Budget Restricted Incremental Learning with Pre-Trained Convolutional Neural Networks and Binary Associative Memories

Ghouthi Boukli Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel and Michel Jezequel
IMT Atlantique, Brest, France
name.surname@imt-atlantique.fr

***Abstract*—Thanks to their ability to absorb large amounts of data, Convolutional Neural Networks (CNNs) have become state-of-the-art in numerous vision challenges, sometimes even on par with biological vision. They rely on optimisation routines that typically require intensive computational power, thus the question of embedded architectures is a very active field of research. Of particular interest is the problem of incremental learning, where the device adapts to new observations or classes. To tackle this challenging problem, we propose to combine pre-trained CNNs with binary associative memories, using product random sampling as an intermediate between the two methods. The obtained architecture requires significantly less computational power and memory usage than existing counterparts. Moreover, using various challenging vision datasets we show that the proposed architecture is able to perform one-shot learning – and even use only a small portion of the dataset – while keeping very good accuracy.**

***Index Terms*—Incremental Learning, Transfer Learning, Convolutional Neural Networks, Associative Memories**

I. Introduction

For the past few years, Deep Neural Networks (DNNs), and in particular Convolutional Neural Networks (CNNs), have achieved state-of-the-art performance [1], [2], [3] in several domains of supervised learning, sometimes even being on par with the visual cortex [4]. DNNs rely on hundreds of millions of parameters that are trained to deal with large amounts of data. In this context a major drawback of the method is the need for intensive computation and memory usage during the learning phase. This limitation is critical for embedded systems such as smartphones or sensor networks.

A lot of effort has been driven towards optimized hardware implementations of DNNs [5]. For example in [6], the authors propose an architecture with state-of-the-art performance on the ImageNet challenge [7] using less than one megabyte, and in [8] a fixed point quantization of CNNs was introduced to reduce the network size. However, learning its parameters still require the processing of the whole dataset, involving many gradient computations. Moreover, the whole training dataset has to be stored in memory for learning.

Incremental methods provide solutions to process the learning data sequentially, using subsets of the training dataset. An incremental technique is defined as such [9]: a) it is able to learn additional information from new data (example-incremental), b) it does not require access to the original data used to train the existing classifiers (in order to limit memory usage), c) it preserves previously acquired knowledge (avoid catastrophic forgetting) and d) it is able to accommodate new classes that may be introduced with new data (class-incremental). Although models have been proposed and studied extensively during the last decades, finding a good compromise between accuracy and required resources remains challenging. Indeed, most of existing works retrain the model when receiving new data [10], [11], and reuse some prior data for the retraining process [9], [12].

In this paper we propose an incremental learning model with the following claims:

- It is possible to adapt the model to new data without retraining it,
- It uses much less computational power than existing counterparts,
- It approaches state-of-art accuracy on challenging vision datasets (CIFAR10, ImageNet),
- It dramatically decreases the memory usage (by several orders of magnitude compared to nearest neighbour search),
- It only requires a few learning examples.

We point out that these claims are of particular interest when targeting embedded applications.

We rely on an increasingly popular method to benefit from the accuracy of DNNs without the need to train them on the targeted dataset, termed "transfer learning" [13], [2]. The idea is to use pre-trained CNNs on large datasets as feature extractors, and retrain the final layer of DNNs.

In this work, we propose to combine transfer learning with binary associative memories to achieve fully incremental learning. Binary associative memories are devices that are able to perform one-shot learning with very limited resources. The output of the DNN is quantized in a specific manner to combine it with the binary associative memories. This solution allows processing data sequentially one subset at a time, without forgetting initially processed data, and using only a sample of the database for learning. An overview of the proposed method is depicted in Figure 1. We evaluate the proposed method on challenging vision datasets (ImageNet and CIFAR10), and compare both accuracy and resources with alternative methods.

