OvA-INN: Continual Learning with Invertible Neural Networks

Guillaume Hocquet
CEA, LIST
Gif-sur-Yvette CEDEX, France
guillaume.hocquet@cea.fr

Olivier Bichler
CEA, LIST
Gif-sur-Yvette CEDEX, France
olivier.bichler@cea.fr

Damien Querlioz
CNRS, Université Paris-Saclay
Gif-sur-Yvette CEDEX, France
damien.querlioz@u-psud.fr

***Abstract*—In the field of Continual Learning, the objective is to learn several tasks one after the other without access to the data from previous tasks. Several solutions have been proposed to tackle this problem but they usually assume that the user knows which of the tasks to perform at test time on a particular sample, or rely on small samples from previous data and most of them suffer of a substantial drop in accuracy when updated with batches of only one class at a time. In this article, we propose a new method, OvA-INN, which is able to learn one class at a time and without storing any of the previous data. To achieve this, for each class, we train a specific Invertible Neural Network to extract the relevant features to compute the likelihood on this class. At test time, we can predict the class of a sample by identifying the network which predicted the highest likelihood. With this method, we show that we can take advantage of pretrained models by stacking an Invertible Network on top of a feature extractor. This way, we are able to outperform state-of-the-art approaches that rely on features learning for the Continual Learning of MNIST and CIFAR-100 datasets. In our experiments, we reach 72% accuracy on CIFAR-100 after training our model one class at a time.**

***Index Terms*—Continual Learning, Catastrophic forgetting**

I. Introduction

A typical Deep Learning workflow consists in gathering data, training a model on these data and finally deploying the model in the real world [1]. If one would need to update the model with new data, it would require to merge the old and new data and process a training from scratch on this new dataset. Nevertheless, there are circumstances where this method may not apply. For example, it may not be possible to store the old data because of privacy issues (health records, sensible data) or memory limitations (embedded systems, very large datasets). In order to address those limitations, recent works propose a variety of approaches in a setting called **Continual Learning** [2].

In Continual Learning, we aim to learn the parameters w of a model on a sequence of datasets $\mathcal{D}_i = \{(x_i^j, y_i^j)\}_{j=1}^{n_i}$ with the inputs $x_i^j \in \mathcal{X}^i$ and the labels $y_i^j \in \mathcal{Y}^i$, to predict $p(y^*|w, x^*)$ for an unseen pair (x^*, y^*). The training has to be done on each dataset, one after the other, without the possibility to reuse previous datasets. The performance of a Continual Learning algorithm can then be measured with two protocols : *multi-head* or *single-head*. In the multi-head scenario, the task identifier i is known at test time. For evaluating performances on task i, the set of all possible labels is then $\mathcal{Y} = \mathcal{Y}^i$. Whilst in the single-head scenario, the task identifier is unknown, in that case we have $\mathcal{Y} = \cup_{i=1}^N \mathcal{Y}^i$ with N the number of tasks learned so far. For example, let us say that the goal is to learn MNIST sequentially with two batches: using only the data from the first five classes and then only the data from the remaining five other classes. In multi-head learning, one asks at test time to be able to recognize samples of 0-4 among the classes 0-4 and samples of 5-9 among classes 5-9. On the other hand, in single-head learning, one cannot assume from which batch a sample is coming from, hence the need to be able to recognize any samples of 0-9 among classes 0-9. Although the former one has received the most attention from researchers, the last one fits better to the desiderata of a Continual Learning system as expressed in [3] and [4]. The single-head scenario is also notoriously harder than its multi-head counterpart [5]. We will mainly be focusing on the single-head setting in the present work.

Updating the parameters with data from a new dataset exposes the model to drastically deteriorate its performance on previous data, a phenomenon known as *catastrophic forgetting* [6]. To alleviate this problem, researchers have proposed a variety of approaches such as storing a few samples from previous datasets [7], adding *distillation* regularization [8], updating the parameters according to their usefulness on previous datasets [9], using a generative model to produce samples from previous datasets [10]. Despite those efforts toward a more realistic setting of Continual Learning, one can notice that, most of the time, results are proposed in the case of a sequence of batches of multiple classes. This scenario often ends up with better accuracy (because the learning procedure highly benefits of the diversity of classes to find the best tuning of parameters) but it does not illustrate the behavior of those methods in the worst case scenario. In fact, Continual Learning algorithms should be robust in the size of the batch of classes.

