Dual-Memory Model for Incremental Learning: The Handwriting Recognition Use Case

Conference Paper · January 2021

DOI: 10.1109/ICPR48806.2021.9411977

5 authors, including:

Melanie Piot
ESIEA University
6 PUBLICATIONS **9** CITATIONS
SEE PROFILE

Jordan Gonzalez
7 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Aurelia Deshayes
Paris-Est Créteil University
12 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Lionel Prevost
ESIEA University
90 PUBLICATIONS **1,160** CITATIONS
SEE PROFILE

Dual-Memory Model for Incremental Learning: The Handwriting Recognition Use Case

1st Melanie Piot
Learning, Data and Robotics Lab
ESIEA
Paris, France
mpiot@et.esiea.fr

2nd Berangere Bourdoulous
Learning, Data and Robotics Lab
ESIEA
Paris, France
bourdoulous@et.esiea.fr

3rd Jordan Gonzalez
Learning, Data and Robotics Lab
ESIEA
Paris, France
gonzalez@esiea.fr

4th Aurelia Deshayes
Learning, Data and Robotics Lab
ESIEA
Paris, France
deshayes@esiea.fr

5th Lionel Prevost
Learning, Data and Robotics Lab
ESIEA
Paris, France
prevost@esiea.fr

***Abstract*—In this paper, we propose a dual memory model inspired by psychological theory. Short-term memory processes the data stream before integrating them into long-term memory, which generalizes. The use case is learning the ability to recognize handwriting. This begins with the learning of prototypical letters . It continues throughout life and gives the individual the ability to recognize increasingly varied handwriting. This second task is achieved by incrementally training our dual-memory model. We used a convolution network for encoding and random forests as the memory model. Indeed, the latter have the advantage of being easily enhanced to integrate new data and new classes. Performances on the MNIST database are very encouraging since they exceed 95% and the complexity of the model remains reasonable.**

***Index Terms*—Incremental learning, random forest, memory, handwriting recognition**

I. Introduction

The past decade has been marked by the incredible advances of Artificial Intelligence and its growing presence in ever more varied fields (autonomous driving, medical diagnosis, emotional recognition, etc.). These advances have been made possible thanks to the advances of Deep Learning,particularly End-to-End Learning and Representation Learning. Although this technique has proved its worth on problems that have been difficult to solve until now (particularly in the field of image analysis), it has two major drawbacks:

- Current models don't easily adapt to new data they haven't been trained on;
- Their training only converges when fed with huge labeled databases.

In the presence of new data or classes, we observe the so-called "catastrophic forgetting": the model focuses on these new data and "forgets" the older ones. To avoid or limit this phenomenon, many algorithms have been proposed that we will group together under the term incremental learning. The latter tends to move away from machine learning and towards human learning, in order to reproduce our ability to learn throughout our lives, with little data. These algorithms include the following:

- Lifelong Learning whose objective is to learn continuously new tasks using the knowledge accumulated so far from the old tasks, without having had access to the new tasks while learning the old ones.
- Multi-Task Learning which performs learning of several tasks simultaneously. The learning system can optimize its knowledge on each of the tasks together, as they would share a common knowledge base. However, this implies that all the data needed to learn all the tasks (which must be relatively similar) is immediately available.
- Tranfer Learning deals with multi-tasking sequentially. The tasks to be learned are divided into source and target tasks. The system is therefore trained on all source tasks in order to establish the "common knowledge base" mentioned above. Once this knowledge base is gathered, it is used to solve the target task. Note that the objective of knowledge transfer is to learn how to correctly solve the target task only, so the knowledge extracted from the source tasks may be completely irrelevant.
- Curriculum Learning aims to optimize the learning of a model by presenting the tasks to be solved from the simplest to the most complex. This aims to optimize the changes made by the model during its learning, and consequently, to accelerate its learning. However, it is difficult to establish the curriculum necessary for optimal learning, and the interpretation of the results remains very difficult.

