Novel Applications for VAE-based Anomaly Detection Systems

Novel Applications for VAE-based Anomaly Detection Systems

Luca Bergamin
Department of Mathematics
University of Padova
Padova, Italy
bergamin@math.unipd.it

Tommaso Carraro
Fondazione Bruno Kessler
Trento, Italy
tcarraro@fbk.eu

Mirko Polato
Department of Computer Science
University of Turin
Turin, Italy
mirko.polato@unito.it

Fabio Aiolli
Department of Mathematics
University of Padova
Padova, Italy
aiolli@math.unipd.it

***Abstract*—Deep generative modeling (DGM) is an increasingly popular approach that can create novel and unseen data, starting from a given data set. As the technology shows promising applications, many ethical issues also arise. For example, their misuse can enable disinformation campaigns and powerful phishing attempts. Research also shows different biases affect deep learning models, leading to social issues such as misrepresentation. In this work, we formulate a novel setting to deal with similar problems, showing that a repurposed anomaly detection system effectively generates novel data, avoiding generating specified unwanted data. We propose Variational Auto-encoding Binary Classifiers (V-ABC): a novel model that repurposes and extends the Auto-encoding Binary Classifier (ABC) anomaly detector using the Variational Auto-encoder (VAE). We survey the limitations of existing approaches and explore many tools to show the model's inner workings in an interpretable way. This proposal has excellent potential for generative applications: models that rely on user-generated data could automatically filter out unwanted content, such as offensive language, obscene images, and misleading information.**

***Index Terms*—anomaly detection, variational auto encoder, deep generative model**

I. Introduction

Deep Generative Models (DGMs) are on the rise [1], [2]. Powered by deep learning, DGM had a remarkable impact, as these models nowadays can generate convincing people's faces [3] and news articles [4]. Novel applications are still researched, as they have been proven effective in many tasks, such as model-based reinforcement learning and semi-supervised learning [5].

A recent work highlighted the need for tools to control the generative capabilities of a DGM is crucial for an ethical and social standpoint [6]. For example, state-of-the-art language models, despite having near human-like performance for specific tasks, have a troublesome behavior: they have been shown to favor misrepresentation, amplify social biases, favor stereotypes and even generate derogatory language. Trivial filtering techniques on training data have not been effective in dealing with the issue [6]. Moreover, powerful language models can be abused. Their misuse can enable automated disinformation campaigns and powerful phishing attempts. Research organizations in the world opted for locking such models behind APIs out of fear of abuse [4]. We firmly believe that innovation cannot be driven by an *opaque* system: we should be able to independently assess learning models to constantly improve them. Ideally, we should strive to create generative models that can be instructed to avoid generating specified inappropriate data, removing the capability to generate it altogether.

Anomaly detection systems based on Variational Autoencoders (VAEs) [7], [8] have been a solid inspiration for this task. Although they are generally employed to perform classification, their generative formulation enables them to perform sampling. Moreover, their neural architecture can be easily adapted to different domains. VAE-based anomaly detection systems have been proved effective and resilient, showing to outclass PCA-based methods [7], random forests [8], classic deep neural networks and one-class SVM [9].

Following these research directions, the main contributions of this paper are two-fold: (1) we propose a novel *negative-unlabeled* setting in which generative methods can be instructed to avoid the generation of unwanted data (e.g., outliers, malicious data); (2) we extend an autoencoder-based anomaly detector to a VAE inspired formulation to achieve this ambitious goal. Understanding and promoting this under-explored setting is important for real-world applications that require a specific removal of sensitive topics from DGM, such as language modeling [6] and image synthesis [3].

The rest of the paper is structured as follows. In Section II, we present a brief survey about the applications of generative methods in settings similar to the proposed one. Section III summarizes important notions and terminology which are used throughout the paper. Section IV presents our novel negative-unlabeled setting and detailedly describes the proposed V-ABC model. Section V presents a comparison between a standard VAE and V-ABC in our novel setting, using both toy and real-world data sets. Section VI discusses the results found and shows the ability of V-ABC to avoid generating unwanted data. Section VII wraps up the paper and presents possible future directions.

II. Related Work

Anomaly detection systems, also known as *outlier analysis*, have been researched for decades. These approaches have been successfully applied in many scenarios, such as quality

control, finance, cybersecurity, medical diagnosis, social monitoring, earth science, and data cleaning [10]. Novel graph-based approaches [11] are used to scale up to real-world big data applications. Variational autoencoders (VAEs) [2] have been successfully applied to anomaly detection tasks in both supervised [12], [13] and unsupervised [7], [8] settings.

A common solution for constrained sampling uses fully labeled data to produce samples with a specified class [14]. This model is ineffective in the setting we propose, since the labeling process is noisy and it would aim to reproduce the same noisy distribution. Considering negative constraints for generative models has been researched using statistical learning tools [15], formulating the *Generative Modeling with Negatives* task. Their contribution is mainly theoretical, while our work has a stronger applicative focus. Alternative settings for generative models that learn from partially labeled data sets have been studied for the *positive-unlabeled* case [16], reporting a setting that requires generalizing the positive concept starting from a small positive set and a larger unlabeled data set for generative modeling (e.g., to perform density estimation).

III. Background

In this section, we provide some useful notation used throughout the paper and present the background knowledge that forms the basis of our method.

