Semi-Supervised Class Incremental Learning

Alexis Lechat, Stéphane Herbin, Frédéric Jurie

Semi-Supervised Class Incremental Learning

Alexis Lechat*† , Stéphane Herbin* and Frédéric Jurie†
*DTIS, ONERA, Université Paris-Saclay, FR-91123 Palaiseau, France
†Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, FR-14032 Caen, France
alexis.lechat@onera.fr, stephane.herbin@onera.fr, frederic.jurie@unicaen.fr

***Abstract*—This paper makes a contribution to the problem of incremental class learning, the principle of which is to sequentially introduce batches of samples annotated with new classes during the learning phase. The main objective is to reduce the drop in classification performance on old classes, a phenomenon commonly called catastrophic forgetting. We propose in this paper a new method which exploits the availability of a large quantity of non-annotated images in addition to the annotated batches. These images are used to regularize the classifier and give the feature space a more stable structure. We demonstrate on two image data sets, MNIST and STL-10, that our approach is able to improve the global performance of classifiers learned using an incremental learning protocol, even with annotated batches of small size.**

I. Introduction

The development of human perception and interpretation of the environment is acquired from a continuous process. Knowledge and skills are progressively accumulated, updated and stored throughout an individual's lifetime, and allow human perceptual system to be globally effective on large domains. The research field of Continual Learning (CL) aims to build models that imitate such behavior.

Deep Neural Network (DNN) learning is the dominant technique used to develop contemporary artificial perceptual systems. It relies on a supervised learning paradigm that assumes the availability of large annotated datasets sampling the expected outcomes of a known task and that can be exploited in a global optimization phase. Once training is done, the system and the task it is expected to solve are kept fixed.

Increasing the capacity of a DNN with new data appears to be difficult. Unless the network is completely retrained from a new set merging all data, the introduction of new samples tends to erase the previously acquired knowledge. This well known phenomenon is referred to as *Catastrophic Forgetting* in the literature [7], [9]. CL solutions focus on implementing strategies able to alleviate the forgetting and optimize a trade-off on the *stability-plasticity* dilemma [18], i.e. the DNN should have enough flexibility to be able to expand its knowledge (plasticity) while being conservative enough to prevent any forgetting (stability).

CL problems are now often divided into three scenarios [11], [21]: class-incremental learning, task-incremental learning and domain-incremental learning. This work addresses the class-incremental scenario. In this setting, the model has to learn how to solve the classification problem from a series of data with the following three goals [19]:

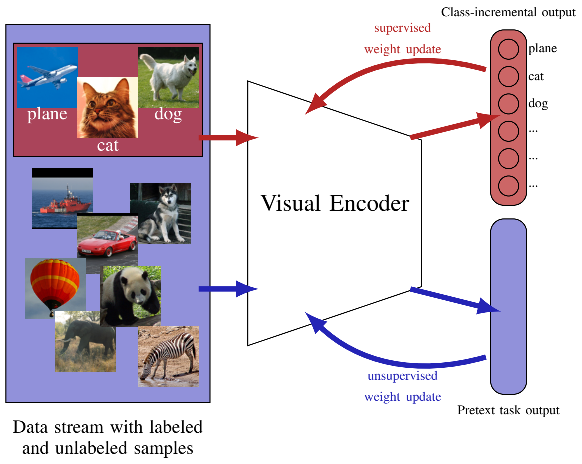

Fig. 1. Semi-supervised incremental learning: the model is fed a data stream containing both labeled and unlabeled samples. The class-incremental learning uses supervised training on annotated samples. The dedicated classification head is able to allocate new outputs when encountering the label of an unseen class for the first time (red pipeline). Meanwhile, all data provided to the model are forwarded to the unsupervised head. It uses a non-incremental pretext task (reconstruction, self-supervision...) to learn visual features without requiring annotations (blue pipeline).

1) learn classes encountered at different times during classifier lifetime;
2) provide competitive classification performance on all classes seen so far;
3) reduce memory footprint and computation cost.