In this study, we propose an original experiment in the intersection of Lifetime Learning and Curriculum Learning. To get closer to human learning, we learn a first supposedly simple task: the recognition of printed digits. Then, we learn incrementally the second task: the recognition of handwritten digits. This second task is far more complex than the

first because of the intrinsic variability of handwriting. The figure 1 shows the difference in complexity between these two tasks. This sequence of learning is reminiscent of the experience of the child in the kindergarten section. First of all, the child learns the digits that have been perfectly written (digit "models") by his teacher. Then, he will generalize this experience throughout his life on handwritten writings that are increasingly varied and complex. The model we propose is therefore incremental in data since the classes remain identical for both tasks.

Fig. 1. Printed *vs handwritten digits.*

II. State of The Art

A. Biological model

Human biological memory is complex and still partially unknown to this day. Although much research has highlighted the existence of different mechanisms necessary for the proper integration of each of the information captured by our senses, the organization of all these mechanisms is still unclear. It has therefore been necessary to make choices among the various existing compositions and organizations of memory, as well as some interpretations of their internal communication. In this study, we choose Baddeley's model [1], known for its representation of short-term memory (STM) in the form of two mechanisms. The first mechanism, the so-called STM buffer, stores new experiences. The second, called working memory (WM), combines/integrates new experiences with those already encountered and stored in the long-term memory (LTM). The STM buffer is, according to Baddeley, a sub-section of the STM and acts as a temporary storage area for information perceived by all our senses, waiting to be processed and integrated by our brain. However, its temporality is ten seconds and does not exceed one minute, [2], [3]. The buffer therefore behaves like a FIFO (First In First Out) list according to the time interval between the arrival of the information and its recall. Moreover, the number of information that can be stored there is very small (of the order of 7+/-2) [4]. The WM is the second part of the STM. It plays a critical role in the integration of information into the LTM. It is composed of the following four mechanisms:

- A phonological loop (or subvocalisation): involving the conversion of information into auditory form (mainly to keep it in the STM buffer);
- A visuo-spatial memory: retrieving information in visual form;
- The episodic buffer: gathering the information from the two previous codings and the information from the LTM (retrieved by the central administrator);
- The central administrator: aiming to combine all the information contained in the episodic buffer via an attentional mechanism.

The LTM, on the other hand, is divided into two subparts: explicit memory (ELTM) and implicit memory (ILTM). Any learning that can be consciously recalled would be stored in the ELTM. This includes not only lived experiences (memories), but also language and encyclopedic concepts. On the other hand, the ILTM would contain knowledge acquired over time without the need for conscious activation. It is mainly about automatic actions such as riding a bicycle or playing an instrument, but also about principles such as conditioning and autosuggestion. Contrary to other models such as Cowan's model, which is more complex and suggestive, Baddeley's model proposes a more sequential vision of the functioning of memory, which allows us to attempt an algorithmic equivalence with more ease.

B. Algorithmic solutions

In this section, we attempt to categorize incremental learning methods that can mitigate, to varying degrees, catastrophic forgetting. It should be noted that most of these methods are developed to address image analysis problems and therefore implement deep convolutional networks. We encourage the reader who wishes to explore these questions in greater depth to consult the very detailed state of the art proposed by [5].

The regularization methods [6]–[8] continue the learning of networks by imposing additional constraints on weight updates. These approaches are based on theoretical models in neural sciences that suggest that consolidated knowledge can be protected from catastrophic forgetting by synapses with a cascade of states producing various levels of plasticity. These approaches are interesting for incremental learning provided that data are not too different from one task to another or that they are presented to the network with some of the previous tasks. In addition, learning incrementally new classes is not possible.

The dynamic methods [9]–[11] modify the network architecture, by adding neurons or layers, dynamically to integrate the new data. This approach solves the problem of incrementally learning new classes that previous approaches faced, while preserving a good consolidation of knowledge. However, it has huge constraints related to available computing power and storage space.

The last class of methods is based on the theory of the complementary learning system which combines short-term and long-term memory [12]–[14]. The former is responsible for learning about single experiences, while the latter aims to learn to generalize about a set of experiences. These models seem to be very effective in consolidating acquired knowledge in addition to countering catastrophic forgetting. Their main drawback is the need to retain data as learning takes place, resulting in an explosion of the storage space required.