In this work, we propose to implement a method specially designed to handle the case where each task consists of only one class. It will therefore be evaluated in the single-head scenario. Our approach, named One-versus-All Invertible Neural Networks (OvA-INN), is based on an invertible neural network architecture proposed by [11]. We use it in a One-versus-All strategy : each network is trained to make a prediction of a class and the most confident one on a sample is used to identify the class of the sample. In contrast to most other methods, the

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

training phase of each class can be independently executed from one another.

The contributions of our work are *(i)* a new approach for Continual Learning with one class per batch; *(ii)* a neural architecture based on Invertible Networks that does not require to store any of the previous data; *(iii)* state-of-the-art results on several tasks of Continual Learning for Computer Vision (CIFAR-100, MNIST) in this setting.

We start by reviewing the closest methods to our approach in Section II, then explain our method in Section III, analyse its performances in Section IV and identify limitations and possible extensions in Section V.

II. Related work

Generative models: Inspired by biological mechanisms such as the hippocampal system that rapidly encodes recent experiences and the memory of the neocortex that is consolidated during sleep phases, a natural approach is to produce samples of previous data that can be added to the new data to learn a new task. FearNet [10] relies on an architecture based on an autoencoder, whereas Deep Generative Replay [12] and Parameter Generation and Model Adaptation [13] propose to use a generative adversarial network. Those methods present good results but require complex models to be able to generate reliable data. Furthermore, it is difficult to assess the relevance of the generated data to conduct subsequent training iterations.

Coreset-based models: These approaches alleviate the constraint on the availability of data by allowing the storage of a few samples from previous data (which are called *coreset*). iCaRL [7] and End-to-end IL [14] store 2000 samples from previous batches and rely on respectively a distillation loss and a mixture of cross-entropy and distillation loss to alleviate forgetting. The authors of SupportNet [15] have also proposed a strategy to select relevant samples for the coreset. Gradient Episodic Memory [16] ensures that gradients computed on new tasks do not interfere with the loss of previous tasks. Those approaches give the best results for single-head learning. But, similarly to generated data, it is not clear which data may be useful to conduct further training iterations. In this paper, we are challenging the need of the coreset for single-head learning.

Distance-based models: These methods propose to embed the data in a space which can be used to identify the class of a sample by computing a distance between the embedding of the sample and a reference for each class. Among the most popular, we can cite Matching Networks [17] and Prototypical Networks [18], but these methods have been mostly applied to few-shot learning scenarios rather than continual.

Regularization-based approaches: These approaches present an attempt to mitigate the effect of catastrophic forgetting by imposing some constraints on the loss function when training subsequent classes. Elastic Weight Consolidation [9], Synaptic Intelligence [19] and Memory Aware Synapses [20] prevent the update of weights that were the most useful to discriminate between previous classes. Hence, it is possible to constrain the learning of a new task in such a way that the most relevant weights for the previous tasks are less susceptible to be updated. Learning without forgetting [8] proposes to use knowledge distillation to preserve previous performances. The network is divided in two parts : the shared weights and the dedicated weights for each task. When learning a new task A, the data of A get assigned "soft" labels by computing the output by the network with the dedicated weight for each previous task. Then the network is trained with the loss of task A and is also constrained to reproduce the recorded output for each other tasks. In [21], the authors propose to use an autoencoder to reconstruct the extracted features for each task. When learning a new task, the feature extractor is adapted but has to make sure that the autoencoder of the other tasks are still able to reconstruct the extracted features from the current samples. While these methods obtain good results for learning one new task, they become limited when it comes to learn several new tasks, especially in the one class per batch setting.

Expandable models: In the case of the multi-head setting, it has been proposed to use the previously learned layers and complete them with new layers trained on a new task. This strategy is presented in Progressive Networks [22]. In order to reduce the growth in memory caused by the new layers, the authors of Dynamically Expandable Networks [23] proposed an hybrid method which retrains some of the previous weights and add new ones when necessary. Although these approaches work very well in the case of multi-head learning, they cannot be adapted to single-head. On the contrary, it is possible to run OvA-INN in a multi-head setting, we demonstrate this in section IV-D.

