Generalization of iterative sampling in autoencoders

Miguel Solinas, Clovis Galiez, Romain Cohendet, Stéphane Rousset, Marina Reyboz, Martial Mermillod

Generalization of iterative sampling in autoencoders

This work has been submitted to IEEE ICMLA 2020

***Abstract*—Generative autoencoders are designed to model a target distribution with the aim of generating samples and it has also been shown that specific non-generative autoencoders (i.e. contractive and denoising autoencoders) can be turned into generative models using reinjections (i.e. iterative sampling). In this work, we provide mathematical evidence that any autoencoder reproducing the input data with a loss of information can sample from the training distribution using reinjections. More precisely, we prove that the property of modeling a given distribution and sampling from it not only applies to contractive and denoising autoencoders but also to all lossy autoencoders. In accordance with previous results, we emphasize that the reinjection sampling procedure in autoencoders improves the quality of the sampling. We experimentally illustrate the above property by generating synthetic data with non-generative autoencoders trained on standard datasets. We show that the learning curve of a classifier trained with synthetic data is similar to that of a classifier trained with original data.**

***Index Terms*—autoencoder, iterative sampling, unsupervised learning, neural networks**

I. Introduction

In machine learning, real-world data may not be accessible for various reasons (e.g. privacy, lack of data, memory footprint issues, etc.). In these situations, generative models can then be trained to model a given training distribution and be used to synthesize artificial data. To address this problem, Deep Boltzmann Machines [1], Generative Adversarial Nets [2] and Variational Autoencoders [3] are among the most popular deep generative models available in the literature and actively used. It has also been shown that specific non-generative autoencoders can be turned into generative models by simulating a Markov chain [4]. Autoencoders are convenient because they are conceptually simple and straightforward to train. Indeed, they simply minimize the loss over the input data and its replication. Furthermore, autoencoders make few assumptions about the learned distribution (i.e. they do not force a prior on the latent space) and can sample directly from the input space [4].

In this context, autoencoders might be divided into two main classes depending on their final objective. Within the first class, there is a growing family of autoencoders for which the final training objective is to generate synthetic data. State-of-the-art generative autoencoders include Variational Autoencoders (VAEs) [3] and Adversarial Autoencoders (AAEs) [5]. The second class comprises all the autoencoders whose purpose is not necessarily to generate data. This class includes, among others, contractive and sparse autoencoders that are trained to learn the manifold on which the data lies to improve the performance on classification tasks [6] or to detect anomalies [7]. Other popular models in this class are Denoising Autoencoders (DAEs) that restore the original distorted input [8].

There are two popular ways for autoencoders to sample from the learned distribution: latent variable sampling and iterative sampling. Latent variable sampling (also called ancestral sampling) [3], [5] produces samples from a probabilistic model by sampling a prior distribution and involves a single pass over the model parameters. On the other hand, iterative sampling consists in injecting an input vector over the entire state space (i.e. latent or input space) of an autoencoder and in reinjecting its output multiple times. It has been shown that this sampling procedure can iteratively improve the quality of the samples [4], [9]–[11]. Analogous to this iterative sampling, the term reinjection has originally been used (in 1997) to refer to the sampling of trained neural networks [12].

In this paper, we present a general theoretical framework for autoencoders that generalizes past theoretical results limited to contractive and denoising autoencoders to the whole family of lossy autoencoders (i.e autoencoders reproducing the input data with a certain loss of information). Consistent with the fact that reinjections in autoencoders can be used to generate synthetic data, we highlight that lossy autoencoders can model the learned distribution. We use the term reinjection to refer to the iterative sampling process in autoencoders.

Finally, to assess the sampling quality of reinjections in lossy autoencoders, we carry out the following experiment. In a first step, we train an autoencoder and a classifier (named ground truth classifier) on a labeled dataset. Using a reinjection procedure, we generate samples that are labeled using the ground truth classifier. In a second step, we train, with only synthetic data and synthetic labels, a new classifier with the same architecture as that of the ground truth classifier. Then, we compare the accuracy of the synthetically-trained classifier to that of the ground truth classifier.

Our key contributions can be summarized as follows:

- We provide mathematical evidence that any autoencoder reproducing the input data with a certain loss of information can model the training distribution.
- We provide mathematical evidence that lossy autoencoders can generate samples using a reinjection procedure.
- We assess the reinjection procedure using different datasets: MNIST, CIFAR-10 and CIFAR-100 and show that classifiers, when learning from synthetic data obtained by the use of the reinjection procedure, show a similar performance to that of classifiers learning from original data.

This work is structured as follows. Related works are presented in Section II. The theoretical framework of autoencoders is exposed in Section III. Next, the evaluation and validation of our experiments are presented in Section IV. We discuss our results in Section V. Finally, the conclusion and perspective are drawn in Section VI.