TABLE I
ALGORITHMIC TRANSPOSITION OF THE BADDELEY'S MODEL

Memory	Biological Mechanism	Algorithmic Equivalent
STM	STM Buffer	FIFO list of size KxN
	Visuo-spatial sketchpad	CNN
	Phonological Loop	Labels
	Episodic Buffer	Improvement
	Central Administrator	Consolidation
LTM	Explicit Memory	Data Distribution
	Implicit Memory	Random Forest

III. COMPUTATIONAL MODEL

Table I details how we propose to algorithmically transpose Baddeley's model. All the elements of the memory are detailed in the following subsections.

A. Learning the first task

Let's study the case of a 6-year-old child who is learning to distinguish digits, in this case printed digits. At this age, the brain has gained enough experience to no longer have to learn to break down what the eyes perceive; this mechanism happens automatically. We propose to transpose this phenomenon as follows.

1) Visuo-spatial sketchpad generation: In order to generate this sketchpad, an obvious solution is to use the output of the convolution layers of a network trained to recognize printed digits. To do this, we first downloaded all the fonts from Google, then sorted them in order to keep only the most sober ones while ensuring great diversity (removing 0's with a slash or a dot in the middle, or having too small font, etc.), e.g. 2.560 different fonts. We have thus generated a balanced base of 76.800 images of black digits on a white background in 28x28 format. This last was divided into two sets: 80% for training and 20% for testing. In a second step, we quickly optimized the CNN architecture: two alternating convolution and pooling layers producing a vector of 256 features, followed by a dense layer (16-sized) and a decision layer (10 outputs). Accuracy of this network on the test data are as follows: 0.985 on the printed data and 0.768 on the handwritten data (test set MNIST). This shows us that the features and rules built to recognize printed digits can be used for handwritten digit recognition but are not optimal.

2) LTM initialization:

a) Implicit Memory: Random forests minimize both bias (deep trees) and variance (majority vote). They have other advantages: very quick learning, possible incrementality in either data and classes, and better interpretability. Moreover, unlike neural networks, their training is not based on the minimization of a loss function. They are therefore not subject to catastrophic forgetting. For that reason, they will be used to model short and long term memories. The initialization of the LTM consists in training a random forest on the printed data we have collected, using the features produced by the CNN. Once this forest is optimized (100 trees with a maximum depth of 22, optimization of nodes in random 16-dimensional subspaces), the performances on the test data are as follows: 0.986 on printed data and 0.742 on handwritten data (test set MNIST). The results are quite similar to the previous ones, which justifies the implementation of an incremental learning of the handwritten data. This first forest therefore constitutes the implicit part of the LTM.

b) Explicit Memory: Once the initial training has been performed using our database of 77k images of printed digits, it is no longer possible to access these data. However, in order to avoid catastrophic forgetting, we cannot totally discard the old data. Indeed, incremental strategies need printed data in addition to handwritten data to be applied. Therefore, the distribution of the 256 featres is stored, in order to be used to regenerate data when needed. To do so, the 256 variables are decorrelated by principal component analysis. Then, they are approximated by a class-conditionnal Gaussian Mixture Model. Since the variables are decorrelated, it is sufficient to estimate a mean and a diagonal covariance matrix for each distribution. This limits the number of parameters to be estimated and the storage space required. These distributions constitute the explicit part of the LTM.

B. Incremental Learning

This process takes place in the working memory (WM). The concept of working memory is sometimes used as a synonym for short-term memory (STM), but it refers to a system that not only temporarily holds information (about 7 for the human brain), as short-term memory does, but also processes and manipulates it. In particular, WM is responsible for consolidation of new knowledge before it is incorporated into long-term memory. At initialization - and after each consolidation - the STM receives a copy of the LTM.

1) Short-Term Memory: Short-term memory is algorithmically modeled by a FIFO queue that can hold up to $K \times N$ data from the MNIST train set. N is the number of new data items that we will simultaneously present to our algorithm. K defines the number of enhancements, i.e. the number of times that N data will be presented before the consolidation step in the LTM.