978-1-5386-0446-5/17/$31.00

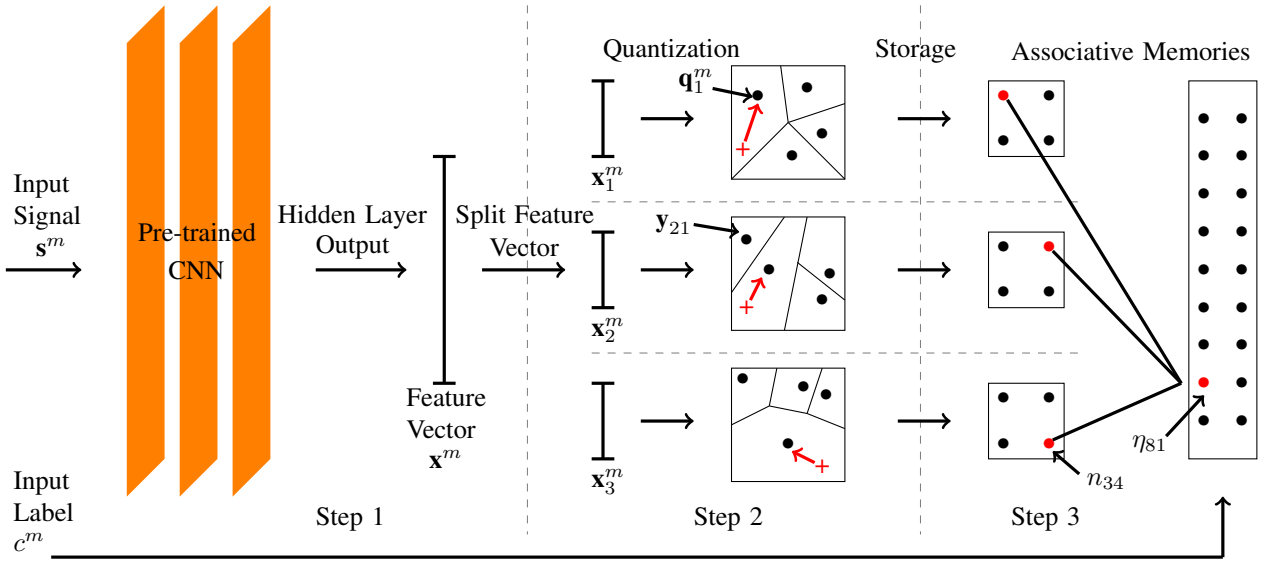

Figure 1. Overview of the proposed method, comprising three main steps. Given a set of samples, we first use a pre-trained CNN for feature extraction. Subsequently, we use a PQ technique to quantize the feature vectors. Finaly, we use a binary associative memory to store and classify the quantized data

The outline of the paper is as follows. In Section II we introduce related work. We present the proposed method in Section III. The experimental results are outlined in Section IV. Finally, Section V is a conclusion.

II. Related work

The term *incremental* usually refers to the ability of a learning process to learn sequentially, thus being able to handle new data and new classes without the need to retrain the whole system [9]. As an example, the "learn++" algorithm introduced in [9] accommodates new classes using weak one-vs-all classifiers. This approach conveniently manages the insertion, deletion and recurrence of classes over learning data [12]. However, this method requires to continuously train new classifiers in order to accommodate for new data, resulting in a potentially large computational intensiveness and memory usage.

Another approach was proposed to deal with a large amount of data [10], [11]. The idea is to replace batches in classical learning methods with a process based on Support Vector Machines (SVM), in which learning is performed using only one subset at a time, independently of the others. As a consequence, it is possible to limit memory usage. However, training the SVMs can be computationally expensive.

In [14] incremental learning refers to three distinct problems: example-incremental learning [10], [11], class-incremental learning [9], [12], and attribute-incremental learning. In [15], the authors propose a SVM inspired method to handle both the first two concepts defined above. However, it requires the training of novel SVMs using new examples and old SVMs. In addition, SVMs suffer from *catastrophic forgetting*, which is the loss of previously learned information [16], [17]. To address this problem a combination between SVMs and learn++ method called "SVMlearn++" [18] was proposed, showing a promising improvement on biological datasets [19]. However, this method still needs to retrain a new SVM each time new data is processed, and some knowledge is forgotten while new information is being learned.