II. Related work

The reinjection procedure was proposed several years ago in a pseudo-rehearsal setting for incremental learning problems [12]–[15]. Instead of storing already seen real data in physical memory, reinjections were used to generate previously learned information in neural networks. When sequentially learning a new class, a model is trained using both new information (real samples and labels) and synthetic data (synthetic samples and labels) that are generated by the neural network memory. The synthetic samples are labeled by a classifier whose input is the latent representation of the neural network memory. In this way, the predictive function of the classifier trained on the learned classes is transferred to the current learning model. It has then been shown that the application of a reinjection procedure, in such a context, improves not only the data generation but also the knowledge transfer performance [12].

Along the last decade, many works have been carried out about specific classes of non-generative autoencoders that can automatically model the training distribution and generate synthetic data. These autoencoders are; Contractive Autoencoders (CAEs) [6], [16]–[18] and Denoising Autoencoders (DAEs) [4], [9], [19]. These works formalized a sampling procedure that simulates a Markov chain to sample from the training distribution. It has been also demonstrated for DAEs that, after each reinjection, the reconstruction function corresponds, at first order, to a small displacement towards higher densities in the training distribution. For DAEs, this reinjection procedure, is equivalent to a Langevin sampling of the training set [4].

Several recent works [9], [20]–[22] have shown that the quality of synthetic samples can be improved through reinjections in generative autoencoders.

We seek to generalize the property of modeling a given distribution and generate samples from it using reinjections to a larger family of autoencoders: autoencoders with a loss of information. We show that lossy autoencoders can sample from the learned distribution using reinjections and that the generated examples are more useful than random samples.

III. General framework for autoencoders

Informally, it is considered that the purpose of autoencoders is to reproduce as close as possible a random variable of $\mathbb{R}^n$, which motivates the following definition:

Definition III.1 (Autoencoder)**.** An autoencoder is a function $r \in \mathcal{M}$ where $\mathcal{M}$ is a subset of the functions $\mathbb{R}^n \to \mathbb{R}^n$ called the universe of *models*.

Definition III.2 (Perfect autoencoder)**.** We say that an autoencoder r is *perfect* with respect to a set of models $\mathcal{M}$, a probability density p on $\mathbb{R}^n$ and a loss function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ if r minimizes over $\mathcal{M}$ the expected loss $\mathbb{E}_{X\sim p}\left[\mathcal{L}(X, r(X))\right]$.

Example III.1 (Perfect autoencoder in $\mathbb{R}^n$-diffeomorphisms universe)**.** Let us consider that the universe of models is the set of $\mathbb{R}^n$-diffeomorphisms and that we define the loss of reproduction $r(x)$ of x as: $L_2(x, r(x)) = ||x - r(x)||_2^2$.

From the previous definitions, a perfect autoencoder of a random variable X having a density p is a function of $r \in \mathcal{M}$ minimizing the expected loss:

$$\mathcal{L}_2(r) = \mathbb{E}_{X\sim p}\left[||X - r(X)||_2^2\right] \tag{1}$$

Trivially, the identity function is a perfect autoencoder for $\mathcal{M}$.

Our ultimate goal would be to describe analytically the perfect autoencoder for general neural networks as the universe of models. As it seems an unreachable goal, in general (although this work has been achieved for simple models, *e.g.* see [23] with multi-layer perceptron), we will consider a larger universe of models consisting of the diffeomorphisms of $\mathbb{R}^n$ showing limited mutual information with the input data:

Definition III.3 (Lossy autoencoders)**.** We define the universe $\mathcal{M}_\tau$ of models of autoencoders of a random variable $X \sim p$ (with $\forall x \in \mathbb{R}^n, p(x) \neq 0$) as the set of diffeomorphisms r of $\mathbb{R}^n$ with limited mutual information, *i.e.* such that:

$$MI(X, r(X)) < \tau \tag{2}$$

where MI is the mutual information defined by:

$$MI(X, r(X)) = \int_{\mathbb{R}^n} p(x, r(x)).\log\frac{p(x, r(x))}{p(x).p(r(x))}dx \tag{3}$$

Although this definition seems to focus on a specific type of autoencoder, its aim is rather to provide a general universe for real-world autoencoders that are basically machine learning models learned on finite data and are likely to lose information between the input and the output. The limitation of the mutual information to a value of τ is intended to model the inability of an empirically trained autoencoder to reproduce perfectly the input data. Note that τ must be lower than the entropy $H(X)$ of the random variable X in order the limitation to be effective (for higher values, the inequality is trivial since r is a function and $MI(X, r(X)) = H(r(X)) \leq H(X)$).

Proposition III.1. A perfect lossy autoencoder of a random variable X with density $p(x) \neq 0$ is a diffeomorphism r of $\mathbb{R}^n$ minimizing the following expected loss:

$$\mathcal{L}_\lambda(r) = \mathbb{E}_{X\sim p}\left[||x - r(x)||_2^2 + \lambda \log|\mathscr{J}r|\right] \tag{4}$$

for some $\lambda > 0$, where $|\mathscr{J}r|$ being the determinant of the Jacobian matrix $(\mathscr{J}r)_{ij} = \frac{\partial r_i}{\partial x_j}$.