III. Class-by-Class Continual Learning with Invertible Networks

A. Motivations and Challenge

We investigate the problem of training several datasets in a sequential fashion with batches of only one class at a time. Most approaches of the state-of-the-art rely on updating a feature extractor when data from a new class are available. But this strategy is unreliable in the special case we are interested in, namely batches of data from only one class. With few or no sample of negative data, it is very inefficient to update the weights of a network because the setting of deep learning typically involves vast amounts of data to be able to learn to extract valuable features. Without enough negative samples, the training is prone to overfit the new class. Recent works have proposed to rely on generative models to overcome this lack of data by generating samples of old classes. Nevertheless, updating a network with sampled data is not as efficient as with real data and, on the long run, the generative quality of early classes suffer from the multiple updates.

B. Out-of-distribution detection for Continual Learning

Our approach consists in interpreting a Continual Learning problem as several out-of-distribution (OOD) detection problems. OOD detection has already been studied for neural networks and can be formulated as a binary classification problem which consists in predicting if an input x was sampled from the same distribution as the training data or from a different distribution [24], [25]. Hence, for each class, we can train a network to predict if an input x is likely to have been sampled from the distribution of this class. The class with the highest confidence can be used as the prediction of the class of x. This training procedure is particularly suitable for Continual Learning since the training of each network does not require any negative sample.

Using the same protocol as NICE [11], for a class i, we can train a neural network f_i to fit a prior distribution p and compute the exact log-likelihood l_i on a sample x :

$$l_i(x) = \log(p(f_i(x)) \tag{1}$$

To obtain the formulation of log-likelihood as expressed in Equation 1, the network f_i has to respect some constraints discussed in Section III-C. Keeping the same hypothesis as NICE, we consider the case where p is a distribution with independent components p_d :

$$p(f_i(x)) = \prod_d p_d(f_{i,d}(x)) \tag{2}$$

In our experiments, we considered p_d to be standard normal distributions. Although, it is possible to learn the parameters of the distributions, we found experimentally that doing so decreases the results. Under these design choices, the computation of the log-likelihood becomes :

$$\begin{aligned} l_i(x) &= \sum_d \log(p_d(f_{i,d}(x)) && (3) \\ &= -\sum_d \frac{1}{2} f_{i,d}(x)^2 + \sum_d \log\left(\frac{1}{\sqrt{2\pi}}\right) && (4) \\ &= -\frac{1}{2}\|f_i(x)\|_2^2 + \beta && (5) \end{aligned}$$

where $\beta = -n \log\left(\sqrt{2\pi}\right)$ is a constant term.

Hence, identifying the network with the highest log-likelihood is equivalent to finding the network with the smallest output norm.

C. Invertible Neural Networks

The neural network architecture proposed by NICE is designed to operate a change of variables between two density functions. This assumes that the network is invertible and respects some constraints to make it efficiently computable.

Invertible Network can be modeled as a stack of several invertible blocks. An invertible block (see Figure 1) consists in splitting the input x into two subvectors x_1 and x_2 of equal size; then successively applying two (non necessarily invertible) networks f_1 and f_2 following the equation :

$$\begin{cases} y_1 = f_1(x_2) + x_1 \\ y_2 = f_2(y_1) + x_2, \end{cases} \tag{6}$$

and finally, concatenate y_1 and y_2. The inverse operation can be computed with :

$$\begin{cases} x_2 = y_2 - f_2(y_1) \\ x_1 = y_1 - f_1(x_2). \end{cases} \tag{7}$$

