Expert Systems with Applications 161 (2020) 113696

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Improving classification accuracy using data augmentation on small data sets

Francisco J. Moreno-Barea, José M. Jerez, Leonardo Franco *

Departamento de Lenguajes y Ciencias de la Computación, Escuela Técnica Superior de Ingeniería Informática, Universidad de Málaga, No.35, Bulevar Louis Pasteur, Málaga, Spain

ARTICLE INFO

Article history:
Received 17 October 2019
Revised 23 June 2020
Accepted 24 June 2020
Available online 15 July 2020

Keywords:
Deep Learning
Data augmentation
GAN
VAE
Unbalanced sets

ABSTRACT

Data augmentation (DA) is a key element in the success of Deep Learning (DL) models, as its use can lead to better prediction accuracy values when large size data sets are used. DA was not very much used with earlier neural network models before 2012, and the reason might be related to the type of models and the size of the data sets used. We investigate in this work, applying several state-of-the-art models based on Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), the effect of DA when using small size data sets, analyzing the results in terms of the prediction accuracy obtained according to the different characteristics of the training samples (number of instances and features, and class unbalance degree). We further introduce modifications to the standard methods used to generate the synthetic samples to alter the class balance representation, and the overall results indicate that with some computational effort a significant increase in prediction accuracy can be obtained when small data sets are considered.

1. Introduction

Since the results obtained in the ImageNet competition in 2012 by Krizhevsky, Sutskever, and Hinton (2012), Deep learning models (DL) have caused a revolution in the field of machine learning that is still impacting in almost every application area as the industry is investing billions of dollars in artificial intelligence techniques. Deep learning has been extremely successful in image recognition problems, voice recognition and temporal series, in particular when using large size data sets (LeCun, Bengio, & Hinton, 2015; Schmidhuber, 2015). Compared to other classic supervised machine learning methods (FFNN, SVMs, Random Forests) deep learning has a more complex model structure, basically including earlier layers that perform feature extraction, and because deep learning architectures include thousands of parameters (mainly synaptic weights) which must be adjusted during training, a large data set with values of thousands of instances for each output category is generally needed to achieve a good level of success (Goodfellow, Bengio, & Courville, 2016). Data augmentation (DA), that refers to the creation of additional synthetic samples, has become an important area of research in recent years, as most of the models that won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) were trained with data augmentation techniques (He, Zhang, Ren, & Sun, 2016; Simonyan & Zisserman, 2014; Szegedy et al., 2015).

Most successful examples of DA are found in the application to data sets consisting of images, cases in which the original patterns can be transformed by applying geometric modifications, like rotation, enlargement, translation, contrast variation, etc., that do not alter the category of the samples (Krizhevsky et al., 2012). Applying DA to non-image data sets is more challenging and apart from some applications of SMOTE techniques (synthetic minority oversampling technique) (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), designed to deal with data sets that have unbalanced classes, the rest of the existing literature previously to 2015 is mainly related to noise injection techniques as a way to prevent overfitting and improve the accuracy of predictions (Fernández-Navarro, Hervás-Martínez, & Gutiérrez, 2011; Moreno-Barea, Strazzera, Jerez, Urda, & Franco, 2018; Piotrowski & Napiorkowski, 2013; Reed & Marks, 1998; Zur, Jiang, Pesce, & Drukker, 2009). A recent work related also to DA is the work by Formentin, Mazzoleni, Scandella, and Previdi (2019) where they applied it to unsupervised non-linear system identification.

The use and application of DL models involves several other techniques that have grown at the same time, and among them Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Kingma & Welling, 2013; Radford, Metz, & Chintala, 2015; Rezende, Mohamed, & Wierstra, 2014;

* Corresponding author.
E-mail addresses: fjmoreno@lcc.uma.es (F.J. Moreno-Barea), jmjerez@uma.es (J.M. Jerez), lfranco@lcc.uma.es (L. Franco).

https://doi.org/10.1016/j.eswa.2020.113696
0957-4174/

Zhao, Mathieu, & LeCun, 2016) have attracted much interest in recent times. These techniques are included within the so-called deep generative network models, whose purpose is to learn the data distribution from a set of samples, to further generate new samples drawn from the learned distribution. Although they share purpose, both techniques are based on different ideas. VAEs are based on autoencoders networks (Bourlard & Kamp, 1988), and try to codify the input data capturing the original distribution; while GANs are based on the idea of producing a confrontation between two networks, where one captures the distribution of the real data and generates synthetic data, and tries to deceive a second network that has to discern whether the data it receives is real or false. Both models can be applied also in several tasks, but VAEs and GANs have become very popular in recent times due to their success in generating fake images that can be considered real by experts (Karras, Aila, Laine, & Lehtinen, 2017; Pu et al., 2016; Radford et al., 2015). They can also be applied in other tasks like image resolution enhancement (Ledig et al., 2017), image translation (Isola, Zhu, Zhou, & Efros, 2017), and speech and natural language processing (Hsu, Hwang, Wu, Tsao, & Wang, 2017; Pascual, Bonafonte, & Serrà, 2017); but in particular they have shown to be very useful for DA (Douzas & Bacao, 2018; Frid-Adar et al., 2018; Hsu, Zhang, & Glass, 2017).

Considering all aspects mentioned before, two are the main objectives of this work: on one hand the current investigation tries to fill a gap in the existing scientific literature related to the use of DA on small size data sets. On the other hand we analyze and propose modifications to state-of-the-art DA methods in order to obtain an increase on the prediction accuracy by modifying the percentage of synthetic generated samples for the different existing categories in the data sets, that will permit to apply efficiently DA techniques to small and non-structured data sets in several domains.

2. Previous related works

We review in this Section recent works about the application of VAE and GAN models in the field of Data Augmentation (DA). Within the medical domain, where the available samples are usually scarce and expensive to obtain, we can mention works that develop a DA process using GANs: Wu, Wu, Cox, and Lotter (2018) applied these class of methods to breast cancer tomography, Frid-Adar et al. (2018) analyzed liver lesion scans, while Zhao et al. (2018) applied a new generator model called F&BGAN (Forward & Backward GAN) obtaining interesting results in its application to lung cancer images.

DA tasks have also been successfully applied to other domains, such as language processing tasks and automatic voice recognition (Hsu et al., 2017); digital signal modulation classification (Tang, Tu, Zhang, & Lin, 2018); and prediction in time series, specifically in machine fault diagnosis (Shao, Wang, & Yan, 2019).

Although the previous works show a remarkable success in the creation of synthetic data and their application in the improvement of classification tasks thanks to the DA, especially in problems whose domain is data with spatial or temporal information (images, signals and time series), the application of deep generative models in other domains has not been as successful and it is more difficult to find related works. We can mention the work of Liu et al. (2019) that uses a Wasserstein GAN as a sample generator for a small data set (78 samples) belonging to the biomedical domain, specifically to identify the stages of cancer. This study shows an increase in the prediction efficiency of DNN, Random Forest and Naïve Bayes classifier models by DA with samples generated through a WGAN, in comparison to traditional methods such as SMOTE. Other interesting and related work was published by Douzas and Bacao (2018) in a study in which the validity of data generation is examined through a conditional GAN using imbalanced sets. This work uses CGAN as an oversampling algorithm, creating only samples that belong to the minority class, so it balances the data, and applies DA on a different classifier model, obtaining a slight improvement with respect to other standard methods.

Thus, even if the use of DA in domains where samples are not images, voice data or temporal series (structured data) has been shown to be somewhat beneficial for increasing the prediction rate, still its application its scarce and there are not many works dealing with small size data sets.

3. Methodology

We include in this section the methods and techniques used for the application of Data Augmentation (DA) to a set of well known labeled benchmark problems. The whole scheme of the approach used for performing the DA and evaluating the results can be seen in Fig. 1, where the two main processes involved are shown: (1) the DA process itself, and (2) the classifier used to obtain the prediction error on non-previously used data.

This Methodology section is organized as follows: it starts with a description of the general DA process that will be used to create synthetic samples that together with the original ones will be used for training a classifier. VAEs and GANs models are involved in the DA process, and so they are described independently in the following subsections. Subsection 3.4 describes further modifications implemented for treating unbalanced multiclass data sets, followed by a subsection describing technical implementation details, like the architecture and parameters used in the models. Finally, the section ends with a description of the benchmark data sets used to test the different approaches.

3.1. Generative process for data augmentation

Before the deep generative models (VAEs and GANs) to be used for the creation of synthetic samples are explained in the next subsections, it is useful to describe the general process used for carrying out the DA generation for an initial set of labeled samples. The whole generative process is shown in Fig. 2, where it is possible to see that it includes two models: a "Generator" (G) and a classifier denoted "Generative Classifier" (C). The key thing for understanding the process is to take into account that the present approach deals with supervised benchmark problems (involving labeled samples), and that the DA process includes two steps in relationship to this: first the Generator (G) to create new synthetic data, and second, the use of a Generative Classifier (C) to label the new created samples. It can be seen from Fig. 2 that the whole process

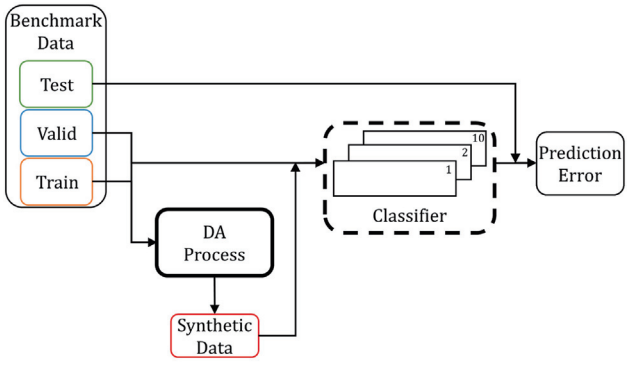

Fig. 1. Flow diagram of the whole process for performing the DA and evaluating the results on a set of benchmark problems. (See text for more details).