The regularization parameter λ corresponds to a Langrange multiplier (see suppl. mat.) of the condition $MI(X, r(X)) < \tau$, so that τ and λ are anti-correlated.

It is worth noticing that when the autoencoder has a weak constraint on its mutual information ($\tau \to H(X)$) the minimized loss is weakly regularized ($\lambda \to 0$).

In the sequel of this section, we provide insights on the behavior of perfect lossy autoencoders, essentially following the derivations of [19] for contractive autoencoders.

We conjecture that the behavior of neural network autoencoders trained over a dataset sampled from a distribution does not differ significantly from the behavior of perfect lossy autoencoders, for some λ that depends on both the data and the structure of the neural networks.

The local minima of the loss $\mathcal{L}_\lambda(r)$ are described by their first order expansion:

Proposition III.2. The first order term in the expansion with respect to λ of a minimizer of the loss defined in equation 4 is $\frac{1}{2}\frac{\partial \log p}{\partial x_i}$, and a perfect lossy autoencoder satisfies:

$$r_i(x) = x_i + \frac{\lambda}{2}\frac{\partial \log p}{\partial x_i} + o(\lambda)\ as\ \lambda \to 0 \tag{5}$$

Intuitively, this means that when the autoencoder model is expressive enough, e.g. the mutual information between the input and the output is high enough, the regularization coefficient λ is low and the perfect lossy autoencoder tends to reproduce the input with a small shift towards higher density of the input data.

A. Sampling Procedure

We consider the Euler-Murayama discretization [24] simulating the Langevin dynamics through the definition of the following Markov chain:

$$\begin{aligned} x_{n+1} &= x_n + \frac{\lambda}{2}\nabla \log p + \sqrt{\lambda}.\epsilon_n \\ &where\ \epsilon_n \sim \mathcal{N}(0, I) \end{aligned} \tag{6}$$

If r is a perfect lossy autoencoder with sufficient expressiveness (e.g. having a low regularization coefficient, at the limit $\lambda \to 0$), by proposition (III.2), the above equation can be approximated (first order in λ) by:

$$x_{n+1} \approx r(x_n) + \sqrt{\lambda}.\epsilon_n \tag{7}$$

Equation (7) defines a Markov chain that samples from the original distribution of density p (see suppl. mat. of [11]). Moreover, when the original data can be approximated by a mixture of Gaussians, the fast mixing can be guaranteed (see [25] for details, in particular Theroem 3.1).

Finally, we obtain the following algorithm that at first order samples from the original distribution without requiring access to the original probability distribution. The Figure 1 illustrates the reinjection procedure of the Algorithm 1 that exploits the information contained in a perfect lossy autoencoder fitted on the original data.

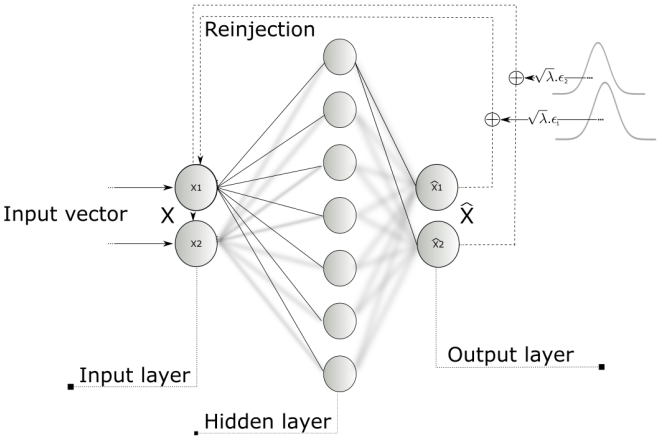

Fig. 1. Autoencoder sampling procedure

Algorithm 1 Approximate Langevin sampling with perfect lossy autoencoders

INPUT:

- AE (Perfect lossy autoencoder)
- N_s (Number of samples to be generated)
- λ (Regularization coefficient)

OUTPUT:

- X: (x_1,x_2,x_3...)

$x_n \sim \mathcal{N}(0, I)$
for $n = 0$ to N_s **do**
  Draw $\epsilon_n \sim \mathcal{N}(0, I)$
  $x_{n+1} = AE(x_n) + \sqrt{\lambda}.\epsilon_n$
end for

In real world applications, we train a neural network autoencoder on the original data where the regularization coefficient is unknown. Its value depends on the input data and on the structure of the neural network. One can infer a value of λ by optimizing it to minimize a measure of divergence between the original samples and the synthetic ones generated by the autoencoder. Let us notice that, if the inferred $\widetilde{\lambda}$ differs from the real λ, Algorithm 2 will generate samples from the distribution with a density proportional to:

$$p_{sampled}(x) \propto p(x)^{\lambda/\widetilde{\lambda}} \tag{8}$$

IV. Methodology

Let us assume now that the original data is a balanced finite mixture of random variables whose densities meet the requirement of the Theorem 3.1 [25]. By Proposition (III.2), the Algorithm 1 samples asymptotically the data distribution as $\lambda \to 0$ and $N_s \to \infty$ (see suppl. mat. of [11]). Then, provided with a perfect lossy autoencoder and a universal classifier indicating the class $\{0, 1, 2, ..., N_c\}$ of the input, we define the sampling procedure in the Algorithm 1 that equally samples from the different modes (i.e. classes).