The recent performance increase of computer vision functions mostly results from the possibility of jointly exploiting very deep – and also wide – neural architectures with large datasets. Somehow, performance is obtained thanks to plasticity. In CL, however, each batch of new data that comes to increase the classifier scope has a limited amount of samples, and implies that the most expressive and powerful networks, that have millions of parameters in computer vision applications, will be difficult to learn reliably.

One classical strategy to alleviate the lack of data is to consider a *semi-supervised learning* scheme with the idea of introducing some degree of regularization, at either the representation or classification levels, using an extra source of non annotated data. More precisely, we propose a new setting where the system can see a large diversity of images without any annotation and is given a few labels each time a new class is to be learned. The classifier has to optimize simultaneously the main supervised task and a pretext task as an unsupervised constraint, the former being incremental. The present work examines how a *Semi-Supervised Incremental Learning* (SSIL) baseline has some benefit to continual learning.

Our contribution is two-fold: (i) we define a baseline for class-incremental learning based on the availability of a large amount of cheap non-annotated data. The system is expected to use those alongside the incremental data stream in order to improve the learning of visual representation; (ii) we investigate how semi-supervised incremental learning provides a way to expand the plasticity with a larger DNN while ensuring enough data to regularize its stability, even on problems with very few annotated data.

The paper is organized the following way. Section II presents the work related to the development of a semi-supervised class incremental approach. The model used is described in section III and its evaluation in sections IV and V.

II. Related work

A. Class-incremental learning

Class-incremental learning relies on various strategies to alleviate the *catastrophic forgetting*. While the core idea is to preserve the previously acquired knowledge of a DNN when training on new data, different approaches are explored in the literature.

Rehearsal / replay consists in replaying old data to the model at each new training step. One way is to select some samples from the incoming data stream and store them inside an episodic memory [4], [19], [22], [26]. Rehearsal is currently the most successful strategy to counter forgetting. However, the performance is generally dependent of the memory budget allocated to the system. A way to compete with batch learning is to store all the examples seen but it is contradictory with the memory restriction aimed for. Some variants learn generative models to replace the memory [20], [23]. The data replayed is then made of artificial samples from the generator.

Regularization is a strategy consisting in implementing a protectionist policy on learned knowledge. Usually, the regularization is directly applied on the output layer using *knowledge distillation* [16], [19], [26]: the previous state of the model is used as a teacher for the new model in order to maintain the discrimination between old classes when optimizing new outputs. Other methods add the regularization on every weights of the model [1], [12], [24]. It controls the amount of knowledge accumulated in each node, preventing to further update parts of the DNN with stored knowledge and favoring the use of unused nodes.

B. Unsupervised Representation learning

Unsupervised representation learning [3] focus on learning the most adapted feature space for a designated task when annotated data are unavailable. The aim is to build an update criterion for the DNN from raw images only without any annotation. Unsupervised task is commonly referred to as *pretext task*. In computer vision, tasks such as image reconstruction or in-painting can be used to train a DNN in an unsupervised fashion. An extension of the unsupervised learning is *self-supervised learning* [13] where the pretext task is supervised with artificially generated labels. Unsupervised learning is mainly used to support a main supervised task when too few annotated data is available. This is to ensure the DNN build enough understanding of visual information in order to learn the main task [2].

The access to both annotated and non-annotated data can be used in a multi-task baseline: the *semi-supervised learning* [17]. The main supervised task and the pretext task are optimized in parallel during the same training process.

The interest for unsupervised representation learning in the incremental setting since the main hypothesis is only very limited parts of the dataset are available at each time step. Recent works show how unlabeled data can be used for incremental knowledge distillation [15], [25].

III. Method

A. Overview

At the heart of our method is a one-against-all classifier whose number of output labels, before training, is much greater than the number of possible classes. Class increment is solved by iteratively maintaining a set of labels assigned to already seen classes, increasing its size each time a new class is to be predicted by the classifier. The assignment of a new class to a label is initially obtained by choosing the best average responding predicted label from the list of unassigned labels, given the current classifier state and the new available samples, and then used as the true label to predict for the new class in a subsequent learning phase. At the end of the whole training phase, the unassigned labels can be ignored and the classifier is left with as many active output labels as classes.

