Design of Hardware-Friendly Memory Enhanced Neural Networks

Ann Franchesca Laguna*, Michael Niemier*,X. Sharon Hu*
*Department of Computer Science and Engineering,, University of Notre Dame
Notre Dame, IN 46556, USA, Email: {alaguna,mniemier,shu}@nd.edu}

***Abstract*—Neural networks with external memories have been proven to minimize catastrophic forgetting, a major problem in applications such as lifelong and few-shot learning. However, such memory enhanced neural networks (MENNs) often require a large number of floating point-based cosine distance metric calculations to perform necessary attentional operations, which greatly increases energy consumption and hardware cost. This paper investigates other distance metrics in such neural networks in order to achieve more efficient hardware implementations in MENNs. We propose using content addressable memories (CAMs) to accelerate and simplify attentional operations. Our hardware friendly approach implements fixed point L_∞ distance calculations via ternary content addressable memories (TCAM) and fixed point L_1 and L_2 distance calculations on a general purpose graphical processing unit (GPGPU). As a representative example, a 32-bit floating point-based cosine distance MENN with $M \cdot D$ multiplications has a 99.06% accuracy for the Omniglot 5-way 5-shot classification task. Based on our approach, with just 4-bit fixed point precision, a L_∞- L_1 distance hardware accuracy of 90.35% can be achieved with just 16 TCAM lookups and $16 \cdot D$ addition and subtraction operations. With 4-bit precision and a L_∞-L_2 distance, hardware classification accuracies of 96.00% are possible. Hence, 16 TCAM lookups and $16 \cdot D$ multiplication operations are needed. Assuming the hardware memory has 512 entries, the number of multiplication operations is reduced by 32x versus the cosine distance approach.**

***Index Terms*—TCAM, Neural Networks, Associative Memories, One-Shot Learning, MANN, Nearest Neighbor**

I. Introduction

Traditional neural networks face a number of challenges. One of them is the inability for a network to recognize unseen classes without retraining. For example, a neural network trained to only recognize cats and dogs will categorize a giraffe as either cats or dogs and not a giraffe. Retraining the network to recognize giraffes can help, but may interfere with the initial learning tasks. That is, when the previous neural network is trained to recognize a giraffe, the neural network may fail to recognize the cats and dogs that it was previously trained to identify, a problem called *catastrophic interference* or *catastrophic forgetting* [1], [2]. In addition, neural networks typically require massive amounts of data to be able to learn while humans typically only need a few and sometimes even just one example of a new class. To address these challenges, adding a dedicated attentional memory has been proposed [3], [4], [5]. In this paper, we refer to neural networks with memory as Memory Enhanced Neural Networks (MENNs). Examples of MENNs include Neural Turing Machine (NTMs) [3], Memory Augmented Neural Networks (MANNs) [4] and Memory Networks [5].

An important application of a MENN is one-shot or few-shot learning. One/few-shot learning refers to rapid learning from one or a few examples. The dataset is split such that the classes in the training set are disjoint from the classes in the test set. The number of classes is large while the number of images per class is typically small. Given a set of support images with one/few image(s) per class and a query image, the goal of one/few shot learning is to be able to identify which support image the query image is most similar to. The training strategy for an N-way K-shot classification takes N different classes and from those classes takes K samples to create a support set S of $N \times K$ images. To compute the necessary gradients during training, a batch B is also sampled from the K classes to be used as query images. The network "learns to learn" from the support set S to minimize the error in predicting the labels in batch B. This is then regarded as an episode. For each iteration, a new episode is carried out to train the network.

In MENNs, a neural network's attentional memory is similar to a human's working memory. When humans make decisions, we need to remember and analyze various information simultaneously. As we learn, we identify information that is essential to focus our *attention* on. Therefore, the attentional memory stores relevant features. Attentional memories are retrieved based on their similarity with a query vector. General purpose graphical processing units (GPGPUs) are commonly used to do parallel computations. When implementing MENNs, the GPGPU becomes more memory-bound as M and D increase (where M is the number of entries and D is the number of dimensions of each entry). This motivates our work to explore memory-friendly hardware approaches to perform MENN distance metrics calculations, as well as subsequent effects on a network's accuracy.