The method we propose in this paper is quite different as it combines incremental aspects with one-shot learning using binary associative memories. Consequently, there is no need to retrain the system with old data, nor to perform computationally intensive processing with a new one. In addition, learning new data does not damage previously learned information, and only a few examples are required for learning, resulting in substantial savings in memory during the learning process.

III. The Proposed Method

The proposed method is built upon three main ideas: 1) using a pre-trained deep CNN to perform features extraction of signals, 2) using product quantizing techniques to embed data in a finite alphabet and 3) using binary associative memories to store and classify data as a proxy to a nearest neighbour search. In the next paragraphs, we detail these three steps.

Step 1: The first key idea is to use the internal layers of a pre-trained deep CNN [3] which acts as a generic feature extractor and associates an input signal $\mathbf{s}^m$ with a feature vector $\mathbf{x}^m$ (cf. Figure 1 step 1).

Step 2: Having a feature vector set $X = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^M\}$, the next step is to embed $\mathbf{x}^m$ in a finite alphabet. This step is crucial as it allows to map outputs of step 1) to the inputs of step 3). There is a lot of literature dedicated to this problem, including methods relying on Product Quantization (PQ) [20]. Because we aim at providing computationally light solutions, we rather use product random sampling in this work. Basically, we split each $\mathbf{x}^m$ into P subvectors of equal sizes

denoted $\left(\mathbf{x}_p^m\right)_{1\leq p\leq P}$, which are quantized independently from each other using random selection of K anchor points $Y_p = \mathbf{y}_{p1}, ..., \mathbf{y}_{pK}$, where each $\mathbf{y}_p^k$ is such that $\exists \mathbf{x}^m \in X, \mathbf{x}_p^m = \mathbf{y}_p^k$.

After step 2), each feature vector $\mathbf{x}^m$ is transformed into a word of fixed length $\left(\mathbf{q}_p^m\right)_{1\leq p\leq P}$ over a finite alphabet (the alphabet of the anchor points) (cf. Figure 1 step 2). This process is performed as follows:

$$\left\{ \begin{array}{lcl} k^\star(m,p) & = & \arg\min_k \|\mathbf{x}_p^m - \mathbf{y}_{pk}\|_2 \\ \mathbf{q}_p^m & = & y_{pk^\star(m,p)} \end{array} \right. . \quad (1)$$

Step 3: The outputs of product random sampling are words of fixed length P over a finite alphabet (containing K distinct symbols). The idea here is to use a neural network comprising two layers, the input one that is organized in P clusters containing K neurons each, and the output one containing RC neurons, where R is the number of neurons for each class and C is the number of classes in our dataset. Consider the neurons in the input layer to be indexed by two variables $p, 1 \leq p \leq P$ and $k, 1 \leq k \leq K$, where p denotes the index of the cluster and k the index of the neuron inside the cluster, and the neurons in the output layer to be indexed by two variables $c, 1 \leq c \leq C$ and $r, 1 \leq r \leq R$. We denote neurons in the input layer n_{pk} and neurons in the output layer η_{cr}.

When processing a training input signal $\mathbf{s}^m$, a number of neurons are activated in the network. Namely, we activate the neurons in the input layer whose indexes p, k are corresponding to the indexes of the activated anchor subvectors $\mathbf{q}_p^m$ obtained in Step 2. We activate in the output layer a neuron whose first index c is the index of the class c^m the training vector is part of, the second index r being drawn uniformly at random. Then we add connections (since the network is binary, there is no connection weight but only presence or absence of connections) between η_{cr} and all n_{pk}, printing a bipartite clique into the network (note that if a connection already existed, it is left unchanged) (c.f. Figure 1, Step 3).

