudi, Hamed F and Khazanov, Char and Lillie, Jeffrey and Gustafson, John L and Kudithipudi, Dhireesha.* Deep positron: A deep neural network using the posit number system. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1421–1426. IEEE, 2019.

[2] *Chaurasiya, Rohit and Gustafson, John and Shrestha, Rahul and Neudorfer, Jonathan and Nambiar, Sangeeth and Niyogi, Kaustav and Merchant, Farhad and Leupers, Rainer.* Parameterized posit arithmetic hardware generator. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 334–341. IEEE, 2018.

[3] *Chen, Tianqi and Goodfellow, Ian and Shlens, Jonathon.* Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[4] *Gustafson, John L and Yonemoto, Isaac T.* Beating floating point at its own game: Posit arithmetic. *Supercomputing frontiers and innovations*, 4(2):71–86, 2017.

[5] *He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian.* Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] *Heaton, Jeff.* Applications of deep neural networks. *arXiv preprint arXiv:2009.05673*, 2020.

[7] *Hinton, Geoffrey and Vinyals, Oriol and Dean, Jeff and others.* Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[8] *Judd, Patrick and Albericio, Jorge and Hetherington, Tayler and Aamodt, Tor and Jerger, Natalie Enright and Urtasun, Raquel and Moshovos, Andreas.* Reduced-precision strategies for bounded memory in deep neural nets. *arXiv preprint arXiv:1511.05236*, 2015.

[9] *Mishra, Asit and Marr, Debbie.* Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv preprint arXiv:1711.05852*, 2017.

[10] *Novac, Pierre-Emmanuel and Boukli Hacene, Ghouthi and Pegatoquet, Alain and Miramond, Benoît and Gripon, Vincent.* Quantization and deployment of deep neural networks on microcontrollers. *Sensors*, 21(9):2984, 2021.

[11] *Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and others.* Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[12] *Sakr, Charbel and Kim, Yongjune and Shanbhag, Naresh.* Analytical guarantees on numerical precision of deep neural networks. In *International Conference on Machine Learning*, pages 3007–3016. PMLR, 2017.

[13] *Shin, Sungho and Hwang, Kyuyeon and Sung, Wonyong.* Fixed-point performance analysis of recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE, 2016.

[14] *Szegedy, Christian and Liu, Wei and Jia, Yangqing and Sermanet, Pierre and Reed, Scott and Anguelov, Dragomir and Erhan, Dumitru and Vanhoucke, Vincent and Rabinovich, Andrew.* Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.