An attractive hardware alternative for performing, parallel comparisons is to use content addressable memories (CAMs). A variant of a CAM is a ternary CAM (TCAM) which allows for the presence of a wildcard or "don't care" state. This is useful for encoding ranges [6], [7] employed in this work.

However, straightforward employment of TCAMs face several key challenges that we address in this work. First, in MENNs, as noted above, cosine similarity calculations necessitate many multiplication and division operations – which TCAM operations do not support. Thus, we systematically study alternative distance metrics, including L_1, L_2 and L_∞ (that may be more TCAM friendly), to determine the resulting impact on accuracy, physical memory size required and the number/type of operations to compare a query with a network's learned memory. Second, feature vectors in MENNs are usually represented in floating point form. Storing these feature vectors in a TCAM memory requires that floating point entries be converted to a fixed point representation. We have investigated the influence of bit precision on accuracy for a chosen distance metric. While decreasing bit precision leads to lower accuracy, reasonable classification accuracies are possible / a network can learn with just 4-bit fixed point precision. Using alternate distance metrics and precision reduction can be utilized because of the high dimensional property of neural network features [8]. Third, energy consumption and latency of a TCAM depends largely on the size of the physical memory

978-3-9819263-2-3/DATE19/©2019 EDAA

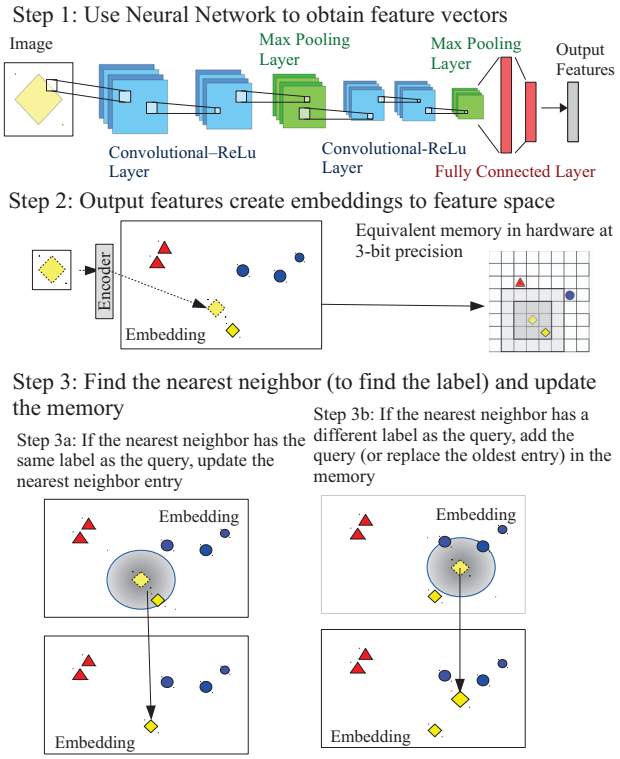


Fig. 1: Step by step implementation of MENN.

array [9]. Reducing the number of TCAM entries can help in lowering energy consumption but may adversely impact a MENN's accuracy. If the number of memory entries are too small the network does not learn. However, larger memories do not always improve a MENN's accuracy as some entries can simply be duplicates of the other entries. We explore how the number of memory entries impacts the training time and accuracy.

II. Design Space Exploration Methodology

In this section, we detail the methodologies for exploring the design space of hardware-friendly attentional memories in MENNs. Fig. 1 illustrates the four main steps in training an MENN. The first step is to extract relevant features from the training dataset through a neural network component in the MENN (Section II-A). These output features are mapped into the feature space and stored in the memory (Step 2). Mapping the attentional memory to its equivalent physical memory requires converting from floating point representation to quantized fixed point representation (Section II-B). To find the class of a given query, a nearest neighbor search using a given distance metric is performed on the memory entries. Different distance metrics are discussed in (Section II-C). The memory (step 3) and the neural network (step 4) are then updated using the new information (Section II-D).