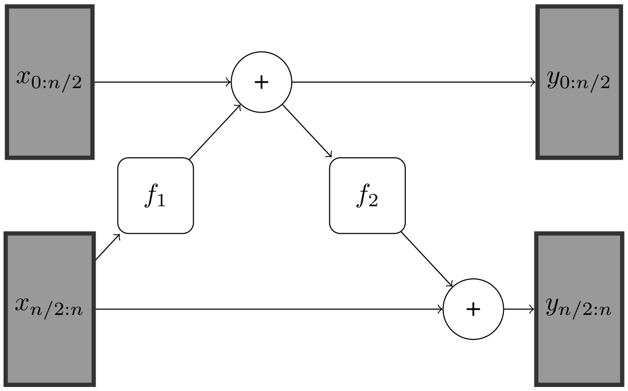


Fig. 1: Forward pass in an invertible block. x is split in $x_{0:n/2}$ and $x_{n/2:n}$. f_1 and f_2 can be any type of Neural Networks as long as the dimension of their output dimension is the same as their input dimension. In our experiments, we stack two of these blocks one after the other and use fully-connected feedforward layers for f_1 and f_2.

These invertible equations illustrate how Invertible Networks operate a bijection between their input and their output. Other invertible architectures can be used to learn a transformation that maximizes the likelihood on a dataset but we choose this one for its simplicity.

D. Continual Learning setting

We propose to specialize each Invertible Network to a specific class by training them to output a vector with small norm when presented with data samples from their class. Given a dataset $\mathcal{X}_i$ of class i and an Invertible Network f_i, our objective is to minimize the loss $\mathcal{L}$:

$$\mathcal{L}(\mathcal{X}_i) = \frac{1}{|\mathcal{X}_i|} \sum_{x \in \mathcal{X}_i} \|f_i(x)\|_2^2 \tag{8}$$

Once the training has converged, the weights of this network will not be updated when new classes will be added. At inference time, after learning t classes, the predicted class y^* for a sample x is obtained by running each network and identifying the one with the smallest output :

$$y^* = \underset{y=1,...t}{\arg\min} \|f_y(x)\|_2^2 \tag{9}$$

As it is common practice in image processing, one can also use a preprocessing step by applying a common pretrained feature extractor beforehand. Using a pretrained feature extractor allow to save time and memory since it is usually not necessary to retrain low level features to discriminate new classes. This fixed representation of data can be transfered from a model trained on Imagenet or from unlabelled data [26]. Noting ϕ this fixed pretrained model, the inference equation can be expressed as :

$$y^* = \underset{y=1,...t}{\arg\min} \|f_y(\phi(x))\|_2^2. \tag{10}$$

IV. Experimental Results

We compare our method against several state-of-the-art baselines for single-head learning on MNIST and CIFAR-100 datasets.

A. Implementation details

Topology of OvA-INN: Due to the bijective nature of Invertible Networks, their output size is the same as their input size, hence the only way to change their size is by changing the depth or by compressing the parameters of the intermediate networks f_1 and f_2. In our experiments, these networks are fully connected layers. To reduce memory footprint, we replace the square matrix of parameters W of size $n \times n$ by a product of matrices AB of sizes $n \times m$ and $m \times n$ (with a compressing factor for the first and second block $m = 16$ for MNIST and $m = 32$ for CIFAR-100).

Regularization: When performing learning one class at a time, the amount of training data can be highly reduced: only 500 training samples per class for CIFAR-100. To avoid overfitting the training set, we found that adding a weight decay regularization could increase the validation accuracy. More details on the hyperparameters choices can be found in Appendix A.

Rescaling: As ResNet has been trained on images of size 224×224, we rescale CIFAR-100 images to match the size of images from Imagenet.

B. Evaluation on MNIST

We start by considering the MNIST dataset [27], as it is a common benchmark that remains challenging in the case of single-head Continual Learning.

Baselines: We compare our approach with methods based on generative models such as Parameter Generation and Model Adaptation (PGMA) [13] and Deep Generative Replay (DGR) [12]; and with methods based on exemplar storage such as iCaRL [7], SupportNet [15], GEM [16] and with RPS-Net [28] which rely on a random path selection algorithm.

We report the results from the original papers; except for iCarL and SupportNet where we use the provided code of SupportNet to compute the results for MNIST with two layers of convolutions with poolings and a fully connected last layer. We have also set the coreset size to $s = 800$ samples.