We do the same process for every new data allowing incremental learning. Our method is a combination of a deep pre-trained CNN that does not change during the training process, and associative memories that are modified after each newly observed example or class. This combination allows to handle both example and class incremental approaches with no other prior about the learning dataset, using only few learning examples and without having to retrain the model or damage the previously obtained knowledge [21]. The overall process is depicted in Figure 1.

IV. Experiments

In this section we first describe the implementation details and strategies followed to quantize feature vectors $\mathbf{X}$. The accuracy of each strategy is then presented and discussed. We also investigate the behaviour of the accuracy, when changing some parameters, as the number of parts P to split feature vectors, as the number of neurons $\tilde{k}$ for each class in the input layer, and as the number of neurons R for each class in the output layer.

A. Implementation Details and Strategies Followed

For our method, we adopt the Inception V3 CNN model. It takes as input an image which is resized to $299 \cdot 299$ pixels and outputs a 2048 dimensional vector from the layer before the first fully-connected one [22]. The output vector represents the feature vector of the input image. To get the subvectors $\mathbf{y}_{pk}$ introduced in the previous section, we split the feature vectors $\mathbf{X}$ into P parts and we choose randomly $\tilde{k}$ subvectors from each class for each part (*i.e.* $K = C\tilde{k}$, C is the number of classes in our dataset and K is the total number of neurons in each cluster of the input layer).

We compare three strategies to emphasize the interest of the proposed method. They are described in the following paragraphs.

1) The "Independent Incremental" Approach (I-I): In this approach, training is not necessary: from each class we sample a portion of the example vectors and directly associate them to the corresponding output neurons using the associative memory. New data does not impact previously acquired knowledge, avoiding catastrophic forgetting.

In the case where $R = 1$, note that the associative memory is equivalent to counting how many quantized subvectors belong to each class and selecting the maximal one.

2) The "Non-Independent Incremental" Approach (N-I): In this approach, learning new elements can affect previously learned data. More precisely, each new input vector is quantized using all the already acquired anchor subvectors, independently of the class of the example that added them. The learning procedure is therefore computationally more complex than for the I-I method.

3) The "Non-Independent Offline" Approach (N-O): In this approach, the selection of anchor vectors is performed prior to any storing in the associative memory, so that the latter becomes independent of the order on which examples and classes are presented to the network.

B. Evaluation

We evaluate the proposed methods using three distinct datasets. The two first datasets (called in this paper ImageNet1 and ImageNet2) use 10 different classes of imageNet which were not used to train the CNN model. We use Cifar10 as the third dataset. Throughout the experimental part, the given accuracy is the average one over 10 realisations of each experiment.

1) Comparing the approches: Our first experiments consist of stressing the effect of the number of neurons per class R in the output layer of the associative memory on the accuracy of the three proposed approaches, for fixed values of the quantization parameters P and $\tilde{k}$. Figure 2 depicts the evolution of the accuracy of the methods as a function of the number of R. Expectedly, we observe that performance increases as a function of the number of neurons in the output layer. More interesting is the behaviour of the I-I method, which seems almost independent on R, while staying very close to the N-O method even when the latter is using a large value of R.

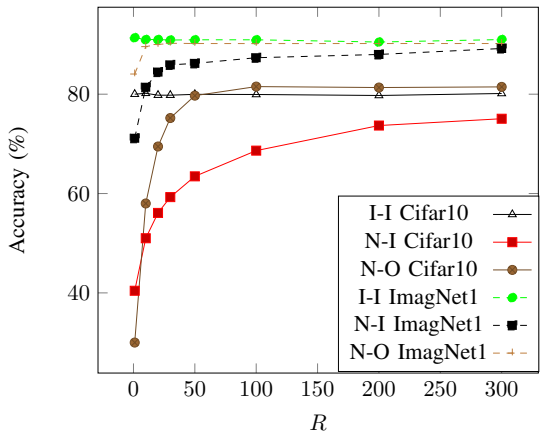

Figure 2. Evolution of the accuracy of the I-I approch ($P = 16$ and $\tilde{k} = 20$) as a function of the number of neurons R in the output layer for each class (ImageNet1 and Cifar10).