A. Feature Extraction using Neural Networks

The parameters of neural networks are typically trained to learn differentiating features between a set of given classes. Traditionally, neural networks culminate with a softmax classifier that generates probability distributions to determine the most likely class a particular object belongs to. For a MENN this classifier is removed and the output features of the final neural network layer are used to embed the object into a feature space.

A dataset commonly used for one/few-shot learning is the omniglot dataset. The omniglot dataset [10] has a large number of classes and only a few examples for each class (i.e., as opposed to the MNIST database of hanwritten digits which has few classes and a large number of examples for each class.) Omniglot is a commonly used benchmark for MENNs [4], [11], [12], and contains 1623 characters from 50 different alphabets; each character is handwritten by 20 different people. 1200 characters are used for training while the remaining characters are used for testing.

Similar to [11], the neural network used in our studies (Fig. 1 step 1) is composed of two, 3x3 convolutional-ReLU layers with 64 filters, a max-pooling layer, another two, 3x3 convolutional-Relu layers with 128 filters, and two fully connected layers. The output of the last fully connected layer forms the feature vector that will be passed to the memory. The number of neurons in the last fully connected layer equals the number of dimensions for each entry in the physical memory, where a dimension represents a feature.

B. Mapping attentional memory to physical memory

The structure of the attentional memory module has similarities to a CAM, where it is composed of rows of memory entries. Each element, representing a feature, in the attentional memory is represented in floating point form. For hardware implementation, these entries are converted to a fixed-point representation. The binary representation of each memory entry is concatenated and stored in each TCAM row. Straightforward conversion from floating point to fixed point does not work because, the TCAM will only return the exact match of a given query. The attentional memory searches for the nearest neighbor entry using a distance metric. Using the wildcard or don't care state in the TCAM enables a nearest neighbor search operation but these requires either hashing or range encoding.

Range encoding with no expansion (RENÉ) [6], [7], another TCAM similarity search, leverages binary reflected gray code (BGRC) range encoding. The query point is translated to a cube of increasing sizes until k-nearest neighbor are found. As the range size increases, the number of don't care states increases as well. Thus, range encoding can be used to compute for the nearest neighbor for L_1 and L_∞ (Sec. II-C).

C. Distance Metrics

The cosine distance metric that is commonly used in MENNs is calculated by finding the angular distance between two points with respect to the origin. Due to the high computational demand of evaluating cosine distance, L_1, L_2 and L_∞ can be considered as alternative distance metrics for measuring similarity in the nearest neighbor approach. Range searching can be used to implement L_1 and L_∞ distance and is adopted in our approach. L_2 distance, is the direct distance between two points but it requires a multiplication and square root operation, which is expensive to implement in hardware. L_1 and L_∞, on the other hand, only require addition and subtraction, which are simpler to implement in hardware. When mapped to TCAM implementations, L_∞ requires fewer

searches as compared to L_1. To compute each range distance in L_1 we need to change each dimension needing D lookups for one range query. L_∞ only requires one TCAM lookup for each range query and is therefore fast, but it does not retain the angular information which may lead to inexact and suboptimal solutions/query matches. In order to achieve a good balance between accuracy and hardware cost, we propose an approach that combines L_∞ and L_1/L_2.