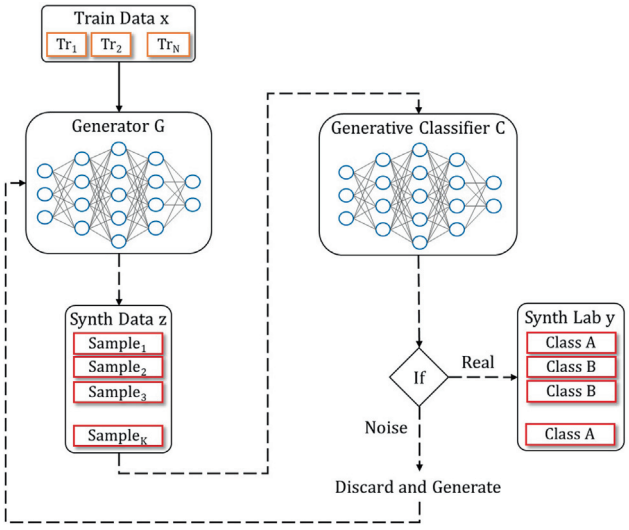

Fig. 2. The generative DA process for the creation of synthetic data. The generative model G is fed with train set. When it is trained, the synthetic samples z are generated and the generative classifier C predicts the appropriate labels y. If the $label_i$ predicted represents noise, $sample_i$ is discarded and G generates another new sample.

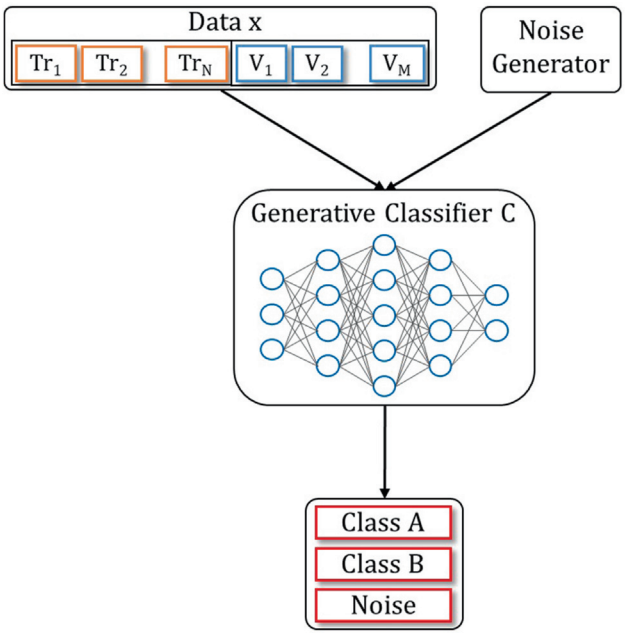

Fig. 3. The training process of the generative classifier (C) involves using training (Tr_i) and validations sets (V_i) from the input data, together with the use of noise sources. The objective of the training is for C to discriminate between the original samples and the noisy ones.

consists essentially in a loop between these two main subprocesses, where the IF control sentence included checks whether the Generative Classifier decides if a created sample is considered as Noise or as a Real sample; and in the last case it assigns a label that will permit its application in a supervised classification scheme.

The training of the Generative Classifier model is carried out using samples from the Train set but it also incorporates some noise from two different sources: (a) through the use of a uniform random distribution; (b) by the addition of Gaussian noise (mean equal to sample mean and standard deviation equals to 0.3). Samples created by both methods have a label that indicates that they are considered as noisy samples. These two different sources of noise are used because the first normal distribution applied is not related to the original distribution of samples, and thus is a way to teach the classifier that this is "purely" noise, while the second source in which a Gaussian perturbation scheme is used creates samples related to the original distribution. Fig. 3 shows an example of an original data set with two classes {A, B}, that with the addition of the noisy samples become a 3-class problem {A, B, Noise}.

On the other hand, the training of the Generator (G) model is carried out using only samples from the Train set. Because different models are used for this purpose (VAEs and GANs), including modifications implemented for treating unbalanced multiclass data sets, the specific training and measuring performance details for each case will be explained separately in its corresponding section. As said before, once the Generator is trained, the generation of synthetic data can be applied followed by the generation of the corresponding labels using the Generative Classifier model. As shown in the Fig. 2, the Generator (G) produces synthetic samples z that are fed into a classifier to get a label for the new samples. The training processes of the Generative Classifier and Generator can be done in any order, or be carried out in a parallel process, since both are totally independent, but it is necessary that the training of both processes have been completed before starting the generation of labels for the synthetic data. A Deep Neural Network, to be described later in subsection 3.5, was used as the model for the generative classifier.

3.2. Variational Autoencoder (VAE) generative models

VAE models (Kingma & Welling, 2013) can be considered a variation of the standard autoencoder network (Bourlard & Kamp, 1988). A VAE, like an autoencoder, consists of two interconnected neural networks denominated encoder and decoder. The encoder is an inference model $q(z|x)$ that maps the input data x to a lower dimension latent variable space z, while the decoder network gets this latent space z variables as input, and outputs the probability distribution of the data $p(x|z)$. The difference of a VAE with an autoencoder network is that, instead of directly generating a latent vector and maximizing the marginal log-likelihood, a vector of means μ and a vector of standards deviations σ are generated, and they are combined to form the latent vector. However, the direct combination of these parameters in the latent vector z implies that this is a continuous random variable, which would not allow the network to learn the distribution that comes from the encoder. To solve this problem and express the random variable z as a deterministic variable, it is necessary to perform the reparameterization trick, whereby z depends on the encoder output parameters (μ, σ) and in addition to a variable ε sampled from a Gaussian distribution (Eq. (1)).

$$z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \varepsilon^{(l)}, \quad \varepsilon^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \tag{1}$$

Another difference of VAEs with respect to standard autoencoders is that the VAE maximizes the evidence lower bound (ELBO) on the marginal log-likelihood of $p(x)$ (Eq. (2)). The term $\mathrm{KL}(q(z|x)\|p(z))$ in the Eq. 2 is the Kullback–Leibler divergence between the distribution of the encoder $q(z|x)$ and the prior distribution $p(z)$. This is used as a regularizer, measuring the amount of information that is lost when the distribution q is used to represent p.

$$\min_p \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathrm{KL}(q(z|x)\|p(z)) \tag{2}$$

Three different types of VAE models were tested in this work, and a scheme of their functioning can be seen in Fig. 4, where standard VAEs are represented on top, Denoising VAEs (DVAE) in the

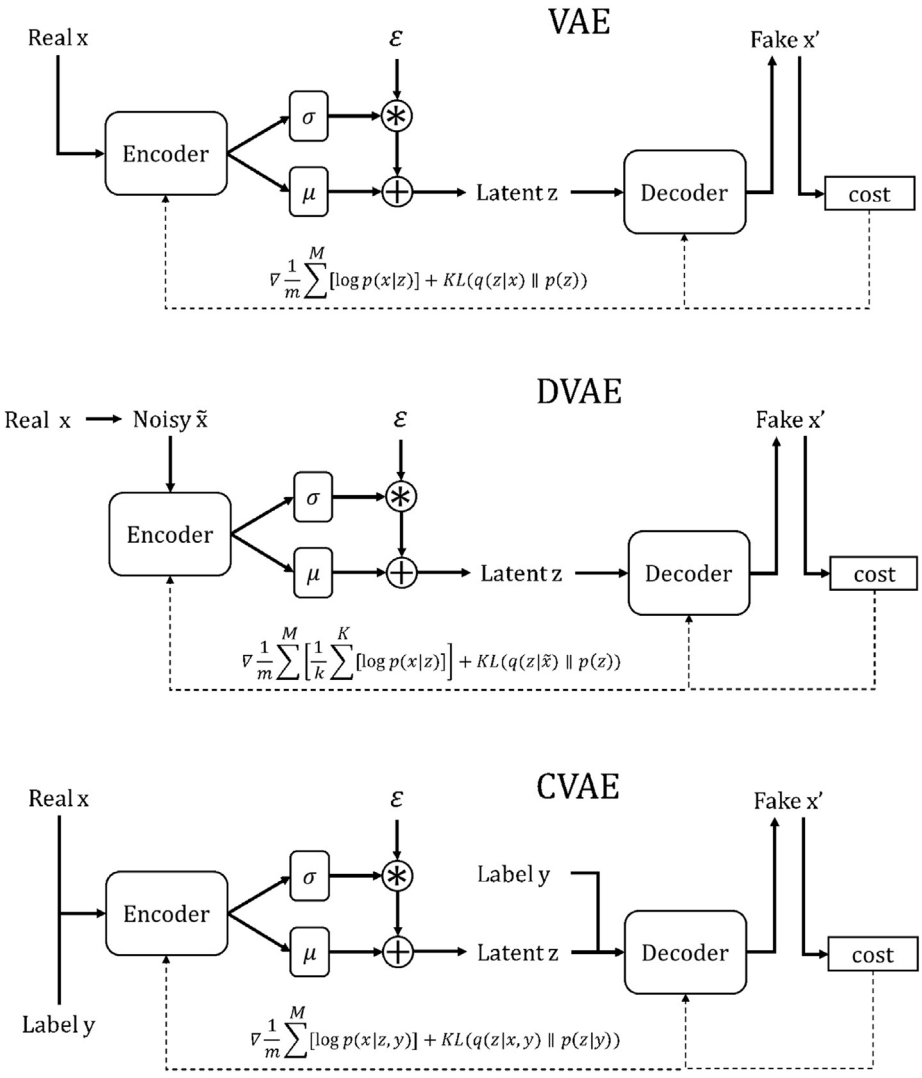

Fig. 4. Schematic diagram of the functioning of the VAE (top figure), Denoising VAE (DVAE) (middle figure), and Conditional VAE (CVAE) (bottom figure). (See text for more details).