2) Working Memory:

a) Visuo-Spatial Sketchpad: As above, the CNN feature extraction layers are used to encode the handwritten digit images.

b) Phonological Loop: This phenomenon, also known as subvocalization, corresponds to the third path of the three-way reading model [15]. The reader subvocalizes difficult words while reading them by converting graphemes into phonemes. Likewise, the child tries to read aloud the digit that is difficult to recognize. We transpose this phenomenon by making the following strong hypothesis: each data is presented with its label and thus the incremental learning is done in a supervised way.

c) Episodic Buffer: As explained earlier, once incremental learning begins, the first task's training data are no longer available. As the forest forming working memory (WM) learns to classify handwritten digits correctly, it must necessarily be modified. We used the incremental strategies proposed by [16] to update this forest:

- ULS (*Update Leaf Statistics*) : updates the histogram of classes in the leaves nodes receiving data. The incremental addition of data allows to update the leaf decision threshold. It ensures better recognition of handwritten digits. sed alone,this strategy helps to improve handwritten digit recognition but may potentially decrease performances on printed digits. To prevent this kind of catastrophic forgetting it is necessary to apply the following strategy.
- IGT (*Incrementally Grow Tree*) : extension of trees by replacing a leaf by a node, according to an impurity threshold (Gini criterion) F . As, for standard training, the task is then to find the best feature/threshold pair in terms of purity. Here, we chose to test the whole feature set since only few data are used at each iteration. Not performing feature bagging removes a random source, thus allowing a better analysis of the algorithm behavior.

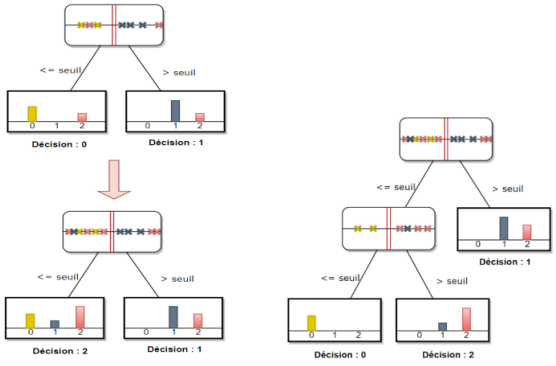

Fig. 2. ULS and IGT incremental learning strategies

These two strategies are described in figure 2. The ULS strategy (on the left) only changes the leaf class histogram. Beyond a certain impurity threshold, the IGT strategy (on the right) creates a new node, so that new leaves are purer.

In order to avoid unnecessary tree extension, for both strategies, we only consider incremental data that have been misclassified by the WM (among the N presented). However, for effective use of these strategies, it is necessary to weight these data and give them a real impact when faced with class histogram with high bins in certain leaves. This weighting is conditional to classes and decreases according to the number of incremental data presented. This avoids both catastrophic forgetting and excessive tree growth.

This is where explicit memory (ELTM) comes in. The ULS and IGT strategies, as presented by [16], assume the storage of the training data in the nodes and thus the access to the all the training data at all times. This is in contradiction with our initial hypothesis limiting data storage. Therefore, the class histogram is stored alone in the leaves. Explicit long-term memory is used during the incremental learning phase. When needed, data are (re)-generated using class-conditional distributions. These are representative to those in the initial training phase. Then, these data (close to initial ones) and weighted misclassified incremental data contained in the STM are merged to train the new node.

Finally, for each k enhancement, the corresponding WM is stored in a list. All the working memories $\{WM_1, ..., WM_k\}$ are then evaluated on the data of the kth short-term memory, containing at most $k \times N$ data or precisely $k \times N$ data if no consolidation has been made at this stage. The best performing one is stored in the working memory and forms the initial forest for the next enhancement.

d) Central Executive: It is in charge of alternating the enhancement and consolidation phases. Let us take the example of the child again. As the day is progressing, new connections are created (enhancement) to deal with the experiences encountered. This corresponds to the tree modification operations (ULS and IGT strategies). At night, the child enters a phase called consolidation. During this phase (biologically corresponding to deep sleep), the brain "sorts" the information encountered during the day. It will strengthen certain connections and make others disappear. Too many disappearances would lead to catastrophic forgetting (biologically called retroactive inference). On the contrary, too few changes could prevent the assimilation of new experiences (proactive inference).