Used this way, this elementary class incremental learning mechanism cannot prevent forgetting, unless stabilizing constraints are introduced. This improvement constitutes the key contribution of the paper.

Its principle is to introduce a clustering of the feature space obtained in an unsupervised way from the source of non annotated data. It consists in learning the clustering allowing the best reconstruction by means of an auto-encoder with the objective of giving some initial prior structure to the space of representation before label assignment to the new classes.

The key idea is that at the beginning of the training process, when the neural network has not seen any class, only a clustering is available. Then, when the first class arrives, the cluster that is closest to the new annotated data, in the representation space, is chosen to represent this class. The cluster thus plays a dual role: that of being the center of a cluster useful for the reconstruction in the auto-encoder, and that of being a class prototype used for classification. The unsupervised clustering task acts as a regularizer for the classifier.

From a technical point of view, an input sample is encoded by a 'one hot' vector which represents its cluster number, and, if this cluster has been assigned to a class, a class number.

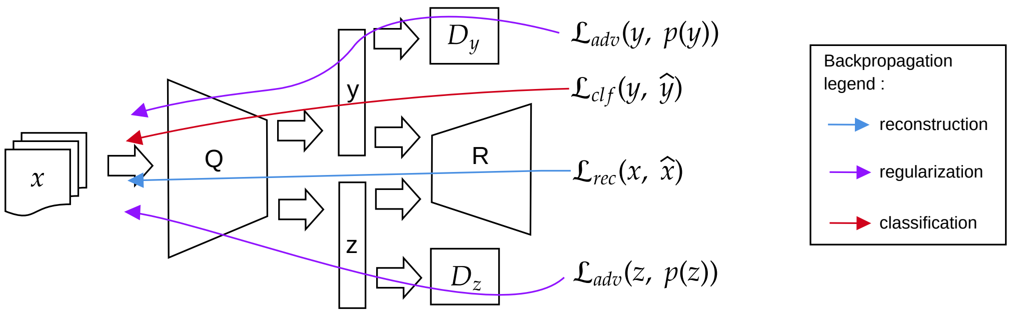
x
Q
y
z
R
D_y
D_z
$\mathcal{L}_{adv}(y,\ p(y))$
$\mathcal{L}_{clf}(y,\ \hat{y})$
$\mathcal{L}_{rec}(x,\ \hat{x})$
$\mathcal{L}_{adv}(z,\ p(z))$
Backpropagation legend :
reconstruction
regularization
classification

Fig. 2. Backpropagation steps on the Adversarial Auto-encoder. The reconstruction and regularization phases are applied for all seen data. The classification loss is back-propagated only when encountering a label.

In the following paragraphs, we present successively the definition of the problem and the notations; then we present in detail, and from a more formal point of view, the model and the associated algorithms.

B. Problem definition and notations

Class-Incremental (CI) assumes that the data live under the form of a stream of subsets $\mathcal{X} = \{X^1, X^2, \ldots, X^j, \ldots\}$ where each subset $X^j = \{x_1^j, ..., x_{n_j}^j\}$ contains n_j images from the same class y_j (one class per subset). In contrast with Task-Incremental (TI) learning, task boundaries (subset of possible classes) are not known and therefore not provided as input to the model. CI is hence more challenging than TI since, fundamentally, CI is a single task being incremented indefinitely.

Standard CI protocols suppose that only few classes (i.e. few subsets) are present at each incremental session. Once a class is submitted in a session, it will never be submitted again. For practical reason and without loss of generality, we define a constant incremental step s which is the number of classes submitted at each session. Therefore, at the beginning session i, the model $\mathcal{M}_{i-1}$ has already seen $(i-1)\times s$ classes. Let $T_i = \{X^{(i-1)s+1}, ..., X^{is}\}$ be the subsets available at session i. The goal of an incremental session is to update the model to a new state $\mathcal{M}_i$ achieving competitive performance on the new classes from T_i while retaining knowledge from the $\mathcal{M}_{i-1}$ model to maintain a global prediction capacity on all previously seen classes. The classifier is subsequently evaluated on test examples that can belong to all classes, including those present in the very first sessions.