Additionally, the I-I approach shows better accuracy than the N-I one even when varying the parameters P and $\tilde{k}$, as shown in Table I. With this in mind, we focus on the I-I approach with $R = 1$ in the following experiments.

Table I
COMPARING THE ACCURACY OF I-I APPROACH WITH N-I APPROACH WITH $R = 300$ AND VARIOUS CLUSTER PARAMETERS (P AND $\tilde{k}$) (CIFAR10).

	I-I accuracy (%)	N-I accuracy (%)
$P = 16, \tilde{k} = 20$	**80.14**	75.06
$P = 16, \tilde{k} = 10$	**77.72**	65.52
$P = 64, \tilde{k} = 15$	**80.09**	75.61
$P = 32, \tilde{k} = 5$	**76.36**	59.38

Next, we consider two incremental protocols: class-incremental and example-incremental.

2) Class-Incremental Protocol: We first evaluate the effect of adding a new class on the accuracy. To do so, we start with an empty quantizer and an empty associative memory, and we add classes one by one. In order to avoid arbitrary decisions in the order in which classes are presented to the method, we perform experiments with random shuffles (200 times) and plot the average. We consider the following parameters: $P = 16$, $\tilde{k} = 20$ and $R = 1$. The accuracy a_c when introducing a novel class C_c is computed from scratch by adding new test examples to old one, according to Equation (2), where z_c represents the number of well classified test examples of all classes (for C_1 to C_c), m_c is the number of test examples of the class C_c and M_c is the total number of all test examples from classes C_1 to C_c.

$$\begin{cases} M_c = M_{c-1} + m_c \\ a_c = \frac{z_c}{M_c} \\ \text{with } M_0 = 0 \end{cases} \quad (2)$$

Each time a novel class is to be learned, we randomly sample $\tilde{k}$ subvectors for each of the P subspaces. We jointly add $\tilde{k}$ corresponding neurons in each cluster of the input layer. Finally, we add a new neuron corresponding to the newly added class in the output layer.

The obtained results are depicted in Figure 3. Of course the effect of adding new classes is more significant for a few number of classes, as it considerably strengthens the problem. The accuracy obtained after training all 10 classes approaches the one corresponding to the N-O approach for each dataset.

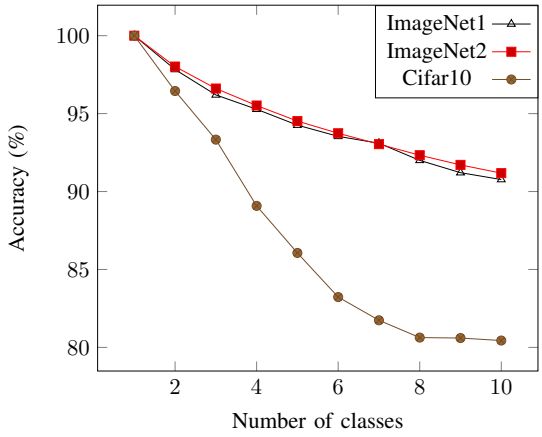

Figure 3. Evolution of the accuracy of the proposed method as a function of number of classes for $P = 16$, $\tilde{k} = 20$ and $R = 1$ (ImageNet1, ImageNet2 and Cifar10).

3) Example-Incremental Protocol: Next we evaluate the effect of adding new learning examples, without introducing new classes, on the accuracy of the I-I approach. To do so, we split the learning database into 5 parts and we learn one part at a time. The same testing dataset is used to measure accuracy at each step. For each part to be learned, we proportionally sample subvectors for each of the P subspaces and add the corresponding neurons in each cluster of the input layer. In Figure 4, for the three datasets, our method handles the incremental learning improving its accuracy and reaching the same final results as in the previous tests.