D. Updating memory and neural network parameters

In a MENN, learning encompasses changing parameters of memory and the neural network (Fig. 1 step 3-4) These updates depend on the class of the nearest neighbors surrounding the query. The label of the nearest neighbor n_1 may be the same or not the same as the label of the query vector q (Fig. 1 step 3). If it is not the same, the query is added to the memory when the memory is not full. If the memory is full, the oldest memory is replaced. If it is the same, the memory A is updated by averaging in Euclidean space per Eq. 1 [11]:

$$A[n_1] \leftarrow \frac{q + A[n_1]}{||q + A[n_1]||} \quad (1)$$

Updating the neural network requires the computation of a loss function that determine the gradients for the neural network parameters. We use the triplet loss (Fig. 1 step 4) to bring same-labeled entries (i.e., positive neighbor n_+) close and differently-labeled entries (i.e., negative neighbor n_-) away from each other.

III. Experimental Results

Implementing an attentional memory for MENNs with TCAM cells is governed by the following parameters: D (dimensions for each entry), M (physical memory entries), P (precision bits used by each coordinate), $\mathcal{H}$ (the number of cubes for the nearest neighbor search), K (the number of nearest neighbor outputs of the L_∞ metric in the $L_\infty + L_1$ and $L_\infty + L_2$ approach). Depending on which distance metric is used, these parameters affect the classification accuracy, memory, and computational complexity.

A. Floating point to fixed point

As fixed point computations are simpler to implement in hardware than floating point computations, a study of the effect of the conversion from floating point to fixed point is conducted. From Tab. I (rows 1-2), changing from cosine to L_1 distance metric results in a 1-2% reduction in accuracy. L_1 distance however does not require multiplication as shown in (Tab. II rows 1-2). Tab. I (rows 2-3) further demonstrates that changing the number representation to fixed point further reduces the accuracy by another 4% for a one shot learning task because of the loss of information from quantization. These quantization errors are minimal in the case of the 5-shot case, where there are more samples for each class for comparison. (Accuracy loss is approximately 1%.) L_∞ does not perform well because of the loss of the angular information (Tab. I, row 4). However, as L_∞ requires fewer TCAM lookups and is not tied to the number of dimensions D (unlike L_1) (Tab. II, row 4), we propose leveraging a TCAM for L_∞ calculations and performing the additional computations for L_1 and L_2 (Tab. II, row 5-6) on other compute units such as a CPU or GPU.

TABLE I: Accuracies with different distance metrics

Distance Metric	32-bit	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
Cosine	Floating	96.59	99.06	95.0	98.5
L_1	Floating	94.93	97.61	93.08	97.64
L_1	Fixed	90.68	96.48	90.77	95.96
L_∞	Fixed	49.07	70.78	31.99	52.56
$L_\infty + L_1$ (K=16)	Fixed	85.68	95.14	90.04	95.91
$L_\infty + L_2$ (K=16)	Fixed	90.29	96.51	90.34	96.18

TABLE II: Required hardware operations for computing distance metric, loss and normalization

Operation	TCAM lookup	Sub/Add	Mult	Div
Computing for cosine	0	MD	MD	0
Computing for L_1 (not TCAM)	0	$2MD$	0	0
Computing for L_1 (in TCAM)	$2^D\mathcal{H}$	0	0	0
Computing for L_∞	$\mathcal{H}$	0	0	0
Computing for $L_\infty + L_1$	$\mathcal{H}$	KD	0	0
Computing for $L_\infty + L_2$	$\mathcal{H}$	KD	KD	0
Computing for Loss	0	2	3	0
Normalization	0	$2D$	D	D

By using L_∞ first we obtain a set of K approximate solutions. From those K possible solutions, we use more accurate distance metrics such as L_1 or L_2 to find an improved solution when compared to L_∞ alone (Tab. I row 5-6). While using L_2 instead of L_1 (Tab. II row 5-6) for the second stage would require $K \cdot D$ additional multiplications this is much less than the $M \cdot D$ multiplications of cosine distance since $K << M$. L_2 is preferable when there are fewer image samples as in the case for a 5-way 1-shot case (where the use of L_2 leads to 5% higher accuracy). However, as the number of support images increases, L_1 shows comparable results. L_2 however is more robust with quantization errors as shown in Tab. III. By using a TCAM, the complexity is no longer bounded by the number of entries M, which greatly helps in the scaling the TCAM up to more complex problems.