Our framework also supposes that we have access to a stream of unlabeled images denoted by $\mathcal{U}$. These images are related to the task domain but are not taken from the training set of the classification task. The whole dataset $\mathcal{U}$ is supposed to be available at each training step in each session i but only a random subset U_i is used from $\mathcal{U}$ for unsupervised learning.

C. Architecture and losses

The core of our method is illustrated Figure 1. It consists of a network hosting an encoder that provides a representation feature space from which two tasks are performed: the incremental classification and a reconstruction of unlabeled and labeled images that acts as a regularizer.

From the so-obtained feature space, two distinct representations are computed using Fully Connected (FC) layers. The first one is a discrete latent class variable y encoding the cluster/class number under the form of a one-hot representation. The second is a continuous latent variable z, representing the *style* of the image, the style being any information which is not related to the classification task. While the classification is done using only the information contained in y (the class number when the cluster has been assigned to a class), the reconstruction uses both y and z.

The reconstruction step relies on a learning mechanism very close to the semi-supervised Adversarial Auto-Encoder (AAE) [17].

Adversarial auto-encoders, like most auto-encoders, aim at learning a low-dimension latent representation of input data through reconstruction objective. The specificity of an AAE is to impose a prior on the latent variables using an adversarial regularization.

In our case, the two latent variables are the style z which is assumed to follow a Gaussian distribution $p(z) = \mathcal{N}(0, I)$, and the class/cluster representation y, a vector of size C, that is expected to encode the true class number as a one-hot categorical vector $p(y) = \text{Cat}(y)$. C is a hyper-parameter allocating the maximal number of possible classes/clusters.

The latent variables are regularized using adversarial training with two discriminators D_y and D_z. D_y ensures that y follows a categorical distribution and does not contain any style information while D_z enforces the continuous Gaussian prior on the style encoding z.

Training the AAE is a three step process: *reconstruction*, *regularization* and *semi-supervision* as illustrated in figure 2. Let $\Theta_Q, \Theta_R, \Theta_y$ and Θ_z be the parameters of each corresponding components of the AAE: the encoder Q, the decoder R, the adversarial discriminators D_y and D_z. We denote $S = S_l \cup S_u$ to be a batch of images with both labeled (S_l) and unlabeled (S_u) samples. For all $x \in S$, we write $\hat{y}, \hat{z} = Q(x)$ the latent variables encoded by Q and $\hat{x} = R(\hat{y}, \hat{z})$ the reconstruction obtained from the latent variables.

Reconstruction: the encoder and the decoder are optimized with a squared error loss on pixels. For any image x:

$$\mathcal{L}_{\text{rec}}(\Theta_Q, \Theta_R, x) = \sum_{x \in S} ||x - \hat{x}||_2^2 \tag{1}$$

Regularization: we first train the discriminators to discriminate between the actual priors and the outputs of Q. Both style and categorical discriminators use the same optimization process. Below, we write the generic adversarial loss for any discriminator D with parameter Θ_D. True samples are denoted S, fake samples, sampled from the Gaussian prior are noted Z:

$$\mathcal{L}_{\text{adv}}(\Theta_Q, \Theta_D, S, Z) = -\sum_{x \in S} \log(1 - d(x)) - \sum_{x \in Z} \log(d(x)) \tag{2}$$

with $d(x)$ the output of D given an input x, a score between 0 (fake) and 1 (true).

Then, we optimize the encoder Q to fool the discriminators, by minimizing the following loss:

$$\mathcal{L}_{\text{adv}}(\Theta_Q, \Theta_D, S) = -\sum_{x \in S} \log(d(x)) \tag{3}$$

Semi-supervision: in this phase, only S_l is considered. We use a cross-entropy loss to update Θ_Q:

$$\mathcal{L}_{\text{clf}}(\Theta_Q, S_l) = -\sum_{(x,y) \in S_l} \sum_{c=1}^{C} \delta_{y=c} \log(\hat{y}) \tag{4}$$

D. Learning the model

To train the AAE architecture described in the previous section, we apply a succession of batch training on small subsets coming from the data streams. Since we are supposed to have no prior on the number of classes, we initialize C, the number of clusters available for categorical latent variable, at a high value (much larger than the potential number of classes). We will see in the experiments that it can be any large number and does not have any impact on performance. At each new session, the new samples, having new labels, are forwarded through the encoder once. The cluster with the highest response, among those unassigned so far, is assigned to the new class label.