C. Complexity and memory usage

A key factor in proposing interesting solutions when targeting embedded architectures are complexity and memory usage. We refer to complexity as the number of arithmetical operations needed to learn the database (complexity-ℓ) or to classify an unlabelled input (complexity-p). The complexity-ℓ of the proposed method is negligible because we do not have to train the model with the whole dataset (c.f. subsection IV-D), while complexity-p is defined as $TK + PRC$ where T is the feature vector size ($T = 2048$ in our case due to the use of inception v3).

Figure 5 represents the accuracy as a function of the complexity-p when varying K and P (the other parameters are fixed to $T = 2048$, $R = 1$ and $C = 10$) and shows the best accuracy-complexity ratio (BACR). For a given K, BACR is obtained as the maximum in accuracy for similar values of complexity-p. We use the set of parameters reaching

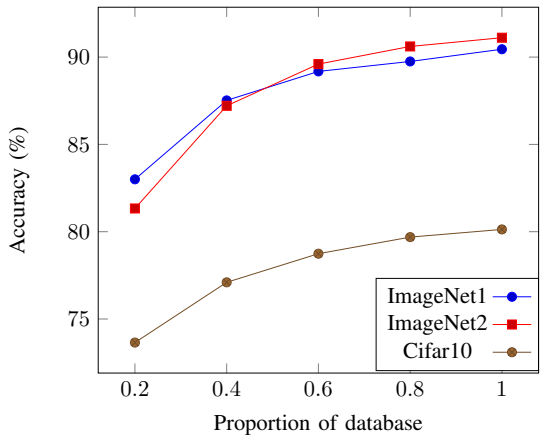

Figure 4. Evolution of the accuracy as a function of the number of learning examples ($P = 16$ and $\tilde{k} = 20$) (ImageNet1, ImageNet2 and Cifar10).

the BACR to compare our method with a nearest neighbour search.

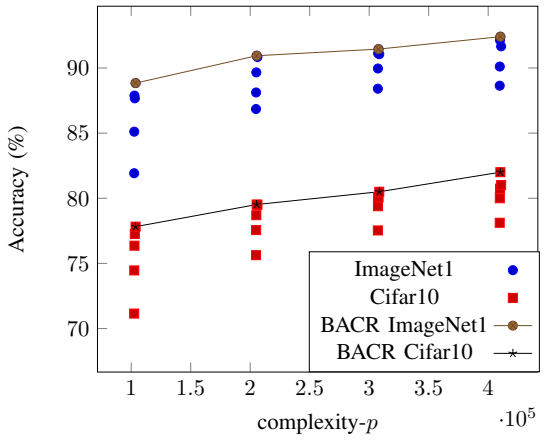

Figure 5. Evolution of the accuracy of I-I approach as a function of complexity-p (ImageNet1 and Cifar10) when varying K and P ($T = 2048$, $R = 1$ and $C = 10$).

Memory usage is defined by the size of clusters on input or output layers, and of the binary matrix which stores the connections between neurons. Memory usage sums up to $KTf + KPRC$ where f is the number of bits used per vector coordinates (we use $f = 32$ bits). Note that the size of the pre-trained CNN is not considered and the memory usage for learning and classifying are considered similar. We estimate accuracy, complexity and memory usage to compare the proposed method with a λ nearest neighbour (λ-NN) approach. The complexity-ℓ of the λ-NN search is also negligible, the complexity-p is defined by MT and memory usage is MTf. Results are shown in Table II.

We observe a loss in accuracy using our method, from 87% for nearest neighbour to 82%. On the other hand we obtain important gains in complexity and memory usage. One of the reason is that nearest neighbour search requires storing all

Table II
ACCURACY, COMPLEXITY AND MEMORY USAGE OF I-I APPROACH ($P = 64$, $K = 200$ AND $R = 1$) COMPARED TO λ-NN SEARCH FOR CIFAR10.