We tried to reproduce this principle by comparing the modified WM after K enhancements with the LTM obtained during the last consolidation (following the previous K enhancements or with the initial forest if it is the very first consolidation cycle). For this purpose, a mixed validation base has been created, composed of both printed and handwritten samples. The latter are obviously independent of the MNIST test set. The memory with the best accuracy on printed AND handwritten digits is therefore updated as LTM. At the next consolidation cycle (e.g. until next night), the LTM is confronted with a new WM having integrated $K \times N$ new samples (e.g. new experiences). It is this LTM which is in charge of processing the current data during the "day". Finally, [17] have shown that there is always an optimal subset of trees in a forest. We have therefore implemented a step of Sequential Backward Selection, in order to reproduce the phenomenon of connection destruction.

3) Full algorithm: The entire process is summarized in Algorithm 1. Let C be the total number of consolidations to be performed, K the number of enhancements between each consolidation and N the number of incremental data presented during each enhancement. RF is the random forest trained on the first task (no SBS applied) and D is the distribution of the training data (c.f. III-A2). Let $slot$ be the stream of incremental data added at each enhancement. This stream, refered as $IncrementalStream$, transmits MNIST training data in packets (e.g. $slots$) of size N. $ValSet$ is the validation set used in the consolidation phase. Note that $\{dataset\}.X$ and $\{dataset\}.Y$ are the images and their labels, for any dataset.

Algorithm 1: Incremental Learning steps

Input: N, K, C
Output: LTM trained on two tasks
Data: RF, D, $IncrementalStream$, ValSet
ILTM ←RF
ELTM ← D
LTM ← SBS(RF)
STM ← {}
for $c \leftarrow 1$ **to** C **do**
  // for each consolidation do
  wmList ← {}
  WM ← ILTM // with all trees
  wmList $push$ WM
  for $k \leftarrow 1$ **to** K **do**
    // for each enhancement do
    slot ← $read$($dataStream$) // read N MNIST samples
    if $size$(STM) = $K \times N$ **then**
      pop(STM)
    end
    STM $push$ slot
    y_pred = WM $predict$ STM.X
    mc ← $misClassified$(y_pred, STM) // mc contains values and labels of WM prediction errors
    weights ← $getWeights$(slot.Y) // get and update incremental classes weights
    WM ← uls_igt(WM, ELTM, mc, weights)
    wmList $push$ WM
    WM ← $argmax_{wm}[accuracy(wm, STM)]$
    // for wm in wmList
  end
  /* consolidation step */
  if *accuracy(SBS(WM), ValSet)* > *accuracy(LTM, ValSet)* **then**
    ILTM ←WM // with all trees
    LTM ← SBS(WM) // part of trees
  end
end

It should be noted that at the consolidation step a full Random Forest is saved in implicit LTM. This one is kept with all trees to initialize WM at next cycle. LTM contains optimized Random Forest at the end of consolidation. This contains the final model at the end of incremental learning algorithm.

IV. Experiments

This section describes the proposed pipeline for incremental learning of handwritten digits. We also evaluate the sample weighting method implemented to increase the impact of incremental data, the value of the refinement process and its influence on the model accuracy. Finally, we analyze the performance of the complete algorithm on task 2.

A. Parameter Sensitivity Analysis

1) Backward selection: Recall that the SBS algorithm is supposed to optimize a random forest in its recognition performance, by removing low-performing trees within the forest [17]. The characteristics of the forest before and after SBS are reported in experiment 1 of Table II. We can observe an improvement in handwritten recognition performance of nearly 1.4%, but more importantly, a sharp decrease in the size of the model.

2) Data weighting: In order to take full advantage of incremental data, it is essential to give them a significant weight. But this weight must also decrease as improvements are made (integration of poorly classified data), to avoid catastrophic forgetting. The class-conditional weighting function we have used is shown in Figure 3. In experiment 2 in Table II, we measure the impact of the initial weight ω_0. We report the performance obtained respectively without weighting the data and by weighting the data. Both experiments are performed for a single enhancement ($K = 1$) and a consolidation on a data sample of size $N = 100$. In both cases no catastrophic omission is observed. Moreover, an initialization of the weight ω_0 at 10 allows to multiply the improvement on task 2 by three. We can see that the average number of nodes created increases only slightly.