Algorithm 2 Approximate Langevin sampling of multiclass labeled data with perfect lossy autoencoders

Input:

- AE (Perfect lossy autoencoder)
- C (Trained classifier)
- N_s (Number of samples)
- λ (Regularization coefficient)

Output:

- X = [ ] (Synthetic samples)
- Y = [ ](Synthetic labels)

$x_n \sim \mathcal{N}(0, I)$
for $n = 0$ to N_s **do**
  Draw $\epsilon_n \sim \mathcal{N}(0, I)$
  $x_{n+1} = AE(x_n) + \sqrt{\lambda}.\epsilon_n$
  $X = X \cup [x_n]$
  $Y = Y \cup [C.predict(x_n)]$
end for

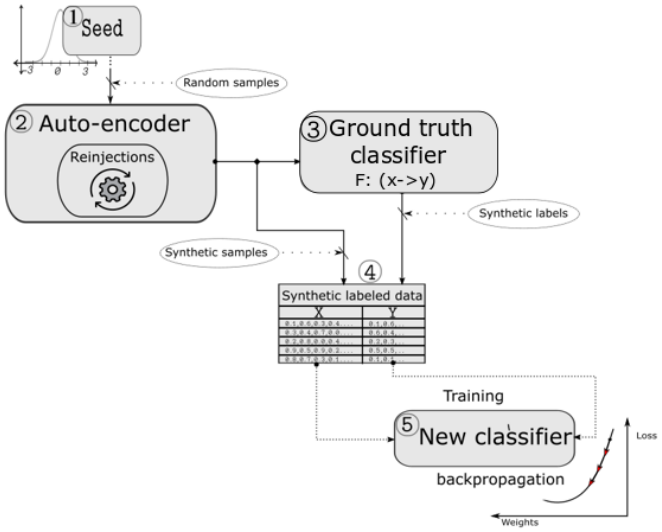

Fig. 2. Experimental workflow

A. Heuristic

In theory, the Sampling Procedure III-A generates samples from the data-generating distribution p, mixing between modes. In practice, the Markov Chain behind the Sampling Procedure III-A can get "trapped" in a small region of the input space (e.g. a high-probability region) before transitioning to another region leading to highly correlated samples of some modes. The overrepresentation of some modes can be due to spurious attractors with sharp density peaks [18] or when several regions are connected only by low-probability transition [9]. This behavior is referred to as slow mixing [26] and it has already been observed in previous works [4], [18]. We have also noticed that depending on how the autoencoders are trained, they will capture some modes better than others. To cope with this issue, we have implemented an heuristic (see Algorithm 1 in supplementary materials) to facilitate the mixing between modes of the training distribution. By stopping the random walk to select a new starting point, we can enhance the mixing between high density peaks (e.g. in non-smooth data distribution cases) and we can prevent the sampling procedure from oversampling the same region. Inspired by previous research [12], [25] and by our experimental results, this heuristic allows us to alleviate the oversampling of some modes.

In our experiments, fixed values for λ and R were used (see supplementary materials). Potentially, a careful optimization of these parameters could lead to improved results, a study that is out of the scope of this present paper. It is worth noticing that we start gathering synthetic data after the second jump (see the condition i > 1 of Algorithm 1 in supplementary materials) in order to minimize the importance of the original starting point. Discarding iterations at the beginning of a Markov chain is known as "burn-in" in the sampling community.

This heuristic can be used in the experimental workflow which generates synthetic labeled data from trained autoencoders and a trained classifier. The workflow described in Figure 2 allows us to make use of this sampling procedure to train neural network classifiers.

B. Experimental Workflow

Initially, we need to provide a trained autoencoder, the regularization coefficient (λ), a trained classifier (i.e. the ground truth classifier) and the amount of required labeled samples *Ns*. Then, we initialize ① the random seed (e.g., isotropic Gaussian noise $\mathcal{N}(0, I)$) for the reinjections of the heuristic. The trained autoencoder, using the random samples, performs multiple reinjections to generate synthetic unlabeled data ②. These data are labeled using the ground truth classifier ③. Finally, the resulting labeled synthetic data ④ is used as a training set for training a new classifier ⑤ (the classifier under training). It should be noticed that the choice of the starting point is arbitrary. In this work, the starting points are sampled from an isotropic Gaussian distribution $\mathcal{N}(0, I)$ because it is a good prior when no information is available from the real distribution.

C. Evaluation and Validation

In general, the parameter N_s in Algorithm 2 is equivalent to the amount of labeled synthetic data required to meet the accuracy of the trained classifier. To illustrate the quality of samples generated by the autoencoders through the reinjection procedure, we show the number of samples that a new classifier requires to meet the accuracy of the ground truth classifier. The lower the number of required labeled samples to yield a satisfying accuracy, the better autoencoders generate samples from the learned distribution.