Regarding the classification task, some components of y are assigned to the seen categories. The predicted class is, therefore, the component of y with the strongest output. The classification loss is hence the loss given Eq. (4). Our method also uses rehearsal to better prevent catastrophic forgetting. We denote by K the hyper-parameter representing the memory budget allocated to the rehearsal. Let $\mathcal{B}$ be the memory buffer storing old samples, we have $|\mathcal{B}| \leq K$. The memory buffer update is done randomly, i.e. a few samples are randomly sampled from $\mathcal{X}$ and stored such that $\mathcal{B}$ contains a balanced number of samples per class.

For each session i, we train $\mathcal{M}_{i-1}$ on T_i and U_i with the three aforementioned optimization phases (Algorithm 2). One epoch means one pass through the whole subset X_i. The complete continual process with data and memory management is detailed in Algorithm 1.

Algorithm 1 CLASS_INCREMENTAL

```
Input: $\mathcal{X}$, $\mathcal{U}$, $p(z)$, $p(y)$, $s$, $K$
   Initialization :
 1: $i = 0$
 2: $\mathcal{M}_0(\Theta_Q, \Theta_R, \Theta_y, \Theta_z)$ Initialize model
 3: $\mathcal{B} = \{\}$ Initialize buffer with memory budget $K$
 4: while unseen classes from $\mathcal{X}$ do
 5:    $i \leftarrow i + 1$
 6:    ASSIGN_CLUSTER($\mathcal{M}_{i-1}, T_i$)
 7:    Receive labeled subset $D_i$ with $s$ new classes from $\mathcal{T}$
 8:    Sample $U_i$ from $\mathcal{U}$
 9:    Train the model:
       $\mathcal{M}_i \leftarrow$ TRAINING_SESSION($\mathcal{M}_{i-1}, \mathcal{B}, T_i, U_i$)
10:    Store $\frac{K}{si}$ samples per class with random selection.
       $\mathcal{B} \leftarrow$ UPDATE_MEMORY($\mathcal{B}, T_i, K$)
11: end while
```

Algorithm 2 TRAINING_SESSION

```
Input: $\mathcal{M}_{i-1}(\Theta_Q, \Theta_R, \Theta_y, \Theta_z)$, $\mathcal{B}$, $T_i$, $U_i$, $p(z)$, $p(y)$
   Initialisation :
 1: $T_i \leftarrow \mathcal{B} \cup T_i$ Update labeled dataset with buffer
 2: $U_i \leftarrow T_i \cup U_i$ Merge both labeled and unlabeled for
    reconstruction task
 3: Generate true samples $Z_i \sim p(z)$
 4: Generate true samples $Y_i \sim p(y)$
 5: for each epoch do
 6:    Current parameters: $\Theta_Q, \Theta_R, \Theta_y, \Theta_z$
 7:    Optimize the reconstruction process:
       $\Theta_Q^*, \Theta_R^* \leftarrow \mathcal{L}_{\text{rec}}(\Theta_Q, \Theta_R, U_i)$
 8:    Train both discriminators:
       $\Theta_y^* \leftarrow \mathcal{L}_{\text{adv}}(\Theta_Q^*, \Theta_y, U_i, Y_i)$
       $\Theta_z^* \leftarrow \mathcal{L}_{\text{adv}}(\Theta_Q^*, \Theta_z, U_i, Z_i)$
 9:    Regularize the encoder:
       $\Theta_Q^* \leftarrow \mathcal{L}_{\text{adv}}(\Theta_Q^*, \Theta_y^*, U_i) + \mathcal{L}_{\text{adv}}(\Theta_Q^*, \Theta_z^*, U_i)$
10:    Supervised classification:
       $\Theta_Q^* \leftarrow \mathcal{L}_{\text{clf}}(\Theta_Q^*, T_i)$
11:    Update parameters:
       $\Theta_Q, \Theta_R, \Theta_y, \Theta_z \leftarrow \Theta_Q^*, \Theta_R^*, \Theta_y^*, \Theta_z^*$
12: end for
13: return $\mathcal{M}_i(\Theta_Q, \Theta_R, \Theta_y, \Theta_z)$
```