For each $y \in \{0, \ldots, 9\}$,

$$W(C_y) = \begin{cases} 10 - \frac{1}{1000} C_y & \text{if } C_y < 1000 \\ 13 - \frac{1}{250} C_y & \text{if } 1000 \leq C_y \leq 3000 \\ 1 & \text{if } C_y > 3000 \end{cases}$$

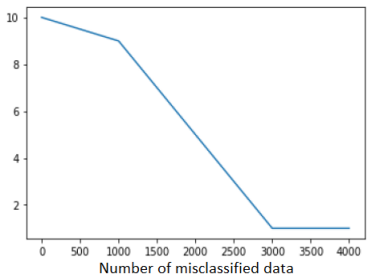

Fig. 3. Conditional weighting function.

TABLE II
SENSITIVITY TO PARAMETERS.

#EXP	K	N	ω_0	Acc. task 1 (%)	Acc. task 2 (%)	#node (%)	#tree
1				98.66	74.22		100
				98.65	**75.60**		**-14**
2	1	100	1	+0.03	+ 1.02	+11	
	1	100	10	=	**+ 3.11**	+13	
3	1	1000		=	+11.60	+120	
	2	500		+0.02	+10.27	**+33**	

3) Impact of enhancement: To validate the interest of the enhancement we compare (experiment 3 in Table II) the performances of the algorithm with and without the use of enhancement for the same number of incremental data. In a first step, without enhancement, we use a sample of size $N = 500$ on 2 consolidations. Then, we use the same number of data, on a single consolidation, but with 2 enhancements. Thus, in both cases, the incremental learning will have used 1000 data but in two different ways. We observe a very clear difference in the size of the final model. Admittedly, the gain in performance is slightly less in the presence of the enhancement. But this difference remains acceptable since we consider only a small sample of incremental data.

4) K and N Parameters: We've extracted a database of 15,000 handwritten data (a quarter of the MNIST training base) to optimize these parameters. We perform a grid search and compare the performance of the incremental learning for all (N, K, F) triples, where $N \in \{100, 500, 1500, 3000\}$, $K \in \{1, 2, 3, 5\}$ et $F \in \{0.1, 0.2, 0.3\}$. Since the number of experiments performed was too large for easy visualization, we report the main conclusions of the analysis below. First of all, there is clearly no single optimal value for K or N. The optimums of these values are related to the product of one with the other. On the other hand, we found that $K = 1$ (i.e., no enhancement) never works well. Moreover, we noticed that the greater N is, the greater the gain in handwritten performance following the first consolidation, and the less it is during subsequent consolidations. This observation is very logical. The closer N is to the size of the database, the more the learning looks like standard learning, and therefore the faster the results progress. Finally, surprisingly, the F factor (trigger point in the IGT strategy) does not seem to impact the performance of the model.

B. Performance on MNIST test database

In the following experiments, we incrementally train the model on the MNIST train set and report its performance on the whole test printed database generated for Task 1 and on the MNIST test set containing the handwritten data. This allows us to observe the performance of the incremental model on task 2, while measuring the possible impact of catastrophic forgetting.

Figure 4 details this behavior for four pairs (K, N). We observe the performances after each consolidation. As it can be seen, performances increase more or less quickly depending on the values of K and N. But all exceed 90%, beyond a tens of thousands of images. From this point on, the performances continue to improve, but much more slowly. We will try to explain to this behavior in the final discussion. We also check that the performances on task 1 do not decrease (or very little). It confirms that the proposed model succeeds in avoiding catastrophic forgetting.

We study deeply the evolution of the misclassification rate for these same pairs of parameters in figure 5. As before, this rate decreases quickly before becoming almost constant. The main difference is the non-monotonous nature of this evolution, which is easily explained. Indeed, we report here the rate of misclassification, on the STM buffer, after each enhancement. Increasing the size of the slots (N) logically tends to "smooth" this evolution.