In order to evaluate and illustrate the general property presented in the previous section, we have decided to implement shallow and deep non-generative autoencoders that are trained on three datasets as case studies: two DAEs and two classic autoencoders (AEs). We have based our selection on three

criteria. First, we decided to implement the DAEs because of their property for generating samples from the learned distribution, which has already been exploited in previous research [4], [11], [19], [27]. Second, we have decided to implement two classic autoencoders to illustrate their general property for capturing the learned data distribution, which goes beyond denoising implementations. Third, we have implemented AEs and DAEs because they can be sampled from the input space.

In this section, we present four autoencoders under test and the three datasets on which they are trained. Also, we analyze the amount of synthetic data as well as the learning speed when learning from the sampling procedure.

D. Autoencoders under test

The best suited model scenario for resource frugal applications would be shallow models that allow good sampling; however, deeper models (with higher-level representations) could help to mix the high-density peaks of the datasets [18]. Since shallow models may be less expressive than deeper models, we expect to see a slow mixing between modes or even a lower performance when sampling from shallow models. We can understand this behavior by using our general framework for autoencoders: the more expressive a model becomes, the higher the mutual information and, in consequence, the smaller the regularization coefficient lambda.

As deeper models (deeper representations) can mix faster between modes than shallow models [18], we study both in order to illustrate their performance. The autoencoders under test consist of two shallow autoencoders and two deep autoencoders for each dataset, as detailed in Table I. All shallow autoencoders have tanh hidden units whereas deep autoencoders have relu activation hidden units. We use the tanh activation function because it improves the expressiveness of shallow autoencoders as it was shown in [4], [9]. However, we use the relu activation function in deep models because it works well in practice and converges faster to valid solutions. Depending on the datasets, the number of neurons per hidden layer varies between 32 and 1000 as it is specified in Table I.

Denoising autoencoders have pre-activation Gaussian noise and salt-and-pepper noise of mean and standard deviation as defined in Table I. The pre-activation Gaussian noise is applied to the first layer. Also, the true dataset is corrupted with salt-and-pepper noise depending on the datasets (e.g. for MNIST, the pixels are corrupted and replaced with some value between 0 and 1 with probability 0.5). We performed a grid search over the pre-activation Gaussian noise and the salt-and-pepper noise because DAEs are not capable of learning if the standard deviation of the added noise is too large. The result of the grid search is visualized in Table I. When training the DAEs on the datasets, we have observed that MNIST allows for a more significant amount of noise than CIFAR. It turns out that CIFAR features are more sensitive to random variations.

Training is performed over 200 epochs at most depending on the models and the datasets as detailed in Table I. In general, good results are obtained after around 50 epochs for shallow AEs and 150 epochs for deep AEs. The learning rate of 0.001 and the Adam optimizer (beta_1=0.9, beta_2=0.999) are selected to minimize the negative reconstruction log-likelihood. Finally, the batch size to train the ground truth classifier, which delivers the labels for synthetic samples, is set to 512, whereas the number of epochs is 50 for all the datasets as detailed in Table I.

It should be noticed that both the mean squared error and the binary cross-entropy loss can be used to train the autoencoders. In Section III, proposition 4 focuses on the MSE loss because it allows us to simplify mathematical treatment. However, we use the binary cross-entropy for our experiments in order to be consistent with previous results [4], [19], [27].

E. Dataset for Validation

We benchmark the sampling procedure on three standard datasets that differ in the number of classes and features. First, we study the raw images from MNIST [28]. Then, we extract the features from CIFAR-10 [29] and CIFAR-100 [29] using the resnet50 [30] preprocess input Keras implementation [31]. It is worth noticing that the maximum prediction accuracy rate after feature extraction of CIFAR-10 and CIFAR-100 datasets is around 92% and 75% respectively. Also, we scale the extracted features between 0 and 1 using the min-max normalization.

By using these three different datasets that vary in the number of features and classes, we expect to evaluate the impact of the reinjection sampling procedure on the experimental workflow of Figure 2.

F. Metrics

All of our experiments follow the experimental workflow in Figure 2. As input for the workflow, we first provide the label assigner, which is the ground truth classifier, and the four trained auto-encoders under test, trained using the hyper-parameters of Table I.