Analysis: We report the average accuracy over all the classes after the networks have been trained on all batches (See Table I). Our architecture does not use any pretrained feature extractor common to every classes (contrarily to our CIFAR-100 experiment) : each sample is processed through an Invertible Network, composed of two stacked invertible blocks. Our approach presents better results than all the other reference methods while having a smaller cost in memory and being trained by batches of only one class.

C. Evaluation on CIFAR-100

We now consider a more complex image dataset with a greater number of classes [29]. This allows us to make comparisons in the case of a long sequence of data batches and to illustrate the value of using a pretrained feature extractor for Continual Learning.

TABLE I: Comparison of accuracy and memory cost in number of parameters (and memory usage for storing samples if relevant) of different approaches on MNIST at the end of the Continual Learning. The Learning type column indicates the number of classes used at each training step.

Model	Accuracy	Memory cost	Learning type
PGMA [13]	81.70%	6,000k	2 by 2
SupportNet [15]	89.90%	940k	2 by 2
GEM [16]	92.20%	4,919k	2 by 2
DGR [12]	95.80%	12,700k	2 by 2
iCaRL [7]	96.00%	940k	2 by 2
RPS-Net [28]	96.16%	4,919k	2 by 2
OvA-INN	**96.40%**	**520k**	**1 by 1**

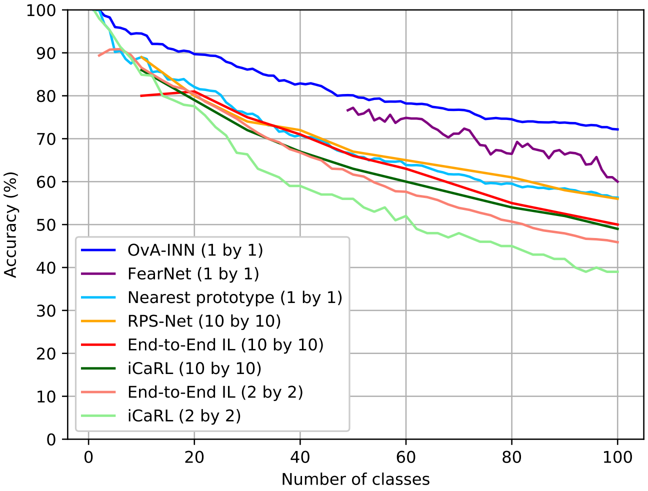


Fig. 2: Comparison of the accuracy of several Continual Learning methods on CIFAR-100 with various batches of classes. FearNet's curve has no point before 50 classes because the first 50 classes are learned in a non-continous fashion.

Baselines: FearNet [10] is based on a generative model. It uses a pretrained ResNet48 features extractor. In their experiments, the authors rely on a warm-up phase. Namely, the network is first trained with all the first 50 classes of CIFAR-100, and subsequently learns the next 50 classes one by one in a continual fashion. iCaRL [7], End-to-end IL [14] both use 2k exemplars from previous classes and retrain a ResNet32 features extractor respectively with a distillation loss and with a cross-entropy together with distillation loss. RPS-Net [28] also use 2k exemplars from previous classes but it trains several ResNet18 in parallel and assign different paths for predicting each classes. Nearest prototype is our implementation of the method consisting in computing the mean vector (prototype) of the output of a pretrained ResNet32 for each class at train time. Inference is performed by finding the closest prototype to the ResNet output of a given sample.

Analysis: Image data are provided by batch of classes. When the training on a batch ($\mathcal{D}_i$) is completed, the accuracy of the classifier is evaluated on the test data of classes from all

TABLE II: Comparison of the accuracy of several multi-head Continual Learning methods on CIFAR-100 on 10 taks of 10 classes.

Model	Accuracy
EWC [9]	81.34%
Progressive Networks [22]	88.19%
DEN [30]	92.25%
OvA-INN	92.58%

previous batches ($\mathcal{D}_1, ..., \mathcal{D}_i$). We report the results from the literature with various sizes of batch when they are available.

OvA-INN uses the weights of a ResNet32 pretrained on ImageNet and never update them. FearNet also uses pretrained weights from a ResNet. iCaRL, End-to-End IL and RPS-Net use a similar architecture but retrain it from scratch at the beginning and fine-tune it with each new batch.