middle, and Conditional VAEs (CVAE) at the bottom. The schematic diagram of the VAE (Fig. 4 (top)), shows the flow of information: the input x of the encoder, the intermediate latent space z that is created from the distribution ε and the vectors of mean μ and standard deviation σ, and the output of the model. The propagation of the loss through the network is also shown, a loss which is inferred from the objective function (Eq. (2)). This loss corresponds to a complete batch in an iteration of the model, and indicates the variation that has to be applied to both the encoder and the decoder.

Denoising VAEs (DVAE) (Im, Ahn, Memisevic, & Bengio, 2015) (Fig. 4 (middle)) are different from standard VAEs as they receive as input a deformation of the original input x, following a random distribution that creates a noisy input $\tilde{x}$. The purpose is that by mapping the noisy deformation $\tilde{x}$ to the real input x, a more robust latent representation is created. Eq. 3 shows the objective function to be minimized for the case of the DVAE model.

$$\min_{p} \mathbb{E}_{q(\tilde{x}|x)}\left[\mathbb{E}_{q(z|\tilde{x})}[\log p(x|z)] - \mathrm{KL}(q(z|\tilde{x})\|p(z))\right] \tag{3}$$

The third VAE variant tested in this work is the Conditional one (CVAE) (Kingma, Rezende, Mohamed, & Welling, 2014; Sohn, Lee, & Yan, 2015). In this variant the information referring to a condition y, or another information of the data, is added to the network, both in the encoder and decoder. This makes the network aware of the type of sample that has to be mapped into the latent space z of the encoder, improving the ability of the network to discern between classes of samples. The objective function in this case has to be modified considering the condition y as shown in Eq. (4).

$$\min_{p} \mathbb{E}_{q(z|x,y)}[\log p(x|z,y)] - \mathrm{KL}(q(z|x,y)\|p(z|y)) \tag{4}$$

3.3. Generative Adversarial Networks (GANs)

GAN models (Goodfellow et al., 2014) essentially consist of two coupled neural networks, called respectively *generator* and *discriminator*, trained simultaneously in an adversarial process. The objective of the discriminator network, denoted D, is to distinguish whether a given sample is real or false (i.e., whether the samples are coming from the original distribution or not), so that for an input sample x the discriminator estimates the probability $D(x)$ that the sample is real. The generator, denoted G, is a network whose purpose is to create new synthetic samples such that D can consider them as real samples. The input of G is a noisy random distribution z, and the output $G(z)$ is a distribution assigned to the space of the real samples. Therefore, the generator process is

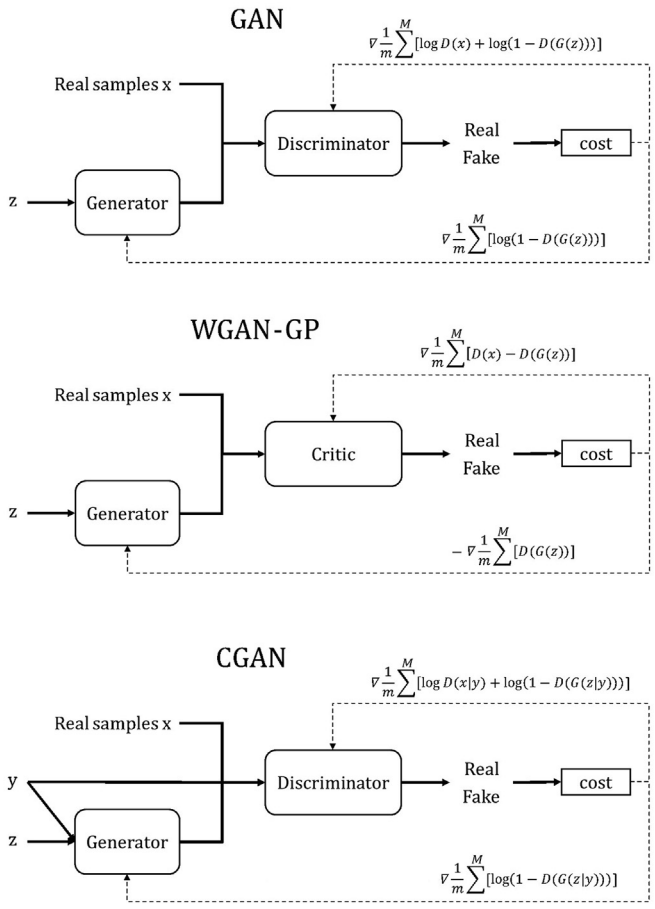

Fig. 5. Schematic diagram of the functioning of the GAN (top figure), Wasserstein GAN with gradient penalty (WGAN-GP) (middle figure), and Conditional GAN (CGAN) (bottom figure). (See text for more details).

opposite to that of the discriminator, giving rise to a competitive environment, which is represented by an objective cost function (Eq. (5)).

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{5}$$

In the GAN objective cost function (Eq. (5)) two parts can be observed that are identified with the behavior of the model: one part related to the recognition of samples that are real ($\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]$), and another one for those samples that are false ($\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$). In this way, the ability of the model to recognize whether the samples are real or false is expressed in Eq. (6), and the error of the network related to the recognition of false samples is modeled by Eq. (7):

$$\max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{6}$$

$$\min_G \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{7}$$

As it has been done for the case of the VAE models, two other variants models of the standard GANs are considered: Wasserstein GANs with gradient penalty (WGAN-GP) and Conditional GANs (CGAN), and the three GAN models used are schematically shown in Fig. 5. In the Figure for each type of GAN the flow of information is shown, including the inputs of the generator and discriminator networks (Critic in the case of WGAN-GP), and the output of the model that corresponds to the probability that the input sample to the discriminator was real.

One of the variants of the standard GAN model considered is the Wasserstein GAN with gradient penalty (WGAN-GP) (Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017), a specific variant of Wasserstein GANs (Arjovsky, Chintala, & Bottou, 2017). GAN models can present stability and speed difficulties in the training process, such as vanishing gradients and collapse mode problems. To reduce these effects, the Wasserstein distance is used in WGANs instead of the Jensen-Shannon divergence for the evaluation of the distance between the synthetic data and the real data, and as a consequence the cost function changes. It is also necessary to make a modification in the discriminator, which in this model is renamed as Critic but retains the notation used: *D*. In the WGAN-GP model used in this paper, the weight clipping operation that is used in the WGAN model is replaced by the use of a gradient penalty term. This term is calculated using a gradient penalty coefficient λ, and using $\hat{x}$, which is calculated from real and synthetic data (Eq. (8)).

$$\text{GP} \longrightarrow \lambda \mathbb{E}_{\hat{x}\sim p(\hat{x})}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2\right] \tag{8}$$

Thus the objective cost function of the WGAN-GP model was modified with respect to standard WGAN case with a gradient penalty (Eq. (9)), and in this way, the WGAN-GP model has a greater stability during training and a greater independence between the Critic and generator, preventing the problems of WGAN due to the size of the weight clipping window.

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] - \text{GP} \tag{9}$$

The second variant of GAN network considered are the Conditional ones (CGAN) (Mirza & Osindero, 2014). As with the CVAE model with respect to VAE, the feature that differentiates the CGAN model regarding GANs, is the addition of information relative to the label *y* of the sample. As shown in Fig. 5 (bottom), the label is added to the latent space *z* at the input of the generating network, and to the real and generated samples at the entrance of the discriminating network. The only variation with respect to the objective cost function of the GAN model is the addition of sample label *y* in the computation of the results obtained from both networks (Eq. (10)). It is assumed that introducing this additional information gives a starting point to the CGAN model to know what features to look for, introducing a bias in the nodes and giving rise to better results, specially in the case of images.

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] \tag{10}$$

3.4. Introduced modifications for treating unbalanced data distribution

When the previously described models (VAEs and GANS) were directly applied to the generation of synthetic data and used for training a supervised predictive model, it was observed that most of the times they generate synthetic data for only one of the category classes, usually for the larger one. Due to this fact, a collapse in the classifier model may occur. In order to avoid the previously mentioned problem, and after experimenting with different options, two methods named 'Multiclass' and 'Balanced Multiclass' are introduced and described below.