Our primary metric is the accuracy of the ground truth classifier on the test set from now on, referred to as *max accuracy*. To analyze our results, we have plotted the in-training accuracy of four classifiers that measure the sampling performance of the autoencoders under test, from now on, we call them *AE classifiers* and the accuracy of two classifiers as performance bound references (the six classifiers have the same architecture as the ground truth classifier). The upper bound reference is the accuracy of a classifier under training when learning from the original data, which should be similar to the max accuracy of the ground truth classifier when learning from the original data. Hopefully, a classifier trained with synthetic data generated using a sufficiently expressive autoencoder will yield similar performance as to this upper bound reference. When no information on the training distribution is available, a learning procedure can be to label random examples coming from a fixed distribution. Referring to the workflow in Figure 2, this corresponds to bypassing the autoencoder. Thus, the lower bound reference is the accuracy of a classifier when learning from random samples, here obtained using an isotropic

TABLE I
MODEL PARAMETERS AND RESULTS

Models	#units per hidden layer	activation function	epochs	noise	acc	acc: classifiers
MNIST						
Random generator (lower bound)	-	-	-	-	-	0.925
Deep AE	[600,400,100,400,600]	[relu]	200	-	-	0.978
Shallow AE	[32]	[tanh]	50			0.975
Deep DAE	[600,400,100,400,600]	[relu]	200	N(0,0.5)		0.978
Shallow DAE	[32]	[tanh]	100	N(0,0.5)		0.965
Ground truth classifier	[500,500]	[relu]	20	-	0.98	-
CIFAR-10						
Random generator (lower bound)	-	-	-	-	-	0.892
Deep AE	[1000,800,400,800,1000]	[relu]	200	-	-	0.915
Shallow AE	[400]	[tanh]	100			0.906
Deep DAE	[1000,800,400,800,1000]	[relu]	200	N(0,0.05)		0.915
Shallow DAE	[400]	[tanh]	100	N(0,0.05)		0.911
Ground truth classifier	[500,500]	[relu]	50	-	0.92	-
CIFAR-100						
Random generator (lower bound)	-	-	-	-	-	0.643
Deep AE	[1000,800,400,800,1000]	[relu]	200	-	-	0.749
Shallow AE	[400]	[tanh]	200			0.731
Deep DAE	[1000,800,400,800,1000]	[relu]	200	N(0,0.05)		0.749
Shallow DAE	[400]	[tanh]	200	N(0,0.05)		0.730
Ground truth classifier	[1000]	[relu]	50	-	0.75	-

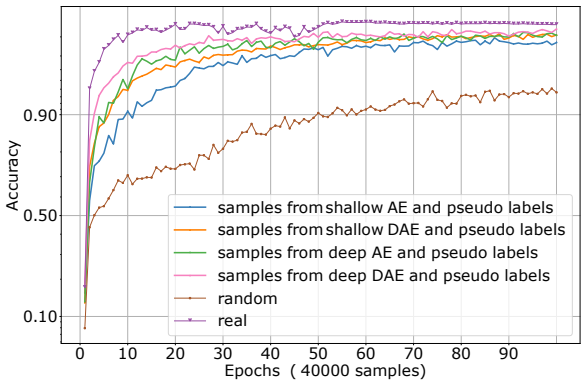

Fig. 3. Classifiers accuracy on MNIST : Average performance through epochs.

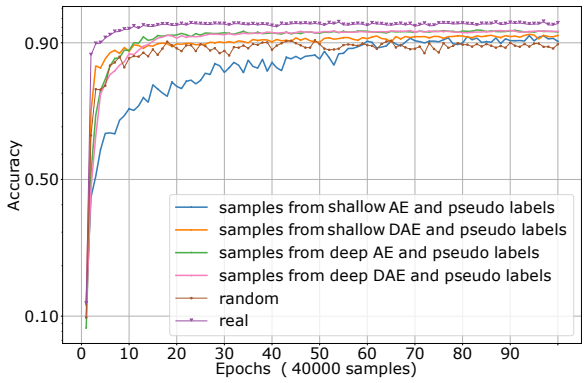

Fig. 4. Classifiers accuracy on CIFAR-10 : Average performance through epochs.

Gaussian distribution $\mathcal{N}(0, I)$ and their labels (e.g., (X,Y) = ([random_sample, C.predict(random_sample)])).

Since synthetic samples that come from autoencoders or from random sampling may not belong sharply to a particular class, we use a logit/distillation method. Logit/distillation is a common practice to assign a probability of all the classes instead of a discrete class. The relative probabilities indicate how a model tends to generalize and helps to transfer the generalization ability of a trained model to a new model. The idea was developed for sequential learning problems in [12], [32] and adopted by the model compression community [33]–[35]. In this way, we can generate pseudo-labels which improve class representations. It should be noticed that using logit/distillation implies training the classifiers to output the logits. For a matter of simplicity, we output the logits of the classes by using the binary-cross entropy loss to train the classifiers.

G. Result Analysis

We measure the accuracy over the learning steps, which is calculated on the real test dataset, to illustrate the sampling performance. The results of the experiments, using the model architectures described in Table I, are summarized in three plots. Figures 3, 4 and 5 show the in-training classifiers accuracy curve on the three datasets respectively: MNIST, CIFAR-10 and CIFAR-100. The learning curve of the classifiers gives a rapid overview of the performance of the data generators. The axis Y indicates the accuracy in the Logit scale. The axis X indicates the training epoch; each one has 40,000 samples. This value results from the multiplication between the number