The performance of the Nearest prototype baseline proves that there is high benefit in using pretrained feature extractor on this kind of dataset. FearNet shows better performance by taking advantage of a warm-up phase with 50 classes. We can see that OvA-INN is able to clearly outperform all the other approaches, reaching 72% accuracy after training on 100 classes. For comparison, we were only able to reach 76% accuracy on a Resnet trained on all the CIFAR-100 data at once. We can see that the performances of methods retraining ResNet from scratch (iCaRL, End-to-End IL and RPS-Net) quickly deteriorate compared to those using pretrained parameters. Even with larger batches of classes, the gap is still present.

It can be surprising that at the end of its warm-up phase, FearNet still has an accuracy bellow OvA-INN, even though it has been trained on all the data available at this point. It should be noted that FearNet is training an autoencoder and uses its encoding part as a features extractor (stacked on the ResNet) before classifying a sample. This can diminish the discriminative power of the network since it is also constrained to reproduce its input (only a single autoencoder is used for all classes).

D. *Experiments in multi-head Continual Learning*

We provide additional experimental results in the multi-head learning of CIFAR100 with 10 tasks of 10 classes each in Table II. The training procedure of OvA-INN does not change from the usual single-head learning but, at test time, the evaluation is processed by batches of 10 classes (instead of the whole dataset). The accuracy score is the average accuracy over all 10 tasks. We report the results from various methods of multi-head learning. Although our approach is able to match state-of-the-art results in accuracy, it should be noticed that it is drastically more memory and time consuming than some baselines. OvA-INN requires to train entire new layers whilst DEN is optimized to use as little additional parameters as possible to learn a new task. That being said, none of the compared methods can be applied in the single-head setting.

V. Discussion

A. *Visualization*

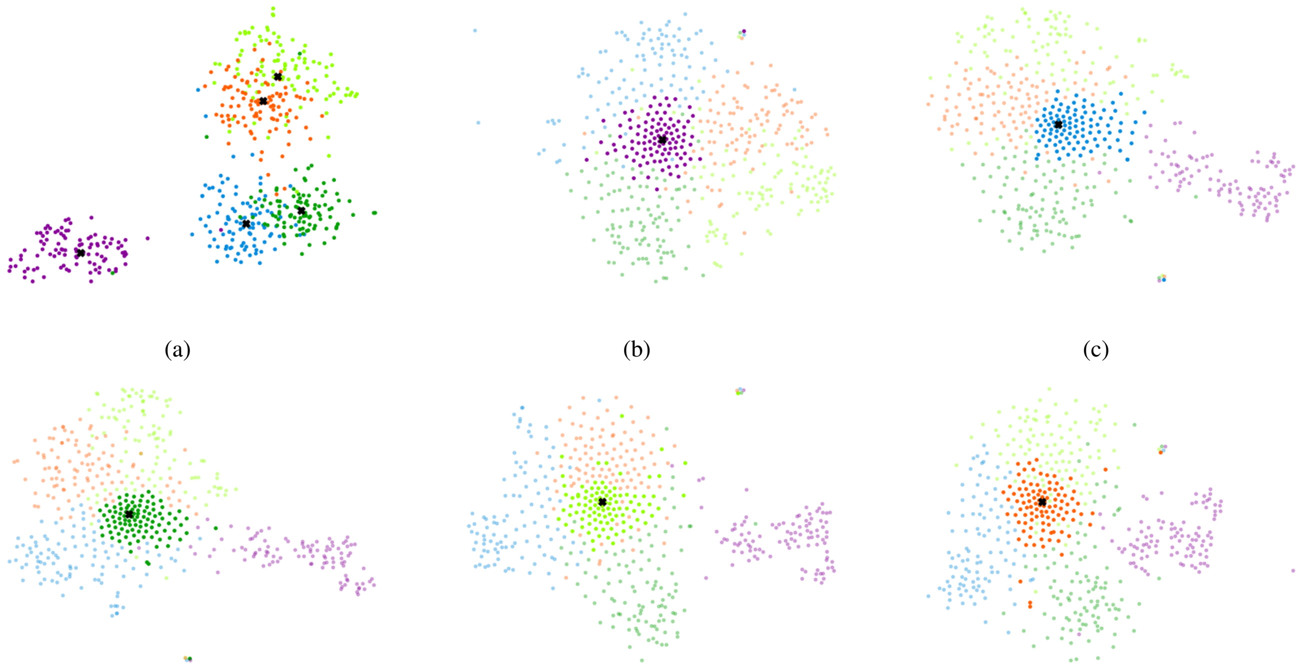
To further understand the effect of an Invertible Network on the feature space of a sample, we propose to project the different feature spaces in 2D. To perform this projection, we rely on the t-SNE algorithm, which is an non-linear dimensionality reduction technique commonly used for high dimensional data visualization [31]. In Figure 3, we project the features of the first five classes of CIFAR-100 test set, each class is represented with a different color. The projection of features extracted by the Resnet is displayed on a single plot since those features are computed from a single network. Cluster centers are represented by black crosses. A sample closer to a cluster center indicates a higher confidence of the network to predict the class corresponding to the cluster. For the Resnet projection, we can see that some samples appears in a cluster of a different class. Those samples are likely to be misclassified by a distance-based method since the dimensionality reduction did not manage to distinguish them from samples of other classes. For the Invertible Networks, we display the projection of features computed by each of the five networks on separate plots. In this case, we observe that classes that are already well represented in a cluster with ResNet features (like violet class) are clearly separated from the clusters of Invertible Networks; an classes represented with ambiguity with ResNet features (like light green and red) are better clustered in the Invertible Network space.