The first thing to take into account is that a single generative model was not used for the whole data set, but we use as many models as classes are present in the original data set. In this way, the first step in the process consists in dividing the input data according to problem labels (classes). For each of the *n* classes, one model will be trained taking all samples that belong to the category of the model together with a random selection of samples from the other classes, trying to reach a 20% proportion of samples that do not belong to the model category (in cases where the total

number of samples of the remaining classes is less than this proportion, all existing samples are used).

The first of the implemented method, named 'Multiclass', essentially works by keeping the proportion of samples of each class as in the original data. For the second introduced method, the 'Balanced Multiclass', more samples are generated for less represented classes, i.e., after the DA process is applied the training set will be more balanced. The number of samples L that the class i generator model has to produce, is given by the following equations for both approaches:

$$L = x \cdot K, \quad \text{for the 'Multiclass' case}$$

$$L = [(1 - x)/(n - 1)] \cdot K, \quad \text{for the 'Balanced Multiclass' case}$$

where x is the proportion of class i in original data, n is the number of classes, and K is the total number of samples we want to generate. The introduced methods do not intend to force the model to generate only synthetic samples of a class, but to better adapt the distribution of samples of this class, trying to avoid the collapse that happens when the model is fed with all the data. The number of samples generated by each of the generators depends on the implemented method used, on the number of samples desired and on the proportion of classes in the original set. For example, if we consider a data set with three categories originally containing 60, 20 and 20 samples for each class, and the objective is to augment the size by a 100%, for the case of the 'Multiclass' method the target is to end up with a number of samples close to 120, 40 and 40, while the distribution of classes will be 80, 60 and 60 for the 'Balanced Multiclass' one. If a sample generated by the generator model i is classified as noise, the same model generates a new sample. The final set of synthetic data z results from the union of the different sets of generated samples.

The process described above is schematically shown in Fig. 6. Once we have the training data x, at this example with a binary class data, A and B, they will be split into two sets. One of them is composed of all samples of the class A and 20% of samples randomly selected from class B, and the other is composed of all samples of B and 20% of A. Then, the generative models G_1 and G_2 are trained with their respectively division. Already in the process of generating synthetic samples, the synthetic samples z_1 and z_2 are

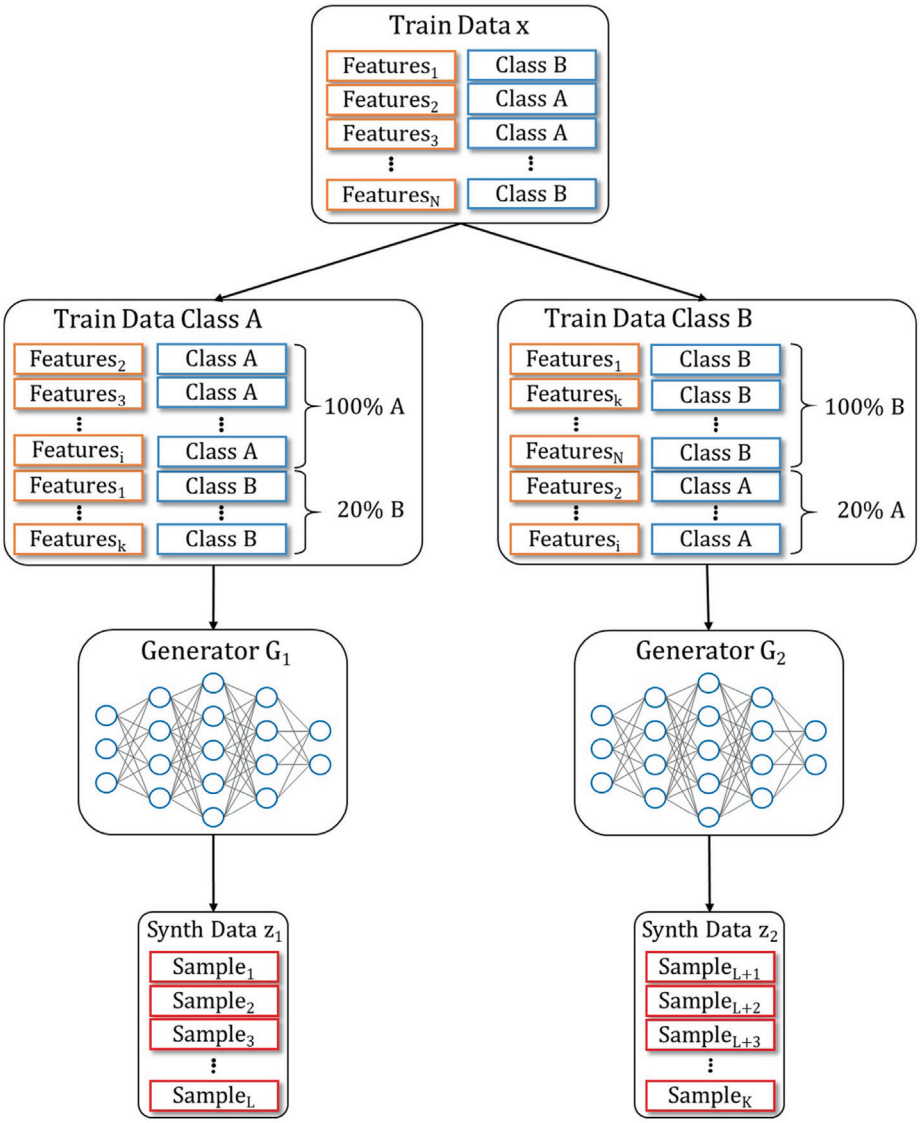

Fig. 6. Scheme behind the modifications introduced to deal with the data augmentation problem. Training data x are split into as many sets as classes. Each division i is composed of all samples from the class i and 20% of samples randomly selected from the rest of the classes. Once the generative models are trained the synthetic samples z are generated following one of the two proposed methods: 'Multiclass' and 'Balanced Multiclass'.

Table 1
Short description of the different models used for the DA experiments.

Model	Abbrv.	Architecture summary	Function
Variational Autoencoder	VAE	Two neural networks: encoder and decoder. The output of the encoder is transformed into a latent variable that becomes the decoder's input, trying to recreate the original data.	Evidence Lower Bound and Kullback–Leibler Divergence.
Denoising VAE	DVAE	Similar to VAE but with noise added to the input of the encoder.	Same as VAE.
Conditional VAE	CVAE	Similar to VAE but with sample labels added to the encoder and decoder inputs.	Same as VAE.
Generative Adversarial Network	GAN	Two neural networks: generator and discriminator. A random distribution is the generator input while synthetic samples are the output. The discriminator receives real and synthetic samples trying to predict whether they are the real ones or not.	Jensen–Shannon Divergence.
Wasserstein GAN with gradient penalty	WGAN-GP	Similar to GAN model but in this case the discriminator is called Critic, and it is trained several times for each iteration of the generator.	Wasserstein distance.
Conditional GAN	CGAN	Similar to GAN model but with sample labels added to the input of generator and discriminator.	Same as GAN.
'Multiclass' GANs and VAEs	_M	This modification creates n models, each one for every existing class of the original data. The input of the model i consist in all samples from class i, plus a 20% from other classes. The number of synthetic samples generated by each model follows the proportion of classes of the original data.	Same as original model.
'Balanced Multiclass' GANs and VAEs	_BM	Similar to the variation _M. There are n models, having each one all the samples from one class and 20% from the rest. The difference with the _M method is the number of synthetic samples generated by each model, that in the present case follows an inverse proportion of the classes in data, and thus tries to balance the distribution of samples.	Same as original model.

generated following one of the methods designed for it, until they have the desired size K.

After explaining all the models studied in this work, Table 1 describes the main characteristics of the different models used, including also the introduced modifications for the synthetic data generation. It presents the name of the models and their formal abbreviation, which in the case of model modifications indicates the suffix that will be added to the abbreviation, '_M' for 'Multiclass' and '_BM' for 'Balanced Multiclass'. The table also shows a brief description of the structure and operation that each model presents, and the objective cost function that they apply.

3.5. Implementations of the models

Details of all models used in this work are given in this subsection, starting with VAEs and GANs models used in the generative part and following with the details of the classifier used when the augmented data is incorporated to the training set. The set of parameters tested in the experiments comes from a selection based on previous experience as the range of parameters should be limited due to involved computation times.

All VAE models (standard VAE, DVAE and CVAE) share the same characteristics: the encoder network comprises 4 hidden layers with 512, 256, 128 and 64 neurons respectively, and the decoder network has a similar structure in reverse order, with 4 hidden layers containing 64, 128, 256 and 512 neurons. In addition, the encoder has two outputs with 32 neurons each, this being the size of the latent space which is the decoder input. Each hidden layer of the encoder network uses Rectified Linear Unit (ReLU) (Xu, Wang, Chen, & Li, 2015) activation functions, (the de facto state of the art of the activation functions in deep learning), but for the decoder the leaky ReLU (Maas, Hannun, & Ng, 2013) activation function was the choice, since it provides more stability for the reconstruction of the data. The decoder output layer uses a sigmoid activation function. In the specific case of the CVAE model, an additional input neuron is included to incorporate the label information.