	Proposed Method	Other techniques	
		1-NN	5-NN
Accuracy(%)	82	85	**87**
complexity-ℓ	**negligible**	**negligible**	**negligible**
complexity-p	$\mathbf{4.1 \cdot 10^5}$	10^8	10^8
Memory usage-ℓ	$\mathbf{1.3 \cdot 10^7}$	$3.3 \cdot 10^9$	$3.3 \cdot 10^9$
Memory usage-p	$\mathbf{1.3 \cdot 10^7}$	$3.3 \cdot 10^9$	$3.3 \cdot 10^9$

training examples, which does not meet the criteria that define incremental learning algorithms [9].

Instead of using λ-NN search, we can accelerate it using PQ (Product Quantification). Namely, we split all feature vectors $\mathbf{x}^m$ into P of equal size denoted $\left(\mathbf{x}_p^m\right)_{1 \leq p \leq P}$, and for each subspace, we perform K-means on the feature vector set $X_p = \{\mathbf{x}_p^1, ..., \mathbf{x}_p^M\}$ to extract K centroids. When using K-means, we lowerbound the complexity-ℓ by taking into account only the MTK operations needed to quantize the learning dataset before storing it, with no consideration for the price of performing K-means. We motivate this choice as one could instead use product random sampling as described in our method. The complexity-p is $TK + MP$ and the memory usage is $TfK + MPlog_2(K)$. Table III shows the obtained results.

Table III
ACCURACY, COMPLEXITY AND MEMORY USAGE RATIO OF I-I APPROACH ($P = 64$, $K = 200$ AND $R = 1$) COMPARED TO λ-NN SEARCH USING PQ ($K = 200$, $P = 64$) FOR CIFAR10. NUMBERS BETWEEN BRACKETS ACCOUNTS FOR PRODUCT RANDOM SAMPLING INSTEAD OF PQ.

	Proposed method	Other techniques	
		1-NN	5-NN
Accuracy(%)	82	82.6(82)	**86.07**(83)
complexity-ℓ	**negligible**	$\geq 2 \cdot 10^{10}$	$\geq 2 \cdot 10^{10}$
complexity-p	$\mathbf{4.1 \cdot 10^5}$	$3.2 \cdot 10^6$	$3.2 \cdot 10^6$
Memory usage-ℓ	$\mathbf{1.3 \cdot 10^7}$	$3.7 \cdot 10^7$	$3.7 \cdot 10^7$
Memory usage-p	$\mathbf{1.3 \cdot 10^7}$	$3.7 \cdot 10^7$	$3.7 \cdot 10^7$

NN search using PQ not only gives a good accuracy (86.07% compared with 82% of the I-I approach), but also reduces the complexity-p and memory usage by a factor of 100. However it requires a large computational power for learning process and stores a quantized version of the whole dataset, again not complying with the incremental learning algorithms criteria [9]. In addition both complexity and memory usage depends of number of learning examples M and could quickly become problematic.

Note that the proposed method uses product random sampling because it offers almost the same accuracy as using K-means instead. Moreover to use K-means it is required to store the whole database and perform expensive operations. The same input data (i.e. the K feature vectors) are used to

train an offline SVM and a Multilayer Perceptron (MLP) with one hidden layer. As a result, the obtained accuracy is 82% and 83% respectively.

Finally, to assess the robustness of the proposed method with regards to the chosen CNN feature extractor, we perform similar experiments using the SqueezeNet [6] architecture. This network makes even more sense with regards to embedded platforms given its very small memory usage. Table IV shows the obtained results that comfort the ones obtained using Inception V3.

Table IV
ACCURACY, COMPLEXITY AND MEMORY USAGE RATIO OF I-I APPROACH ($P = 64, K = 200$ AND $R = 1$) USING SQUEEZENET COMPARED TO λ-NN SEARCH FOR IMAGENET2.