E. Justification of the proposed method

There are three justifications of our approach. First, training our model is simple and straightforward. Second, the two-component latent space allows the disentanglement of class information from style, with adapted regularization on both. The regularization with the categorical prior allows to cluster the samples regardless of their annotation status. Thus, the model is able to organize its feature space with unlabeled data, anticipating the potential apparition of labels corresponding to those instances later on. Finally, the DNN is no longer optimized on the incremental task alone but on a pretext task as well, task which is consistent throughout its lifetime. Thus, the pretext task provides a natural regularization since the model weights must maintain their ability to perform the auxiliary task.

IV. Experimental settings

A. Datasets

We experimentally validated our method on two datasets, namely MNIST and STL10, used in a class incremental setting.

a) MNIST digits: [14] is a popular classification task with 60,000 training images from 10 classes. For class incremental, MNIST is split into several subset containing s classes each introduced as in [24]. The annotated data stream consists of subsets being provided consecutively to the DNN before being discarded.

Non-annotated data comes from EMNIST [6], a dataset built upon the same source as MNIST with the same 32×32 grayscale format. EMNIST contains 814,255 handwritten characters of both digits [0-9] and letters [A-Z]. In this work, we use 3 different splits of EMNIST: EMNIST-full (complete dataset with unbalanced classes), EMNIST-digits (280,000 digits, 10 balanced classes) and EMNIST-letters (145,600 letters balanced across 26 classes).

b) STL-10 [5]: it's a more realistic dataset for unsupervised feature learning evaluation. STL-10 has two subsets: the first has 5,000 annotated images from 10 different classes, the second contains 100,000 unlabeled images sampled from a larger set of classes. All images are from IMAGENET and re-scaled to $96 \times 96 \times 3$.

Like Split-MNIST, we divide the annotated set of STL10 into disjoints subsets of s classes. We directly use the 100,000 other images as our non-annotated stream from which we randomly sample data at each session.

B. Network architectures and experimental settings

For the experiments with the MNIST dataset, we use a simple convolutional network (ConvNet) as the encoder Q, with 4 convolutional layers and one fully-connected (FC) layer producing features of dimension 256. Those features are fed into two FC layers, one producing the style code (z) of dimension 8 and one for the categorical clusters (y), one hot encoded. Number of clusters available in y is initialized to $C = 50$.

For the experiments with the STL-10 dataset, the encoder is a modified ResNet18 [10]: the number of neurons in each layer has been divided by 2 for computational constraints. This reduction should have limited impact on the performance since ResNet18 is originally designed to work with higher resolution images ($224 \times 224 \times 3$) from ImageNet. Style latent space has 64 dimensions and $C = 120$.

For both datasets, the discriminators are MLP with 2 layers of 3,000 hidden units similar to the one used in the original AAE paper. In our experiments, we found out that the architecture of the decoder had less impact on the overall performance. We opted for simple convolutional decoders with transpose convolution for upsampling. The size of the decoder is adapted to have about the same number of parameters as the encoder.

V. Experiments

A. Class-incremental learning performance

The class incremental is evaluated with $s = 2$, which means 5 consecutive sessions for both datasets. Since the aim of our contribution is to illustrate how SSIL provides an adapted framework when few labeled data are available, we only use 2,000 annotated data from MNIST (200 per class), instead of the full 60,000. No restriction is imposed on the amount of unlabeled data available.

We compare our solution to other competitive methods from the literature: LwF [16], DMC [25], iCaRL [19] and WA [26]. All of them are regularized and use Knowledge Distillation (KD). iCaRL and WA combine KD with a rehearsal routine using a memory buffer. Aside from those methods, we also evaluate the performance of Fine-Tuning, i.e. training a DNN on the class-incremental task with no solution to alleviate the catastrophic forgetting by simply fine tuning the network to the novel classes, and Naive Rehearsal which is fine-tuning with a memory buffer replaying old data. The Oracle performance is obtained using Naive Rehearsal with an unlimited buffer size (all samples seen before are stored). Since those methods are conceived for fully supervised, the whole 60,000 labels are used when training on MNIST. In comparison, and as mentioned above, our method uses only 2,000 labeleled images.