For the GAN models also 4 hidden layer architectures were used. Generator networks contain 4 hidden layers with 1024, 512, 256 and 128 neurons with ReLU activation function, and an output layer with the size of the data and the hyperbolic tangent (tanh) activation function was used. The noise vector dimension of the generator's input, z, was set to 32. The discriminator network (the Critic in the case of WGAN-GP) also has 4 hidden layers, but with 64, 128, 256 and 512 neurons, respectively, and leaky ReLU activation functions. The discriminator output layer that decides if a created sample is real or fake, contains a single neuron, except for the case of WGAN-GP where an activation function is not applied to the output. WGANs also includes an extra parameter, setting the number of iterations that the Critic performs for each iteration of the generator, a value of 5 was used. An extra neuron is added to input of CGAN in order to include the information about sample label. The Adam algorithm (Kingma & Ba, 2014), a popular optimization algorithm with adaptive learning rate was used for training the models. The parameters used were: 2e−4 as learning rate, 0.5 and 0.9 for the exponential decay rate for 1st and 2nd moments respectively, and 1e−8 for $\hat{\epsilon}$. Batch normalization was also implemented as a regularization technique (Ioffe & Szegedy, 2015), and used in all hidden layers.

In relationship to the classifier models used both for the process of generating synthetic samples, and for the final testing of the models when using the augmented data, the model implemented is a feedforward three hidden layer architecture. Fig. 7 shows a

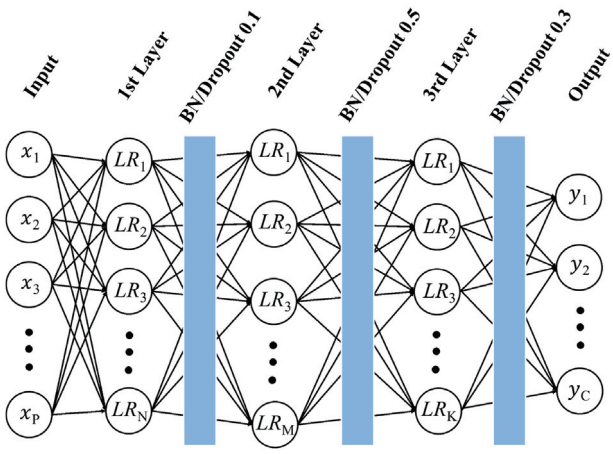

Fig. 7. Scheme of the multilayer perceptron architecture, with 3 hidden layers with N, M and P neurons respectively, using Leaky ReLU (LR) activation function. The boxes represent batch normalization (BN) and dropout.

scheme of such an architecture with (N, M, K) neurons at first, second and third hidden layers. The number of neurons included in each layer was dependent on the data set being analyzed, and previous experiments were used as evidence in order to choose the architecture. Leaky ReLU activation functions were used in the hidden neurons, and the sigmoid activation function was chosen for the output neurons that classify the patterns in the different classes.

A combination of dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), batch normalization and L2 norm (also known as weight decay) were used as regularization techniques in order to prevent overfitting effects. The dropout rate was 0.1, 0.5 and 0.3 in first, second and third hidden layers respectively. The Adam algorithm, mentioned above in the case of the generative models, was used with the same parameter settings.

3.6. Benchmark data

Table 2 shows some characteristics of the benchmark data used for the set of experiments. The data was taken from the UCI (Asuncion & Newman, 2007) and PROBEN1 benchmark sets (Prechelt, 1994) frequently used in the literature. All data sets come from real world data representing problems which that could be called diagnosis tasks. In Table 2, the columns show the name of the benchmark data set, the number of classes, features and instances, and the proportion of classes. For this last column, instead of showing the percentage of instances that belongs to each class, a balance measure (Eq. (11)) based on the Shannon entropy (H) was used. The measure is computed from the number of instances n in the data set, the number of classes k, and the size of each class c_i. A value of Balance 1 means that the set is completely balanced, while values close to 0, corresponds to completely unbalanced sets.

$$\text{Balance} = \frac{H}{\log k} = \frac{-\sum_{i=1}^{k} \frac{c_i}{n} \log \frac{c_i}{n}}{\log k} \tag{11}$$

Two data sets shown in the table are not the original ones from the UCI and PROBEN1 databases, as they were slightly transformed. The idea of these modifications, was to increase the number of data sets used and at the same time analyze similar problems with different number of classes and balance structure. The modified data sets are 'geneN', derived from the data set 'gene' from PROBEN1, and the 'non bal scale' set modified from the data set 'bal scale' originally from the UCI database. In the data set 'gene', each nucleotide taken in the gene sequence window is represented by a pair of inputs that result in 120 features encoded such that nucleotide C is encoded as (−1, −1), A as (−1, 1), G as (1, −1), and T as (1, 1). We have modified this coding such that C was encoded as 0, A as 1, G as 2, and T as 3, leading to a set ('geneN') which has 60 features. In the other case, the 'bal scale' data set contains three classes with 4 features and 625 instances, as shown in Table 2, and it is unbalanced, having two major classes with a representation of 46.08% and a minority class with only 7.84%. The modified data set, 'non bal scale', has only two classes, those that are majority in 'bal scale', so that the instances are reduced to 576, and the data set becomes fully balanced. Finally, it should be mentioned that all the benchmark data sets have been normalized before the experiments were carried out.

Table 2
Characteristics of the nineteen benchmark data sets selected to carry out the experiments. (See text for more details).

Data set	Classes	Features	Instances	Balance
bal scale	3	4	625	0.83
breast cancer	2	9	699	0.93
card	2	15	690	0.99
cleveland	2	13	303	0.99
gene	3	120	3175	0.93
geneN	3	60	3175	0.93
glass	6	9	214	0.84
horse colic	3	58	364	0.84
ionosphere	2	34	351	0.94
iris	3	4	150	1.00
non bal scale	2	4	576	1.00
pima diabetes	2	8	768	0.93
seeds	3	7	210	1.00
sonar	2	60	208	0.99
soybean	19	82	683	0.90
thyroid	3	21	7200	0.28
waveform	3	12	5000	0.99
wdbc	2	30	569	0.95
wine	3	13	178	0.99

4. Experiments and results

First, the synthetic data generation process for each of the models is carried out, following the procedure described in Section 3.1. Once this process is finished, the synthetic data z are added in the needed percentage to the previously named training and validation data and a cross validation procedure is implemented. Test data is not used in the cross validation scheme as it is kept separately for external honest test of the accuracy. As shown in the Fig. 8, the Train and Valid sets are put together and used to make a split through a stratified 10-fold cross-validation procedure, such that 9 folds will be used for training, 1-fold will be used to perform early stopping through validation in the classifier models. Note that the validation folds do not include synthetic data, and that the subset of synthetic data to be added is the same for all the folds, in order to minimize possible random effects.

We then proceed with the training of the classifiers, which present the structure proposed above in the Section 3.5, and with the number of neurons in each layer adjusted to the benchmark data set to be analyzed. Once training is finished, we proceed to make the prediction with the Test data. The result of the classification process is then the average of the accuracy results obtained for the 10-folds. This process is further repeated 10 times with different seeds to reduce random effects.

Table 3 shows the results obtained for the 19 data sets used as benchmarks and described previously in Table 2. First column ('Original') shows the test accuracy when no Data Augmentation (DA) is applied, while following columns show results for different values of the size of the augmented training data (50%, 100%, 200%). The results corresponds for each data set to the model for which the highest validation value was obtained, for which the accuracy is evaluated on the test set. Last column in the Table ($\widehat{RD}$) shows the relative difference computed on the maximum results obtained previously (boldface font indicate the best values for each data set). The relative difference (*RD*) is a measured that indicates the percentage increase when the augmented data set is used in comparison to the original data and is computed using Eq. (12).

$$\text{RD} = \frac{(\text{Acc\_Aug} - \text{Acc\_Ref})}{\text{Acc\_Ref}} \times 100 \tag{12}$$

An analysis of the number of instances of the data sets corresponding to the best obtained values (indicated in bold in Table 3) shows that for 50% of DA the mean of instances is 2214.3 (std. 2476.2), for 100% DA it is 908.6 instances (std. 1154.4), and for 200% it is 404.6 instances (std. 190.5), i.e. that sets with lower number of instances benefit more by using larger percentages of DA. To further analyze the influence of the size of the data sets in the accuracy obtained, we plot in Fig. 9 the test maximum rela-

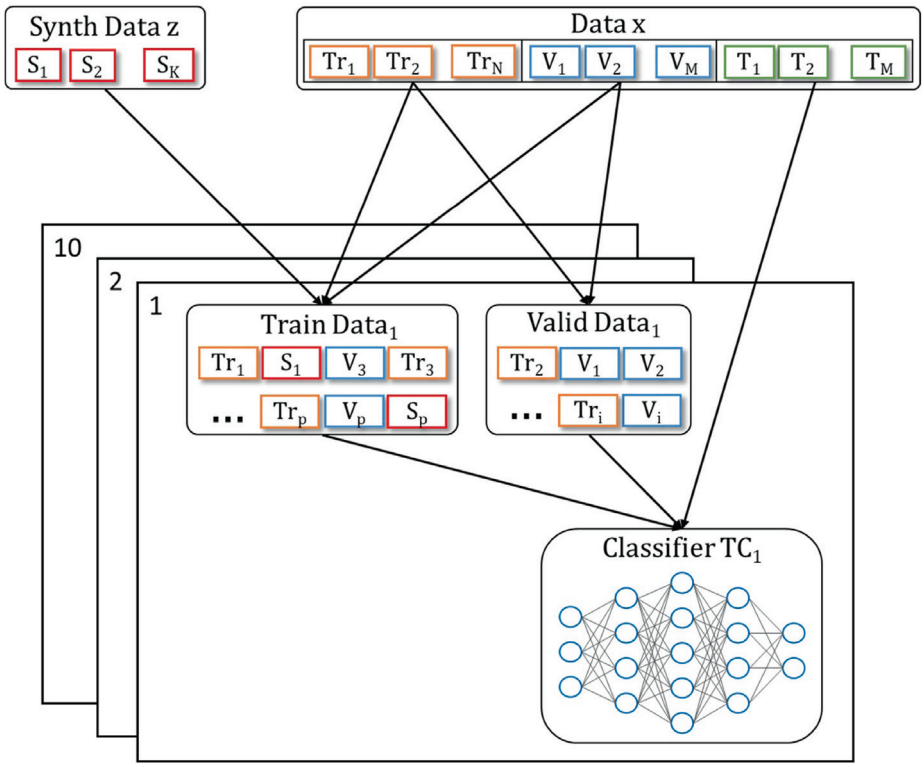

Fig. 8. Classification process for performing the experiments. The real data x that were split at the generative process is used to perform a 10-fold cross-validation process. The training sets are increased by adding synthetic samples, implementing the data augmentation. Once the classifiers are trained, the prediction is made with the Test set and the accuracy is measured.