	Proposed method	Other techniques	
		1-NN	5-NN
Accuracy(%)	84	88	**89**
complexity-ℓ	**negligible**	**negligible**	**negligible**
complexity-p	$\mathbf{2 \cdot 10^5}$	$8.4 \cdot 10^5$	$8.4 \cdot 10^5$
Memory usage-ℓ	$\mathbf{6.5 \cdot 10^6}$	$3.2 \cdot 10^8$	$3.2 \cdot 10^8$
Memory usage-p	$\mathbf{6.5 \cdot 10^6}$	$3.2 \cdot 10^8$	$3.2 \cdot 10^8$

D. Discussion

The proposed method achieves incremental learning (Figures 3 and 4) with substantial reduction of computational complexity and memory usage, compared to nearest neighbour search, without compromising classification accuracy (Tables II and III). Note that we preferred the I-I approach in our tests, since it obtained better accuracy than N-I. Since we also use product random sampling to feed the associative memories and require only one neuron per class in the output layer, we obtain that the neuron n_{pk} corresponding to y_{pk} is only connected to the neuron η_c, where $\exists \mathbf{x}^m \in X, \mathbf{x}_p^m = \mathbf{y}_{pk}$ and x^m belongs to the class C_c. As a consequence, knowing the connections of neurons obtained from random sampling with the neurons of the output layer, we need only few examples to train our model. Thus, the proposed method needs only a portion of the learning dataset to train, resulting in even lighter computational intensiveness and memory usage.

V. CONCLUSION

We introduced a novel incremental algorithm based on pre-trained CNNs and associative memories to classify images, the first ones using connection weights to process images, the second one using existence of connections to store them efficiently. This combination of methods allows to learn and process data using very few examples, memory usage and computational intensiveness. The obtained accuracy is close to other state-of-the-art methods based on transfer learning. As a consequence, we believe this method is promising for embedded devices and consider proposing thrifty hardware implementations of it as future work.

REFERENCES

[1] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," *CoRR*, vol. abs/1502.06796, 2015. [Online]. Available: http://arxiv.org/abs/1502.06796

[2] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[3] 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.

[4] C. F. Cadieu, H. Hong, D. L. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. DiCarlo, "Deep neural networks rival the representation of primate it cortex for core visual object recognition," *PLoS Comput Biol*, vol. 10, no. 12, p. e1003963, 2014.

[5] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 26–35.

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[7] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and F.-F. Li, "Large scale visual recognition challenge," *www. image-net. org/challenges/LSVRC/2012*, vol. 1, 2012.

[8] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *International Conference on Machine Learning (ICML)*, June 2016.

[9] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, no. 4, pp. 497–508, 2001.

[10] N. A. Syed, S. Huan, L. Kah, and K. Sung, "Incremental learning with support vector machines," 1999.

[11] T. Poggio and G. Cauwenberghs, "Incremental and decremental support vector machine learning," *Advances in neural information processing systems*, vol. 13, p. 409, 2001.

[12] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, "Online ensemble learning of data streams with gradually evolved classes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1532–1545, 2016.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[14] Z.-H. Zhou and Z.-Q. Chen, "Hybrid decision tree," *Knowledge-based systems*, vol. 15, no. 8, pp. 515–528, 2002.

[15] J. Zheng, F. Shen, H. Fan, and J. Zhao, "An online incremental learning support vector machine for large-scale data," *Neural Computing and Applications*, vol. 22, no. 5, pp. 1023–1035, 2013.

[16] N. Kasabov, *Evolving connectionist systems: Methods and applications in bioinformatics, brain study and intelligent machines*. Springer Science & Business Media, 2013.

[17] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[18] Z. Erdem, R. Polikar, F. Gurgen, and N. Yumusak, "Ensemble of svms for incremental learning," in *International Workshop on Multiple Classifier Systems*. Springer, 2005, pp. 246–256.

[19] J. F. G. Molina, L. Zheng, M. Sertdemir, D. J. Dinter, S. Schönberg, and M. Rädle, "Incremental learning with svm for multimodal classification of prostatic adenocarcinoma," *PloS one*, vol. 9, no. 4, p. e93600, 2014.

[20] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[21] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv preprint arXiv:1512.00567*, 2015.