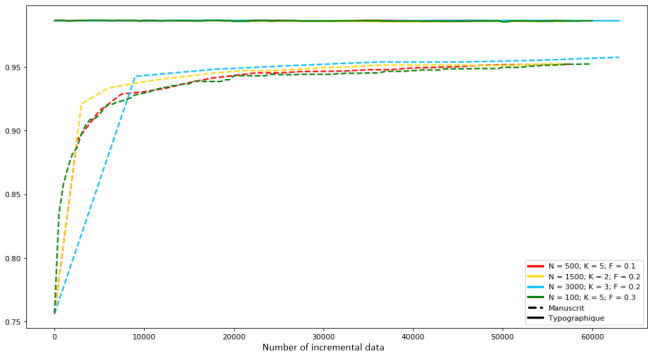

Fig. 4. Evolution of the recognition rate.

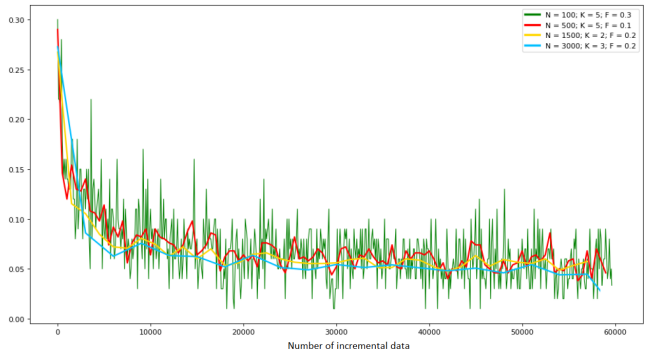

Fig. 5. Evolution of the misclassification rate.

Figure 6 shows that the model becomes more complex as it encounters incremental data. It results in an increasing number

of leaves per tree, a behavior due to the triggering of the IGT strategy (see section III-B2c). It can also be noted that the model refinement follows the behavior of the class-conditional weighting function. The higher the weight of the slot data, the more leaves are added. Thus, the forest increases by about $+400$ leaves on average per tree after the first data slot, while the weight of the data in this slot is more consequent. Then, the growth tends to be slower. The number of added leaves decreases, with about $+200$ leaves between two slots and so on, while the weight of the data of the last slots decreases. Nevertheless, a worrying phenomenon is observed: although performance doesn't progress beyond a certain number of slots (see above), the model continues to become more complex. We will try to explain this phenomenon in the next section.

Finally, when jointly analyzing the previous results (figure 7), it can be seen that the best performances (accuracy on both printed and handwritten digits AND "weights" of the model) are obtained for small STM buffers, which tends to bring us closer to biological functioning.

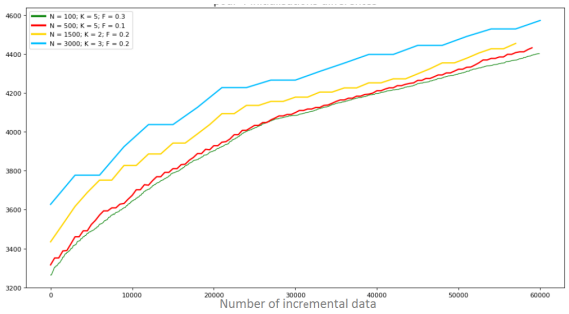

Fig. 6. Evolution of the mean number of leaves per tree.

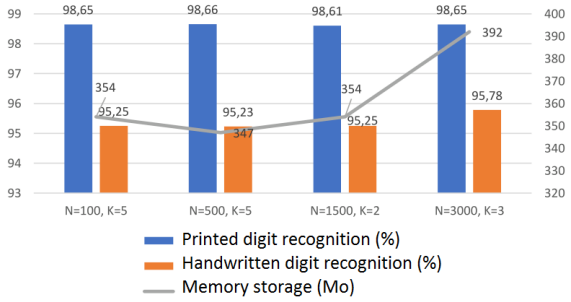

Fig. 7. Performance comparison.