Table 3
Test accuracy using 50%, 100% and 200% of augmented data, and maximum relative difference ($\widehat{RD}$) for all benchmark problems ordered according to the number of instances. All different methods were applied, and for the one leading to the largest validation error, the obtained test results are reported.

Data set	Original	Data augmentation			$\widehat{RD}$
		50%	100%	200%	
iris	0.9126	0.9323	0.9327	**0.9427**	3.295
wine	0.9633	**1.0000**	**1.0000**	**1.0000**	3.810
sonar	0.6245	0.7519	**0.7781**	0.7676	24.595
seeds	0.8917	**0.8938**	0.8921	0.8933	0.237
glass	0.6351	0.6572	**0.6749**	0.6658	6.264
cleveland	0.8493	0.8636	0.8690	**0.8769**	3.248
ionosphere	0.8862	0.8945	0.9051	**0.9118**	2.892
horse colic	0.6574	0.6593	0.6660	**0.6697**	1.875
wdbc	0.9705	**0.9789**	0.9778	0.9769	0.870
non bal scale	0.9835	0.9869	0.9855	**0.9886**	0.521
bal scale	0.9258	0.9252	0.9250	**0.9441**	1.975
soybean	0.9241	**0.9283**	0.9281	0.9256	0.457
card	0.8536	0.8591	0.8541	**0.8618**	0.962
breast cancer	0.9736	**0.9754**	0.9754	0.9741	0.188
pima diabetes	0.7537	0.7566	**0.7578**	0.7555	0.543
geneN	0.8499	0.8833	**0.8839**	0.8768	3.996
gene	0.9008	**0.9051**	0.9028	0.8985	0.479
waveform	0.8525	**0.8634**	0.8630	0.8632	1.273
thyroid	0.9780	**0.9808**	0.9806	0.9767	0.281
MEAN	0.8624	0.8787	0.8817	**0.8826**	3.040

tive difference $\widehat{RD}$ (last column values from Table 3) versus the number of instances on a logarithmic scale on the x-axis. A linear regression model was fit to the results, obtaining the following equation: $y = -0.59x + 5.69$, with a correlation coefficient of -0.38, indicating a moderate negative correlation between instances number and prediction accuracy gain. (Note that the results from the 'sonar' data set were excluded from this analysis as the RD values are excessively large, fact that can be explained on the extremely low values obtained with the original data. This kind of result was observed previously for this data set (reference Gorman & Sejnowski (1988)) and the explanation is that some patterns are not well represented in the training or test sets in the splitting used.).

We further perform an analysis to see which of the models lead to the better results for each data set and the results are shown in Table 4 for the different values of augmented training data (50%,

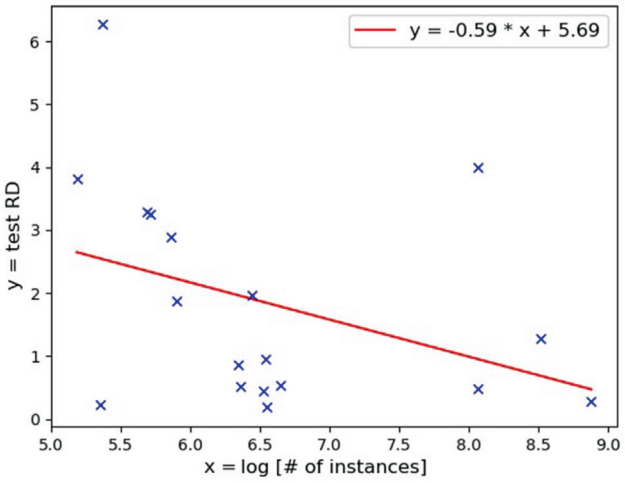

Fig. 9. Relative prediction accuracy difference (RD) versus the logarithm of the number of instances using the values from Table 3. The continuous line is a linear regression adjusted to the data.

Table 4
Models that lead to the best validation results when using 50%, 100% and 200% of augmented data. (See text for more details).

Data set	Data Augmentation		
	50%	100%	200%
iris	GAN_BM	WGAN-GP_M	**WGAN-GP**
wine	**WGAN-GP_M**	**WGAN-GP_M**	**WGAN-GP_M**
sonar	GAN_M	**WGAN-GP_M**	WGAN-GP_M
seeds	**CVAE_M**	CVAE_BM	CVAE_M
glass	CVAE_BM	**CVAE_BM**	CVAE_M
cleveland	WGAN-GP_M	WGAN-GP_M	**WGAN-GP_M**
ionosphere	CGAN_M	GAN_BM	**GAN_BM**
horse colic	DVAE_BM	CVAE_BM	**GAN_BM**
wdbc	**GAN_M**	CGAN_BM	CVAE_M
non bal scale	CVAE_M	CVAE_M	**CVAE_BM**
bal scale	DVAE_M	VAE_M	**CVAE**
soybean	**CGAN_BM**	WGAN-GP_M	CVAE_M
card	GAN	DVAE	**GAN_M**
breast cancer	**DVAE**	DVAE	DVAE
pima diabetes	CVAE_M	**CGAN_M**	CVAE_BM
geneN	CVAE	**CVAE**	CVAE
gene	**CGAN_BM**	CGAN_BM	CGAN_BM
waveform	**GAN_M**	GAN_M	GAN_M
thyroid	**CVAE_BM**	CGAN	CVAE

100%, 200%) for the whole set of benchmark problems ordered according to the number of instances. A summary of these results is shown in Table 5. For each of the six models, three different variations were analyzed, where 'S' refers to the standard model, 'M' refers to the modified model denoted 'Multiclass', while 'BM' refers to the modified model named 'Balanced Multiclass'.

The WGAN-GP model with the 'Multiclass' scheme leads to 10 best results out of the 57 analyzed cases and to the best overall accuracy with a mean in prediction of 0.877. The second most preferred option was CVAE also with the use of the 'Multiclass' scheme that lead to the obtaining of 8 best cases, but noting that on average the accuracy obtained is much lower (0.869).

A careful analysis of the results shown in Fig. 9 and those from Table 5 leads to the conclusion that for the analyzed problems, sets with smaller number of instances behaves differently to those with larger number of patterns. Thus we consider different scenarios as follows, in order to come to a conclusion about which can be considered the best performing method:

If only one method should be applied to the whole data set, our results indicate that the best option is the CVAE model with the 'Balanced Multiclass' scheme and 50% of DA. The first row in Table 6 indicates for this case, the average number of instances for the whole set of problems, the validation results based on which the chosen method was selected, and the test results. For both cases (validation and test) accuracy and RD% values are indicated. Better results can be obtained if the set of problems is divided in two groups according to the number of instances available, and in this case WGAN-GP with the 'Balanced Multiclass' scheme with 100% DA works better than the rest for problems with a mean number of instances of 283, leading to a relative large RD of 4.12%. For problems with larger number of instances (mean equal to 2259), standard VAEs seems to be the best choice, in this case using the 'Multiclass' scheme and 100% of DA, leading to a test RD% of 0.77. The average result for the combined strategy will lead to 0.88 and 2.36 test accuracy and RD%, a bit larger than for the case of choosing a single method (0.87 and 1.03).

All previous results were obtained using the neural architecture shown in Fig. 7. For comparison, a logistic regression (LR) model classifier was also implemented. The results obtained for the 19

Table 5
Number of times that the different models tested are chosen as best models (smaller validation error). The total number of cases (57) corresponds to the 19 data sets considered for the three different DA percentages. First column corresponds to the different models tested and the sampling augmentation schemes. Last column reports the accuracy for the chosen model averaged across all 19 problems. (See text for more details).

Models		Data Augmentation			Best cases	Acc Test
		50%	100%	200%		
	S	0	0	0	0	0.867
VAE	M	0	1	0	1	0.871
	BM	0	0	0	0	0.870
	S	1	2	1	4	0.869
DVAE	M	1	0	0	1	0.870
	BM	1	0	0	1	0.869
	S	1	1	3	5	0.870
CVAE	M	3	1	4	8	0.869
	BM	2	3	2	7	0.870
	S	1	0	0	1	0.871
GAN	M	3	1	2	6	0.874
	BM	1	1	2	4	0.874
	S	0	0	1	1	0.871
WGAN-GP	M	2	5	3	10	0.877
	BM	0	0	0	0	0.876
	S	0	1	0	1	0.874
CGAN	M	1	1	0	2	0.872
	BM	2	2	1	5	0.874

Table 6
Best option model and their results for three cases: considering all data sets together or selecting the model for two groups of problems according to the number of instances. (See the text for more details).