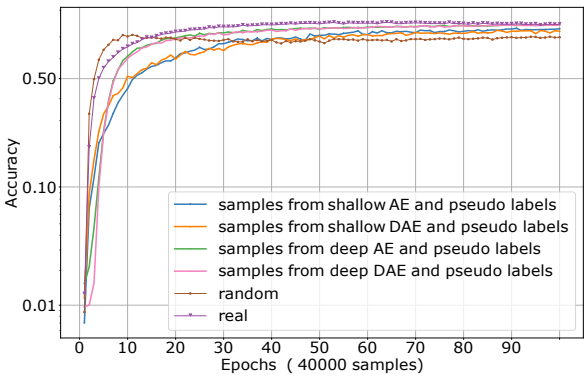

Fig. 5. Classifiers accuracy on CIFAR-100 : Average performance through epochs.

of reinjections (8), the synthetic data mini-batch (50) and the learning steps carried out (100). In other words, every 100 learning steps, we measure the accuracy of *AE classifiers* (40,000 = 8*50*100). Overall, the accuracy of the classifier under training serves as a quality measure of the reinjection procedure. In the following, we analyze the results for the three datasets.

a) MNIST: In Figure 3, the learning curve of the classifiers, when learning from deep models, has a similar slope to the one obtained when learning from the real dataset. As expected, deeper autoencoders sample better from the real distribution than shallow autoencoders. Also, the accuracy of the *AE classifiers* is between the upper-lower bounds during the entire training process. Furthermore, the results show that the autoencoder reinjection procedure is sampling from the learned distribution while showing a substantial performance gain compared to the zero reinjection scenario (random samples and their pseudo labels). No matter which deep autoencoder the *AE classifiers* learn from, they show an accuracy that is above 97% relative to max accuracy after 50 epochs. They roughly require the same amount of samples which means a similar learning speed.

b) CIFAR-10: In Figure 4, all classifiers have a similar learning curve. At the end of the training stage, the accuracy of the *AE classifiers* outperforms the zero reinjection scenario (lower bound reference). Since the features of the CIFAR 10 dataset can be modeled by a Gaussian distribution, the gain of sampling from the autoencoders may appear minor. As a consequence, a classifier, when learning from random samples, learns faster than when learning from the autoencoders. That is, the amount of random samples used is enough to sample most of the dataset modes. We provide additional analysis of this behavior in Section V. The *AE classifiers* require more learning stages and synthetic data than the upper-bound classifiers. Also in Figure 4, it is possible to see a poorly trained shallow autoencoder.

c) CIFAR-100: In Figure 5, the accuracy of the *AE classifiers* is between the upper and lower bound reference. Moreover, the reinjection procedure is sampling from its learning distribution while it is also delivering a remarkable gain of performance compared to a zero reinjection scenario. In a similar fashion to the previous plot, classifiers, when learning from random samples, learn faster than when learning from the autoencoders. However, this behavior reaches a maximum of 69% of accuracy and then it decreases slowly. Meanwhile, the *AE classifiers'* accuracy steadily increases until the *AE classifiers* reach the *max accuracy*. The *AE classifiers*, which learn from deep autoencoders, meet the *max accuracy* 75% at the end of the training stage. The classifiers, which learn from shallow autoencoders, reach an accuracy of 73% at the end of the training stage.

V. Discussion

a) Performance analysis: As previously shown in Table I, classifiers, when learning from autoencoders, may deliver a lower accuracy than when learning from the real dataset. We have systematically observed a loss of accuracy between 0.15% and 2% that depends on the autoencoder undersampling and the datasets. These observations are consequent to the extra training time required to meet the *max accuracy*. We explain these results based on two observations. First, autoencoders may show a slow mixing between modes, which means that consecutive samples tend to be correlated, resulting in a reduced representative set of the training distribution. In these cases, we incur a significant burden because we need to do further iterations to obtain a balanced set of samples. We note that mixing between modes can be a critical problem when sampling autoencoders. Second, trained autoencoders do not always sample from the real distribution when they have spurious attractors or are not expressive enough. Spurious attractors are mostly found in autoencoders due to inadequate training [18]. There is a probability spread over a broader region of the input space, as a consequence, the sampling procedure visits the region that does not belong to the trained distribution. The noise used to train denoising autoencoders makes DAEs more resilient to local spurious attractors [19].

b) Optimization: In general, the autoencoder design (e.g. activation function, the number of hidden units per hidden layers, optimizer, etc.) determines how the model shapes the training distribution. In our view, the value of λ and R could be considered as hyper-parameters to tune the sampling procedure. Furthermore, the differences observed in the reinjection procedure can be intuitively interpreted in terms of the strength of the regularization coefficient. In final implementations, the sampling performance of an autoencoder will be strictly related to this coefficient.

c) Datasets: The differences observed between the AE classifiers and the lower bound reference during training can be explained by observing the training distributions. In CIFAR datasets, particularly, the resulting extracted features can be well modeled by a normal distribution. In consequence, during the first training steps, the use of autoencoders delivers a gain that is equivalent to the gain obtained when using random samples generated from an appropriate distribution (e.g. a

normal distribution). However, the gain obtained when using random samples is limited and it is always outperformed when using samples from autoencoders. The MNIST dataset follows a more particular feature structure that cannot be modeled by a normal distribution (see PCA in suppl. mat.). In this case, the gain when using the autoencoders is evident.