B. *Limitations*

A limiting factor in our approach is the necessity to add a new network each time one wants to learn a new class. This makes the memory and computational cost of OvA-INN linear with the number of classes. Recent works in networks merging could alleviate the memory issue by sharing weights [32] or relying on weights superposition [33]. This being said, we showed that Ova-INN was able to achieve superior accuracy on CIFAR-100 class-by-class training than approaches reported in the literature, while using less parameters.

Another constraint of using Invertible Networks is to keep the size of the output equal to the size of the input. When one wants to apply a feature extractor with a high number of output channels, it can have a very negative impact on the memory consumption of the invertible layers. Feature Selection or Feature Aggregation techniques may help to alleviate this issue [34].

Finally, we can notice that our approach is highly dependent on the quality of the pretrained feature extractor. In our CIFAR-100, we had to rescale the input to make it compatible with ResNet. Nonetheless, recent research works show promising results in training feature extractors in very efficient ways [35]. Because it does not require to retrain its feature extractor, we can foresee better performance in class-by-class learning with OvA-INN as new and more efficient feature extractors are discovered.



(a) (b) (c)

(d) (e) (f)

Fig. 3: t-SNE projections of feature spaces for five classes from CIFAR-100 test set (colors are given by the ground truth). *(a)*: feature space before applying Invertible Networks (black crosses are the clusters centers). *(b),(c),(d),(e),(f)*: each feature space after the Invertible Network of each class. The samples of a class represented by a network are clustered around the zero vector (black cross) whilst the samples from other classes appear further away from the cluster.

C. *Future research directions*

One could try to incorporate our method in a Reinforcement Learning scenario where various situations can be learned separately in a first phase (each situation with its own Invertible Network). Then during a second phase where any situation can appear without the agent explicitly told in which situation it is in, the agent could rely on previously trained Invertible Networks to improve its policy. This setting is closely related to *Options* in Reinforcement Learning.

Also, in a regression setting, one can add a fully connected layer after an intermediate layer of an Invertible Network and use it to predict the output for the trained class. At test time, one only need to read the output from the regression layer of the Invertible Network that had the highest confidence.

Other invertible architectures, such as Neural Ordinary Differential Equations [36], could be studied to alleviate the limitations of the NICE architecture used in this work.

VI. Conclusion

In this paper, we proposed a new approach for the challenging problem of single-head Continual Learning without storing any of the previous data. On top of a fixed pretrained neural network, we trained for each class an Invertible Network to refine the extracted features and maximize the log-likelihood on samples from its class. This way, we show that we can predict the class of a sample by running each Invertible Network and identifying the one with the highest log-likelihood. This setting allows us to take full benefit of pretrained models, which results in very good performances on the class-by-class training of CIFAR-100 compared to prior works.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[2] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *arXiv preprint arXiv:1802.07569*, 2018.