Data	Method	$\overline{\#inst}$	Acc Val	RD	Acc Test	RD
Sets 1–19	$CVAE\_BM_{50}$	1271	0.92	0.40	0.87	1.03
Sets 1–9	$WGAN\text{-}GP\_BM_{100}$	283	0.90	0.73	0.85	4.12
Sets 10–19	VAE_M_{100}	2259	0.94	0.43	0.91	0.77

Table 7
Test accuracy obtained when a Logistic Regression classifier is used on the original data and different cases of augmented data. Last column shows the maximum relative difference ($\widehat{RD}$). (See text for more details).

Data set	Original	Data augmentation			$\widehat{RD}$
		50%	100%	200%	
iris	0.7500	0.9100	0.9207	**0.9317**	24.222
wine	0.9667	0.9894	**0.9964**	0.9950	3.075
sonar	0.7457	**0.7629**	0.7590	0.7533	2.299
seeds	0.8333	**0.8802**	0.8681	0.8433	5.629
glass	0.5163	0.6263	**0.6442**	0.6347	24.775
cleveland	0.9230	**0.9328**	0.9301	0.9293	1.066
ionosphere	0.8803	0.8890	0.8855	**0.8901**	1.120
horse colic	0.6836	0.6914	**0.7130**	0.7096	4.309
wdbc	0.9430	0.9462	0.9451	**0.9495**	0.688
non bal scale	0.9371	0.9351	**0.9366**	0.9379	0.092
bal scale	0.8608	**0.8996**	0.8772	0.8718	4.507
soybean	0.9241	0.9258	0.9300	**0.9335**	1.019
card	0.8703	**0.8883**	**0.8883**	0.8823	2.073
breast cancer	0.9629	0.9641	**0.9649**	0.9644	0.215
pima diabetes	0.7630	**0.7795**	0.7738	0.7734	2.170
geneN	0.7005	**0.7049**	0.6741	0.6778	0.630
gene	**0.8976**	0.8898	0.8864	0.8946	−0.342
waveform	0.8592	**0.8618**	0.8617	0.8610	0.301
thyroid	0.9347	**0.9434**	0.9425	0.9416	0.929
MEAN	0.8396	**0.8642**	0.8630	0.8618	4.146

data sets are shown in the Table 7. The table shows test accuracy values measured for those models with larger validation scores with different values of training DA (50%, 100%, 200%), and the maximum relative difference ($\widehat{RD}$) found.

In comparison to the neural network results, the LR model leads to a larger RD average value (4.146) but noting that the maximum average test accuracy is obtained with 50% DA with a value of 0.8642, lower than the obtained from the NN models when DA is used. With respect to the comparison between the models with which the best results have been obtained, the use of the WGAN-GP model lead to 18 best results, followed by CGAN and GAN with 14 and 12 best results for each model, so the variations of GAN represent the best options in 44 out of 57 cases analyzed. Regarding the modifications, the use of 'Balanced Multiclass' models lead to 22 best results, and 'Multiclass' models lead to 21 best results for each model, so our proposal for dealing with class unbalanced problems represents the best options in 43 out of 57 cases.

A further experiment was carried out using only synthetic data for training the predictive models in order to analyze the quality of the generated samples. Specifically, to obtain the prediction accuracy for these cases, synthetic data were used for training and validation a feedforward neural network classifier using a distribution of samples equal to the one from the original data. The test set is the same one reserved for perform the testing in all experiments. Table 8 shows the accuracy obtained in the test data set when only the original data, only the synthetic data, and when augmented data sets are used (these values corresponds to previously analyzed cases shown in Table 3). The values shown for the column "Data Aug.", are the maximum values reached using the percentages 50 %, 100 % and 200 % of augmented data. The boldface font indicates best values between the test accuracy obtained by original, synthetic and augmented data, clearly showing that in most cases (15 out of 19) it is beneficial to use the augmented sets, but unexpectedly for 4 cases higher values were obtained by using only the synthetic created data. These 4 cases might need further analysis to determine the exact reason but it seems that the synthetic samples are more regularized and thus the generation process eliminates some noise present in the original data.

Table 8
Test accuracy using the original data and the synthetic samples generated, and best test accuracy using 50%, 100% and 200% of augmented data. (See text for more details).

Data set	Original	Synthetic	Data Aug.
iris	0.9126	0.9337	**0.9427**
wine	0.9633	0.9936	**1.0000**
sonar	0.6245	0.5312	**0.7781**
seeds	0.8917	**0.9143**	0.8938
glass	0.6351	0.5979	**0.6749**
cleveland	0.8493	0.8115	**0.8769**
ionosphere	0.8862	0.8821	**0.9118**
horse colic	0.6574	**0.6718**	0.6697
wdbc	0.9705	0.9694	**0.9789**
non bal scale	0.9835	0.9762	**0.9886**
bal scale	0.9258	0.9154	**0.9441**
soybean	0.9241	0.9088	**0.9283**
card	0.8536	**0.8720**	0.8618
breast cancer	0.9736	0.9729	**0.9754**
pima diabetes	0.7537	0.7577	**0.7578**
geneN	0.8499	0.7533	**0.8839**
gene	0.9008	0.6878	**0.9051**
waveform	0.8525	**0.8657**	0.8634
thyroid	0.9780	0.9174	**0.9808**
MEAN	0.8624	0.8386	**0.8851**

Fig. 10 shows the computation times (in seconds and in a logarithmic scale) that have been necessary for the processes of synthetic data generation and classification, plotted as function of the number of instances plus the number of features of the benchmark data sets. Represented by a line consisting of dashes and dots is the computation time necessary to perform the classification process used in the experimentation if only one fold is taken. With a higher value, it is shown computation time necessary for the generation of the synthetic data with the WGAN-GP model represented by a dashed line, and the time necessary for the generation with the WGAN-GP_BM model represented by a continuous line. The number of instances plus the number of features of the data has been used to distinguish the values obtained for the data sets 'gene' and 'geneN', since they present the same number of samples, but different features.

To give a clearer idea of the computation times involved we report that the average time across the 19 benchmark sets for the execution of one experiment with one set of fixed parameters for the WGAN-GP method is 69 s. Further, if we had to repeat all experiments that leads to the results reported in this works (this excludes different tests carried out to tune the models, etc.) the overall computation time will be of 76.37 days. All experiments have been executed on a PC running under Windows 10 64-bits OS, equipped with an Intel Core i7-6700K CPU 4.00 GHz, 32 GB RAM, NVIDIA® Titan Xp GPU with 12 GB RAM. The software utilized for the experiments was CUDA® Toolkit 9.0, cuDNN v7.0, Python 3.6.2v, and TensorFlow 1.8v with GPU support (Abadi et al., 2016).

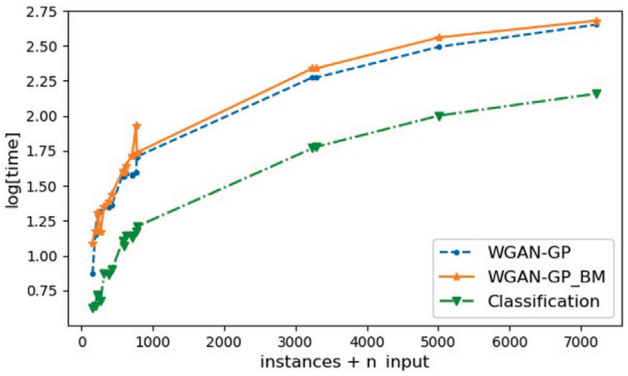

Fig. 10. Computation time used in the different experiments. With a dashed line is represented the time used by the process of synthetic data generation through the WGAN-GP model, and with a continuous line the time used through the WGAN-GP with the 'Balanced Multiclass' scheme with 100% DA. Below is the time spent in the classification process, represented by a line of dashes and dots.

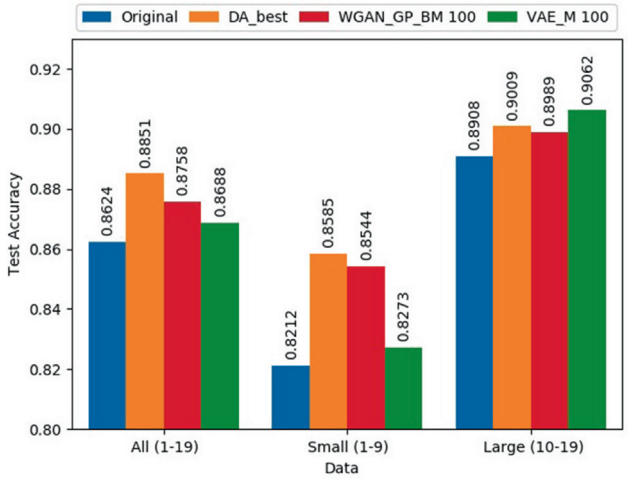

Fig. 11. Test accuracy using the original data, the best option for DA, the WGAN-GP model with the 'Balanced Multiclass' with 100% DA and the VAE model with 'Multiclass' with 100% DA. The results are indicated for three groups: considering all data sets together, and grouped according to the number of instances.