B. Fixed point representation and alternative distance metrics

To further decrease the physical memory, we study the effect on accuracy when the number of bits is reduced. We also look at alternative distance metrics to reduce computational cost without sacrificing the accuracy. Tab. III shows that a minimum of 4-bit accuracy is needed for the hardware memory module for the cases of L_1 distance, the combined L_∞ and L_1 distance, and the combined L_∞ and L_2 distance. The combined L_∞ and L_2 approach also is the most resilient with quantization errors. The 4-bit L_∞ + L_2 approach results in 96.00% accuracy which is comparable with its 32-bit equivalent of 96.51%. Still, neural networks typicalliy need at least 5-6 bits to compute for gradients [13]. As we are using the memory entries to compute the back-propagation gradients, we cannot reduce precision below 4-bits.

C. Evaluation of varying Memory Size

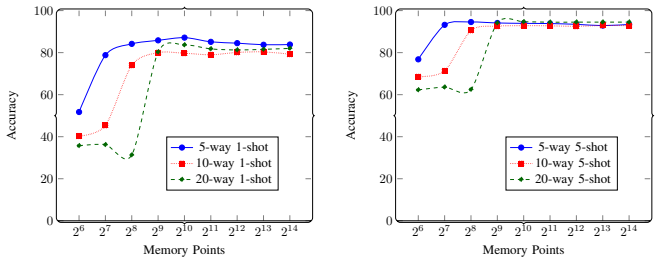
Increasing the physical memory (i.e., TCAM) size will increases energy consumption, etc. Therefore, we also study the effect of physical memory size on accuracy. Initial experiments consider 32-bit precision fixed point and L_1 distance. Fig. 2 illustrates the effect of varying the memory sizes in the case of 5-way, 10-way and 20-way classification tasks. It can be readily seen that the effective memory size increases as the complexity of the problem increases.

Further increasing the number of memory points does not necessarily improve the accuracy and can actually result in

TABLE III: Effect of varying the no. of precision bits for different distance metrics

No. of Precision Bits	L_∞		L_1	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
32	49.07	70.78	90.68	96.48
16	48.22	69.39	85.13	94.58
8	25.16	26.88	84.59	94.27
4	24.99	26.93	86.31	94.87
2	20.00	20.00	20.00	20.00
1	20.00	20.00	60.39	68.36

No. of Precision Bits	$L_\infty + L_1$		$L_\infty + L_2$	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
32	85.68	95.14	90.29	96.51
16	80.12	91.98	88.01	95.88
8	79.83	92.46	88.0	96.36
4	78.40	90.35	89.25	96.00
2	20.00	20.00	20.00	20.00
1	20.00	20.00	20.00	20.00



(a) One-shot (b) Five-shot

Fig. 2: Effect of varying the number of memory entries

lower accuracy. This observation can be explained as follows. While, a neural network continually updates its weights during backpropagation the record in the memory may become outdated. Large memory sizes fail to forget outdated information which conflicts with new memory entries. With larger physical memory, it takes more time before these memory entries are forgotten. This is also supported by the results in MANN [4], where memory wiping for each episode results in better accuracy and faster learning because memory from previous episodes can interfere with the current episode. For a 5-way, 10-way and 20-way Omniglot dataset classification, the effective memory sizes are 128, 256, 512, respectively. Increasing the memory beyond this does not further improve the accuracy of the system.

IV. Discussions and Conclusion

In this paper, we proposed using TCAMs using range encoding to implement the attentional memory in a MENN and investigated different distance metrics and their effects on classification accuracy. Using a TCAM can reduce the number of computations that must be done on GPUs (e.g., for cosine similarity metrics) and also can address memory bottlenecks in terms of speed and size on the GPU. Furthermore, the simpler operations can give substantial speedup when implemented in hardware [6], [14].