To ensure fairness in our evaluation, all methods were re-implemented using the same model as the encoder used in our AAE (ConvNet and ResNet18). When rehearsal is involved, we impose the memory budget $K = 400$ and $K = 500$ respectively for MNIST and STL-10. Accuracy after each session is measured on the validation set using only seen classes. Final accuracy (after the 5-th session) and average accuracy over all the sessions are measured on the test set.

Table I gives the performances of all the methods we have just mentioned. Our method outperforms all previous methods, on both dataset. Even on MNIST where our model was fed only 2,000 labels through its lifetime (about 3% of the labels seen by other methods), our method gets overall better performance. Repeating the experiment with various splits of EMNIST as unlabeled data stream $\mathcal{U}$ gives consistent results. Even when unlabeled data are made of instances of classes never learned, i.e. handwritten letters, the model still reaches state-of-the-art accuracy, showing that our AAE profit from additional data despite them being not directly related to the main task. Still, the best model is the one seeing only digits in $\mathcal{U}$, hinting that if the model is able to specialize its features on a distribution similar to the main task, continual learning inherently benefits from it.

On STL-10, our model gets significantly better results compared to previous methods. However, it is important to

TABLE I
COMPARISON OF LATEST AND AVERAGE ACCURACY OF DIFFERENT CLASS-INCREMENTAL LEARNING METHODS ON MNIST AND STL-10

Method	MNIST		STL-10	
	Latest (%)	Average (%)	Latest (%)	Average (%)
Oracle	99.4	99.7	67.2	73.5
Fine-Tuning	19.8	44.9	16.2	38.3
LwF [16]	71.3	85.2	17.9	42.5
DMC [25]	81.1	87.4		
Naive Rehearsal	93.7	97.6	43.8	62.0
iCaRL [19]	95.3	97.9	42.6	63.0
WA [26]	96.0	98.3	47.3	63.5
Ours[a]	96.9	98.5	**57.3**	**72.0**
Ours[b] (EMNIST-digits)	**98.1**	**99.0**		
Ours[b] (EMNIST-letters)	95.9	98.5		

[a] Our standard baseline on MNIST uses EMNIST-full as unlabeled data stream.
[b] Additional results on MNIST benchmark when using EMNIST-digits and EMNIST-letters as unlabeled data stream instead of the whole EMNIST.

TABLE II
LATEST ACCURACY OBTAINED FOR VARIOUS CHOICE OF C, THE HYPER-PARAMETER CONTROLLING THE NUMBER OF CLUSTERS AVAILABLE IN THE AAE. EXPERIMENTS ON MNIST WITH EMNIST-FULL.

C	10	15	50	100	200
%	96.9	97.4	96.9	96.9	96.6

note that the STL-10 dataset is built for unsupervised learning evaluation. This means that the classification problem considering only the few labeled images is challenging. This explains the low performance of previous methods (even for the oracle) while ours successfully exploits the vast amount of non-annotated data available inside STL-10.

Table II gives the performance of our methods on MNIST with EMNIST-full as unlabeled data stream, for different values of C, the number of clusters. We can see that as long as this number is large enough, the performance is good, even if there is a very slight decrease when the number of clusters is very large compared to the number of classes

B. Self-supervision to learn representations ahead of the continual learning

In order to provide a fairer ground of comparison, since our method uses unlabeled data and not the other methods, we add a pre-training step using self-supervised learning to all the methods. We have chosen the RotNet algorithm [8] for this purpose. The images are randomly rotated (0°, 90°, 180° and 270°) and the self-supervision uses the rotation angles as labels, the task being to predict the rotation angle. This task showed good results as self-supervised learning to prepare the model to a classification task [13]. In practice, for this task, we train the Resnet18 on the unlabeled set of STL-10.