5. Discussion and conclusions

We studied in this work the effect of using different state-of-the-art techniques for Data Augmentation (DA) on the prediction accuracy that can be obtained in supervised problems when small benchmark data sets (number of instances per class lower than 1000) are considered. There are several relatively new studies showing the usefulness of using DA, but up to our knowledge, no previous ones have shown its application and effect on small size data sets. In the present study we applied different types of VAE and GAN models for the generation of augmented data, and have also introduced modifications to the original sample class distribution in order to increase the classification accuracy (two methods named 'Multiclass' and 'Balanced Multiclass' were introduced in subsection 3.4).

The results shown in Table 3, together with those shown in Table 6 can be considered the most relevant results of this work, as they shown the best results that can be obtained when all the different methods tested can be chosen for a given data set or alternatively the results if only one method is to be chosen for all data, considering also the option of grouping the data according to the number of samples available (Table 6). In the first case, the result of this work is that DA can lead to an increase in prediction accuracy of approximately 3% (all tested method are evaluated, choosing the best one according to the validation error). If only one method should be applied, our results show that the best performing method among all data sets tested is the CVAE with the 'Balanced Multiclass' scheme and 50% of DA, method that leads to a 1.03% relative difference increase in accuracy (RD), but noting that better results can be obtained if problems are grouped according to the number of instances available for training; in these case WGAN-GP with the 'Balanced Multiclass' scheme with 100% DA was the best option for problems with lower number of samples leading to an RD of 4.12%, while standard VAE with the 'Multiclass' scheme and 100% DA was the best alternative for data sets with a number of samples larger than 600 (mean number of instances equal to 2259) (cf. Table 6). Nevertheless, it is worth noting that the results show that not all data sets benefit on the same degree, and results with lowest values close to 0.2% to very high values up to 24.6% increase in prediction accuracy were observed. Fig. 11 summarizes the main results for the prediction accuracy obtained with the original and augmented training samples grouped in three cases: all data sets considered together, the nine sets with the lowest number of instances, and the remaining ten sets with the largest number of samples.

In order to have an idea of the quality of the generated augmented samples, we further analyzed the results by training predictive models only with synthetic samples, and the result is that the quality of the generated data is high, as a 0.839 prediction accuracy was obtained when using only synthetic data, a little bit lower than the 0.8624 obtained with the original data (cf. Table 8).

The most important results of this work were obtained by using classifiers based on Artificial Neural Networks with three hidden layers of neurons, in the border between what can be considered shallow or deep architectures, but in order to analyze the effect of DA when using alternative methods, we further tested it using a linear regression classifier that confirms the previous results (cf. Table 7).

The overall conclusion that can be extracted from the obtained results is that DA techniques based on VAEs and GANs constitute useful approaches for increasing the prediction accuracy when small data sets are analyzed, as they generate good quality data, that lead on average to a 1–3% relative prediction increase. In relationship to this, several future studies are planned, in particular the application of the present methods to genomic data, a challenging problem in biomedicine where usually few samples are available, and for which no great advantage have been obtained so far from the application of deep learning problems (Liu et al., 2019).

CRediT authorship contribution statement

Francisco J. Moreno-Barea: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Visualization, Writing - original draft, Writing - review & editing. **José M. Jerez:** Funding acquisition, Resources, Supervision, Writing - review & editing. **Leonardo Franco:** Conceptualization, Funding acquisition, Project administration, Resources, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors acknowledge the support from MINECO (Spain) through grants TIN2014-58516-C2-1-R and TIN2017-88728-C2-1-R, and from Universidad de Málaga (all including FEDER funds). The authors thank NVIDIA Corporation for the donation of a Titan Xp GPU used for this research.

References

Abadi, M., Agarwal, A., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265–283)..

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning* (pp. 214–223).

Asuncion, A., & Newman, D. (2007). UCI machine learning repository..

Bourlard, H., & Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics, 59*, 291–294.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Douzas, G., & Bacao, F. (2018). Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with Applications, 91*, 464–471.

Fernández-Navarro, F., Hervás-Martínez, C., & Gutiérrez, P. A. (2011). A dynamic over-sampling procedure based on sensitivity for multi-class problems. *Pattern Recognition, 44*, 1821–1833.

Formentin, S., Mazzoleni, M., Scandella, M., & Previdi, F. (2019). Nonlinear system identification via data augmentation. *Systems & Control Letters, 128*, 56–63.

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing, 321*, 321–331.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Gorman, R., & Sejnowski, T. J. (1988). Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks, 1*, 75–89.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of Wasserstein GANs. In *Advances in neural information processing systems* (pp. 5767–5777)..

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778).

Hsu, C.-C., Hwang, H.-T., Wu, Y.-C., Tsao, Y., & Wang, H.-M. (2017). Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. In *18th Annual conference of the international speech communication association (INTERSPEECH 2017)* (pp. 3364–3368)..

Hsu, W.-N., Zhang, Y., & Glass, J. (2017). Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In *2017 IEEE automatic speech recognition and understanding workshop (ASRU)* (pp. 16–23). IEEE..

Im, D. J., Ahn, S., Memisevic, R., & Bengio, Y. (2015). Denoising criterion for variational auto-encoding framework. In *Thirty-first AAAI conference on artificial intelligence*. .

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456).

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1125–1134).

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). *Progressive growing of GANs for improved quality, stability, and variation*. CoRR,abs/1710.10196. https://arxiv.org/abs/1710.10196..

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980. https://arxiv.org/abs/1412.6980..

Kingma, D., & Welling, M. (2013). *Auto-encoding variational bayes*. CoRR,abs/1312.6114. https://arxiv.org/abs/1312.6114..

Kingma, D. P., Rezende, D. J., Mohamed, S., & Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems* (pp. 3581–3589)..

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105). Vol. 25..

LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). Deep learning. *Nature, 521*, 436–444.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 4681–4690).

Liu, Y., Zhou, Y., Liu, X., Dong, F., Wang, C., & Wang, Z. (2019). Wasserstein GAN-based small-sample augmentation for new-generation artificial intelligence: A case study of cancer-staging data in biology. *Engineering, 5*, 156–163.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *International conference on machine learning* (pp. 3).

Mirza, M., & Osindero, S. (2014). *Conditional generative adversarial nets*. CoRR,abs/1411.1784. https://arxiv.org/abs/1411.1784..

Moreno-Barea, F. J., Strazzera, F., Jerez, J. M., Urda, D., & Franco, L. (2018). Forward noise adjustment scheme for data augmentation. In *IEEE symposium series on computational intelligence (IEEE SSCI 2018)*..

Pascual, S., Bonafonte, A., & Serrà, J. (2017). SEGAN: Speech enhancement generative adversarial network. In *18th Annual conference of the international speech communication association (INTERSPEECH 2017)* (pp. 3642–3646)..

Piotrowski, A. P., & Napiorkowski, J. J. (2013). A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. *Journal of Hydrology, 476*, 97–111.

Prechelt, L. (1994). *Proben1 – A set of neural network benchmark problems and benchmarking rules*. Technical Report, 21/94..

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., & Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems* (pp. 2352–2360)..

Radford, A., Metz, L., & Chintala, S. (2015). *Unsupervised representation learning with deep convolutional generative adversarial networks*. CoRR,abs/1511.06434. https://arxiv.org/abs/1511.06434..

Reed, R. D., & Marks, R. J. (1998). *Neural smithing: Supervised learning in feedforward artificial neural networks*. Cambridge, MA, USA: MIT press.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). *Stochastic backpropagation and approximate inference in deep generative models*. CoRR,abs/1401.4082. https://arxiv.org/abs/1401.4082..

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision, 115*, 211–252.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85–117.

Shao, S., Wang, P., & Yan, R. (2019). Generative adversarial networks for data augmentation in machine fault diagnosis. *Computers in Industry, 106*, 85–93.

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. CoRR,abs/1409.1556. https://arxiv.org/abs/1409.1556..

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems* (pp. 3483–3491)..

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*, 1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *2015 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1–9).

Tang, B., Tu, Y., Zhang, Z., & Lin, Y. (2018). Digital signal modulation classification with data augmentation using generative adversarial nets in cognitive radio networks. *IEEE Access, 6*, 15713–15722.

Wu, E., Wu, K., Cox, D., & Lotter, W. (2018). Conditional infilling GANs for data augmentation in mammogram classification. In *Image analysis for moving organ, breast, and thoracic images* (pp. 98–106). Springer.

Xu, B., Wang, N., Chen, T., & Li, M. (2015). *Empirical evaluation of rectified activations in convolutional network*. CoRR,abs/1505.00853. https://arxiv.org/abs/1505.00853..

Zhao, J., Mathieu, M., & LeCun, Y. (2016). *Energy-based generative adversarial network*. CoRR,abs/1609.03126. https://arxiv.org/abs/1609.03126..

Zhao, D., Zhu, D., Lu, J., Luo, Y., Zhang, G., Zhao, D., et al. (2018). Synthetic medical images using F&BGAN for improved lung nodules classification by multi-scale VGG16. Symmetry, 10, 519..

Zur, R. M., Jiang, Y., Pesce, L., & Drukker, K. (2009). Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Medical Physics, 36*, 4810–4818.