We evaluated the impact of reducing data representation and found that a minimum of 4-bit precision is necessary for the network to learn features from different classes. Alternative distance metrics, specifically L_1, L_2 and L_∞ (in lieu of cosine distance), were studied. L_∞ is the simplest to implement with TCAMs but suffers from lower accuracies for classification tasks. Thus, we introduced an approach that combines L_∞ with distance metrics such as L_1 and L_2 distances to achieve higher accuracy. For a 5-way 5-shot classification, the cosine distance reaches 99.06% accuracy with $512 \cdot 128$ multiplication operations. The combined $L_\infty + L_1$ distance metric hardware approach with just 4-bit precision resulted in 90.35% classification accuracy using 16 TCAM lookups and 16 addition and subtraction operations. The combined $L_\infty + L_2$ distance metric hardware approach (again with 4-bit precision) resulted in 96.00% classification accuracy using 16 TCAM lookups and $16 \cdot 128$ multiplications. The number of multiplication operations is reduced by 32x. For more complex problems, the reduction of the multiplication operations becomes more beneficial when M and D increases.

We will conduct energy and performance evaluations on TCAM-based MENNs. Furthermore, there is growing interest in developing TCAM cells based on emerging technologies such as resistive random access memory [15], ferroelectric field effect transistors [16], [17], etc. As such, as new logic/memory technologies mature, they may lead to multiplicative benefits via our approach – i.e., owing to faster, more energy efficient, or more compact TCAM cells.

Acknowledgment

This work was supported in part by ASCENT, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] M. McCloskey et al. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109 – 165. Academic Press, 1989.
[2] S. Hochreiter, et al. Learning to learn using gradient descent. In *Artificial Neural Networks - ICANN 2001, International Conference Vienna, Austria, August 21-25, 2001 Proceedings*, pages 87–94, 2001.
[3] A. Graves, et al. Neural turing machines. *CoRR*, abs/1410.5401, 2014.
[4] A. Santoro, et al. One-shot learning with memory-augmented neural networks. *CoRR*, abs/1605.06065, 2016.
[5] J. Weston, et al. Memory networks. *CoRR*, abs/1410.3916, 2014.
[6] A. Bremler-Barr, et al. Encoding short ranges in tcam without expansion: Efficient algorithm and applications. *IEEE/ACM Transactions on Networking*, 26(2):835–850, April 2018.
[7] A. Bremler-Barr, et al. Ultra-fast similarity search using ternary content addressable memory. In *Proceedings of the 11th International Workshop on Data Management on New Hardware, DaMoN 2015, Melbourne, VIC, Australia, May 31 - June 04, 2015*, pages 12:1–12:10, 2015.
[8] A. G. Anderson et al. The high-dimensional geometry of binary neural networks. *CoRR*, abs/1705.07199, 2017.
[9] B. Agrawal et al. Modeling tcam power for next generation network devices. In *2006 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 120–129, March 2006.
[10] B. M. Lake, et al. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
[11] L. Kaiser, et al. Learning to remember rare events. *CoRR*, abs/1703.03129, 2017.
[12] J. W. Rae, et al. Scaling memory-augmented neural networks with sparse reads and writes. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3621–3629, 2016.
[13] I. Hubara, et al. Quantized neural networks: Training neural networks with low precision weights and activations. *CoRR*, abs/1609.07061, 2016.
[14] D. Fujiki, et al. In-memory data parallel processor. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '18, pages 1–14, New York, NY, USA, 2018. ACM.
[15] C. C. Lin, et al. 7.4 a 256b-wordlength reram-based tcam with 1ns search-time and 14?? improvement in wordlength-energyefficiency-density product using 2.5t1r cell. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 136–137, Jan 2016.
[16] X. Yin, et al. Design and benchmarking of ferroelectric tcams. *submitted to IEEE Transactions on Circuits and Systems II*, 2018.
[17] X. Yin, et al. Design and benchmarking of ferroelectric FET based TCAM. In *DATE*, pages 1444–1449, March 2017.