We used the self-supervised Resnet18 weights as initialization of the DNN inside all the rehearsal-based methods, including our AAE encoder. This way, we ensure that all the models have both seen the unlabeled and the labeled images of STL-10. Figure 3 shows the accuracy measured at each session on seen classes. Average accuracy is provided in parenthesis inside the legend of the figure. The impact of the self-supervised pre-training is clear, all the models stepped up and are now above the ones from the previous experiment. Our method continues to outperform the others.

Like batch learning, a better initialization results in a better incremental performance. There are multiple ways to pre-train a DNN: using a model trained on another supervised dataset like ImageNet or using unsupervised/self-supervised learning like we did with RotNet. In fact, when the amount of labeled data is as limited as in STL-10, using a deep model like our modified Resnet18 ($\approx$ 28M parameters) raises other training issues inherent to few shot learning. This illustrate our interest for representation learning in this incremental learning work since both topics share some common issues. In fact, we think that if our method works better than the others in the experiment without pre-training, the reason is because having more data (even without labels) allowed us to optimize many more parameters, the reconstruction task providing regularization.

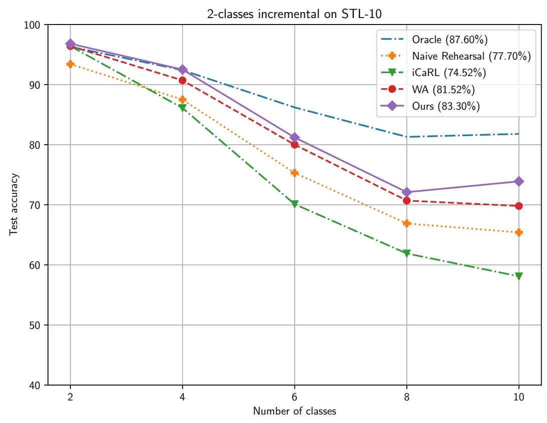

Fig. 3. Comparison of different rehearsal strategies initialized with a self-supervised encoder. Legend gives the average accuracy for each method.

VI. CONCLUSION

In this article we have proposed a new approach to continual learning. The key idea is to structure the representation space by training an adversarial auto-encoder to reconstruct images as incremental learning takes place. This brings to the network a regularization constraint that stabilizes the space of representation over time. We have shown through experiments on two image datasets the relevance of the proposed approach.

REFERENCES

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11207. Springer, 2018, pp. 144–161.

[2] Y. M. Asano, C. Rupprecht, and A. Vedaldi, "A critical analysis of self-supervision, or what we can learn from a single image," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[3] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012.

[4] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11216. Springer, 2018, pp. 241–257.

[5] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, ser. JMLR Proceedings, G. J. Gordon, D. B. Dunson, and M. Dudík, Eds., vol. 15. JMLR.org, 2011, pp. 215–223.

[6] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: extending MNIST to handwritten letters," in *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*. IEEE, 2017, pp. 2921–2926.

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

[8] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *6th International Conference on Learning Representations, ICLR*, 2018.

[9] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks," *arXiv:1312.6211 [cs, stat]*, Dec. 2013.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778.

[11] Y.-C. Hsu, Y.-C. Liu, and Z. Kira, "Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines," *arXiv:1810.12488 [cs]*, Oct. 2018.

[12] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[13] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 1920–1929.

[14] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[15] K. Lee, K. Lee, J. Shin, and H. Lee, "Overcoming catastrophic forgetting with unlabeled data in the wild," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 312–321. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00040

[16] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[17] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow, "Adversarial autoencoders," *CoRR*, vol. abs/1511.05644, 2015.

[18] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[19] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[20] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 2990–2999.

[21] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv:1904.07734 [cs, stat]*, Apr. 2019.

[22] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 374–382.

[23] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu, "Incremental classifier learning with generative adversarial networks," *CoRR*, vol. abs/1802.00853, 2018.

[24] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 3987–3995.

[25] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. P. Heck, H. Zhang, and C. J. Kuo, "Class-incremental learning via deep model consolidation," in *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*. IEEE, 2020, pp. 1120–1129.

[26] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia, "Maintaining discrimination and fairness in class incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 208–13 217.