VI. Conclusion

Previous work described a reinjection procedure for specific autoencoders (namely, contractive and denoising autoencoders [4], [16], [19]) allowing to sample the data-generating distribution. We provide a generalization of these results to any autoencoder trained on empirical data (reproducing the input data with a certain loss of information) and we experimentally illustrate that the sampling obtained is indeed good enough to enable its usage in real applications. In particular, we show that the performance of a neural network classifier trained with synthetic data generated by classical autoencoders is similar to that of a classifier trained with original data.

The theoretical framework presented is very general; however, it has the disadvantage of including a theoretical parameter (the regularization coefficient λ) influencing the quality of the sampling that cannot be straightforwardly estimated. In future works, we would like to provide guidelines and ideally a measure that predicts the sampling quality of an autoencoder given its architecture and its loss. We would also like to investigate a more diverse set of autoencoders such as convolutional autoencoders and other possible applications of this sampling procedure in the context of incremental learning settings as a solution to overcome catastrophic forgetting.

References

[1] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial intelligence and statistics*, 2009, pp. 448–455.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[4] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Advances in neural information processing systems*, 2013, pp. 899–907.

[5] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[6] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Icml*, 2011.

[7] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, and X. Chen, "A sparse auto-encoder-based deep neural network approach for induction motor faults classification," *Measurement*, vol. 89, pp. 171–178, 2016.

[8] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[9] G. Alain, Y. Bengio, L. Yao, J. Yosinski, E. Thibodeau-Laufer, S. Zhang, and P. Vincent, "Gsns: generative stochastic networks," *Information and Inference: A Journal of the IMA*, vol. 5, no. 2, pp. 210–249, 2016.

[10] K. Arulkumaran, A. Creswell, and A. A. Bharath, "Improving sampling from generative autoencoders with markov chains," *arXiv preprint arXiv:1610.09296*, vol. 3, p. 13, 2016.

[11] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug n play generative networks: Conditional iterative generation of images in latent space," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[12] B. Ans and S. Rousset, "Avoiding catastrophic forgetting by coupling two reverberating neural networks," *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, vol. 320, no. 12, pp. 989–997, 1997.

[13] A. Bernard and R. Stephane, "Neural networks with a self-refreshing memory: knowledge transfer in sequential learning tasks without catastrophic forgetting," *Connection science*, vol. 12, no. 1, pp. 1–19, 2000.

[14] B. Ans, S. Rousset, R. M. French, and S. Musca, "Self-refreshing memory in artificial neural networks: Learning temporal sequences without catastrophic forgetting," *Connection Science*, vol. 16, no. 2, pp. 71–99, 2004.

[15] S. C. Musca, S. Rousset, and B. Ans, "Artificial neural networks whispering to the brain: nonlinear system attractors induce familiarity with never seen items," *Connection Science*, vol. 21, no. 4, pp. 359–377, 2009.

[16] S. Rifai, Y. Bengio, Y. Dauphin, and P. Vincent, "A generative process for sampling contractive auto-encoders," *arXiv preprint arXiv:1206.6434*, 2012.

[17] Y. Bengio, G. Alain, and S. Rifai, "Implicit density estimation by local moment matching to sample from auto-encoders," *arXiv preprint arXiv:1207.0057*, 2012.

[18] B. Yoshua, M. Grégoire, D. Yann, and R. Salah, "Better mixing via deep representations," *International conference on machine learning*, pp. 552–560, 2013.

[19] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3563–3593, 2014.

[20] A. Creswell, K. Arulkumaran, and A. A. Bharath, "Improving sampling from generative autoencoders with markov chains," *arXiv preprint arXiv:1610.09296*, 2016.

[21] C. Antonia, A. Kai, and B. A. A, "On denoising autoencoders trained to minimise binary cross-entropy," *arXiv preprint arXiv:1708.08487*, 2017.

[22] A. Creswell and A. A. Bharath, "Denoising adversarial autoencoders," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 4, pp. 968–984, 2018.

[23] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[24] G. Maruyama, "On the transition probability functions of the markov process," *Nat. Sci. Rep. Ochanomizu Univ.*, vol. 5, pp. 10–20, 1954.

[25] R. Ge, H. Lee, and A. Risteski, "Beyond log-concavity: Provable guarantees for sampling multi-modal distributions using simulated tempering langevin monte carlo," *CoRR*, vol. abs/1710.02736, 2017.

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

[27] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," *International Conference on Machine Learning*, pp. 226–234, 2014.

[28] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

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

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[31] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[32] J. Wen, Y. Cao, and R. Huang, "Few-shot self reminder to overcome catastrophic forgetting," *arXiv preprint arXiv:1812.00543*, 2018.

[33] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

[34] J. Ba and R. Caruana, "Do deep nets really need to be deep?" *Advances in neural information processing systems*, pp. 2654–2662, 2014.

[35] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.