[3] S. Farquhar and Y. Gal, “Towards robust evaluations of continual learning,” *arXiv preprint arXiv:1805.09733*, 2018.

[4] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” *CoRR*, vol. abs/1904.07734, 2019. [Online]. Available: http://arxiv.org/abs/1904.07734

[5] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, “Riemannian walk for incremental learning: Understanding forgetting and intransigence,” *arXiv preprint arXiv:1801.10112*, 2018.

[6] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” *Psychology of Learning and Motivation - Advances in Research and Theory*, vol. 24, pp. 109–165, 1989.

[7] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: Incremental classifier and representation learning,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 5533–5542.

[8] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018.

[9] 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*, p. 201611835, 2017.

[10] R. Kemker and C. Kanan, "Fearnet: Brain-inspired model for incremental learning," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[11] L. Dinh, D. Krueger, and Y. Bengio, "NICE: non-linear independent components estimation," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[12] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 2990–2999.

[13] W. Hu, Z. Lin, B. Liu, C. Tao, Z. Tao, J. Ma, D. Zhao, and R. Yan, "Overcoming catastrophic forgetting for continual learning via model adaptation," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[14] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 233–248.

[15] Y. Li, Z. Li, L. Ding, Y. Hu, W. Chen, and X. Gao, "SupportNet: a novel incremental learning framework through deep learning and support data," *bioRxiv*, 2018.

[16] D. Lopez-Paz *et al.*, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6467–6476.

[17] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 3637–3645.

[18] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 4077–4087.

[19] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 3987–3995.

[20] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[21] A. R. Triki, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, "Encoder based lifelong learning," *CoRR*, vol. abs/1704.01920, 2017.

[22] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *CoRR*, vol. abs/1606.04671, 2016. [Online]. Available: http://arxiv.org/abs/1606.04671

[23] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. [Online]. Available: https://openreview.net/forum?id=Sk7KsfW0-

[24] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," *arXiv preprint arXiv:1711.09325*, 2017.

[25] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[26] T. Chen, X. Zhai, M. Ritter, M. Lucic, and N. Houlsby, "Self-supervised gans via auxiliary rotation loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 154–12 163.

[27] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] J. Rajasegaran, M. Hayat, S. Khan, F. S. Khan, and L. Shao, "Random path selection for incremental learning," *Advances in Neural Information Processing Systems*, 2019.

[29] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[30] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[31] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[32] Y.-M. Chou, Y.-M. Chan, J.-H. Lee, C.-Y. Chiu, and C.-S. Chen, "Unifying and merging well-trained deep neural networks for inference stage," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 2049–2056. [Online]. Available: https://doi.org/10.24963/ijcai.2018/283

[33] B. Cheung, A. Terekhov, Y. Chen, P. Agrawal, and B. A. Olshausen, "Superposition of many models into one," *CoRR*, vol. abs/1902.05522, 2019. [Online]. Available: http://arxiv.org/abs/1902.05522

[34] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data classification: Algorithms and applications*, p. 37, 2014.

[35] Y. M. Asano, C. Rupprecht, and A. Vedaldi, "Surprising effectiveness of few-image unsupervised feature learning," *arXiv preprint arXiv:1904.13132*, 2019.

[36] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 6572–6583.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Workshop on Autodiff*, 2017.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

Appendix

Our implementation is done with Pytorch [37], using the Adam optimizer [38] and a scheduler that reduces the learning rate by a factor of 0.5 when the loss stops improving. We use the resize transformation from torchvision with the default bilinear interpolation.

TABLE III: MNIST Hyperparameters

Hyperparameter	**Value**
Learning Rate	0.002
Number of epochs	200
Weight decay	0.0
Patience	20

TABLE IV: CIFAR-100 Hyperparameters

Hyperparameter	**Value**
Learning Rate	0.002
Number of epochs	1000
Weight decay	0.0002
Patience	30

TABLE V: t-SNE Hyperparameters

Hyperparameter	**Value**
Perplexity	15.0
Principal Components	50
Steps	400