V. Conclusion

In this paper, we propose a dual memory model inspired by the one proposed by Baddeley in neural science. We have tried to reproduce as closely as possible the different components of this model, which lends itself quite well to algorithmic transposition. The chosen use case is the learning of handwriting recognition. This starts with recognizing letter models. We have imitated this first task by training a model able to recognize printed digits. It continues throughout life and gives the individual the ability to recognize increasingly varied handwritten digits. This second task is achieved by incrementally training our dual-memory model. The performance, measured on the MNIST database is very encouraging, exceeding 95%. However, given the specificity of the experiment, it is difficult to compare these performances with those reported in the literature. Although these are sometimes higher than those of our model, the latter has the advantage of not requiring the storage of all the training data and is less computationally expensive.

We have found during experiments that performance tends to stagnate. This can easily be explained. As the CNN has been trained on the first task, the features produced are no longer suitable for the second task. One way is to fine-tune the networks. But the simplest solution is to use a pre-trained network (such as ResNet, InceptionV4 or UNet) that would extract features that are generic enough to suit any task. This solution would have the advantage of simplifying the issue of class incrementality. Feature extraction would remain unchanged (avoiding the need for complex dynamic architectures) and incrementality would be at the forest level (LTM/STM), which are perfectly suited to solve this problem.

Finally, there remains the question of the supervised learning mode. Man is able to learn by association, in an unsupervised way, once he has benefited from supervised learning. It is quite possible to transpose this operation algorithmically. Short-term memory would have the additional task of exploring the incremental data to find regularities (clusters) before integrating them into long-term memory. Thereby, we would get a little closer to the theoretical model of the complementary learning system.

References

[1] A. Baddeley and G. Hitch, "Working memory," in *G.A. Bower (Ed.), Recent advances in learning and motivation*, vol. 8, 1974, pp. 47–89.

[2] L. Postman and L. W. Phillips, "Short-term temporal changes in free recall," *The Quarterly Journal of Experimental Psychology*, vol. 17, no. 2, p. 132–138, 1965.

[3] M. Glanzer and A. R. Cunitz, "Two storage mechanisms in free recall," *Journal of Verbal Learning Verbal Behavior*, vol. 5, no. 2, pp. 351–360, 1966.

[4] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[5] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," vol. abs/1802.07569, 2018. [Online]. Available: http://arxiv.org/abs/1802.07569

[6] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, and et al., "Overcoming catastrophic forgetting in neural networks," *Proc. of the National Academy of Sciences*, vol. 114, no. 13, p. 3521–3526, 2017. [Online]. Available: http://dx.doi.org/10.1073/pnas.1611835114

[7] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. on PAMI*, vol. 40, no. 12, p. 2935–2947, 2018. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2017.2773081

[8] D. Maltoni and V. Lomonaco, "Continuous learning in single-incremental-task scenarios," *Neural Networks*, vol. 116, p. 56–73, 2019. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2019.03.010

[9] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016.

[10] G. Zhou, K. Sohn, and H. Lee, "Online incremental feature learning with denoising autoencoders," in *ICAIS*, ser. Proc. of Machine Learning Research, N. D. Lawrence and M. Girolami, Eds., vol. 22, 2012, pp. 1453–1461.

[11] L. Mici, G. I. Parisi, and S. Wermter, "An incremental self-organizing architecture for sensorimotor learning and prediction," *IEEE Trans. on Cognitive and Developmental Systems*, vol. 10, no. 4, p. 918–928, 2018. [Online]. Available: http://dx.doi.org/10.1109/TCDS.2018.2832844

[12] A. Gepperth and C. Karaoguz, "A Bio-Inspired Incremental Learning Architecture for Applied Perceptual Problems," *Cognitive Computation*, vol. 8, pp. 924 – 934, 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01418123

[13] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," 2017.

[14] D. Lopez-Paz and M. A. Ranzato, "Gradient episodic memory for continual learning," 2017.

[15] K. R. Paap and R. W. Noel, "Dual-route models of print to sound: Still a good horse race," *Psychological research*, vol. 53, no. 1, pp. 13–24, 1991.

[16] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool, "Incremental learning of ncm forests for large-scale image classification," in *CVPR*, 2014, pp. 3654–3661.

[17] S. Bernard, S. Adam, and L. Heutte, "Dynamic random forests," *Pattern Recognition Letters*, vol. 33, pp. 1580–1586, 2012.