A. Notation

Bold notation is used to differentiate between vectors, e.g., $\mathbf{x} = [3.2, 2.1]$, and scalars, e.g., $\alpha = 5$. Probability distributions are denoted with lower case letters, e.g., $p(\cdot)$. Parametric distributions are denoted as $p_{\boldsymbol{\theta}}(\cdot)$, where $\boldsymbol{\theta}$ is a vector of parameters. With $\mathbf{z} \sim p(\mathbf{z})$, we indicate a sampling of some vector $\mathbf{z}$ from distribution $p(\mathbf{z})$. $p(\mathbf{x}|\mathbf{z})$ denotes the probability distribution of $\mathbf{x}$ conditioned by $\mathbf{z}$. A training set $\mathbf{X}$ of N examples is denoted as $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$, where $\mathbf{x}^{(i)}$ is the i_{th} example of $\mathbf{X}$.

B. Autoencoders

Autoencoders (AEs) are unsupervised machine learning models based on neural networks. Their purpose is to learn an identity function that reconstructs the original input while compressing data in the process. These models are based on two neural networks, namely an encoder $f_{\boldsymbol{\phi}}$ and a decoder $g_{\boldsymbol{\theta}}$, parameterized by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. The encoder learns a function which maps a high-dimensional input $\mathbf{x}$ into a lower dimensional vector $\mathbf{z}$, while the decoder learns how to reconstruct the original input $\mathbf{x}$ from $\mathbf{z}$. We refer to the reconstructed version of $\mathbf{x}$ with $\tilde{\mathbf{x}}$. So, given an input $\mathbf{x}$, an autoencoder learns functions $f_{\boldsymbol{\phi}}$ and $g_{\boldsymbol{\theta}}$ such that $f_{\boldsymbol{\phi}}(g_{\boldsymbol{\theta}}(\mathbf{x})) = \tilde{\mathbf{x}} \approx \mathbf{x}$.

C. Variational autoencoders

Variational autoencoders are generative models based on variational inference, with an architecture similar to vanilla autoencoders. VAEs assume the input $\mathbf{x} \in \mathbf{X}$ is generated according to the following generative process: $\mathbf{z} \sim p_{\boldsymbol{\theta}^*}(\mathbf{z})$ and $\mathbf{x} \sim p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$, with $\dim(\mathbf{z}) \ll \dim(\mathbf{x})$. In other words, VAEs assume that the input vector $\mathbf{x}$ is modeled as a function of an unobserved random vector $\mathbf{z}$ of lower dimensionality.

The objective of VAE is to estimate parameters $\boldsymbol{\theta}^*$ by maximizing the likelihood of the data (Maximum Likelihood Estimation, MLE), i.e., $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} p_{\boldsymbol{\theta}}(\mathbf{x})$.

Computing the MLE requires solving $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}) d\mathbf{z}$, which is often intractable. However, in practice, for most $\mathbf{z}$, $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \approx 0$, namely these $\mathbf{z}$ provide a minimal contribution to compute $p_{\boldsymbol{\theta}}(\mathbf{x})$. For this reason, the main idea of VAE is to sample values of $\mathbf{z}$ that are likely to have produced $\mathbf{x}$, and compute $p_{\boldsymbol{\theta}}(\mathbf{x})$ just from those. However, for doing that, we need to compute the posterior distribution $p_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, which is also intractable. For this reason, VAEs rely on variational inference, which allows the approximation of the intractable posterior via $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, known as recognition model in VAE parlance. To make the approximation feasible, $q_{\boldsymbol{\phi}}$ is assumed to follow a specific family of parametric distributions, usually a Gaussian distribution with 0 mean and unitary variance. The closeness between $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and the true posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is ensured by the minimization of the Kullback-Liebler divergence (KL), which can be written as:

$$\begin{aligned}\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) = \\ \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] + \log p_{\boldsymbol{\theta}}(\mathbf{x}). \quad (1)\end{aligned}$$

After some rearrangements of Equation (1), it is possible to obtain the so-called Evidence Lower BOund (ELBO) [2], which defines the objective function that a VAE tries to maximize:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x}) &\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z})) \\ &= \mathcal{L}\left(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\phi}\right). \end{aligned} \quad (2)$$

This equation can be interpreted as a reconstruction error (first term), plus the so-called KL loss, which acts as a regularization term.

The autoencoder comes into play when $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\theta}}$ are parameterized by two (deep) neural networks, i.e., the encoder ($f_{\boldsymbol{\phi}}$) and the decoder ($g_{\boldsymbol{\theta}}$), respectively. The parameters of $f_{\boldsymbol{\phi}}$ and $g_{\boldsymbol{\theta}}$ are optimized using stochastic gradient ascent with the aid of the reparameterization trick [2], which enables the computation of the gradient w.r.t. $\boldsymbol{\phi}$. To this end, given an input vector $\mathbf{x}$, the encoder network provides the parameters which define the probability distribution over the $\mathbf{z}$ that are likely to produce $\mathbf{x}$, i.e., $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. This Gaussian distribution is defined by the two outputs of the encoder, namely the mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$ and the diagonal co-variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x})$. The sampling over this distribution is performed via an additional input $\boldsymbol{\epsilon}$, which allows the reparameterization $\mathbf{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}) + \boldsymbol{\epsilon} \odot \boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x})$, where $\odot$ is the Hadamard product.

D. Auto-encoding Binary Classifiers

The Auto-encoding Binary Classifier (ABC) [9] is a supervised anomaly detector based on the vanilla autoencoder. The authors' major contribution is the proposal of a hybrid

approach in anomaly detection. Instead of using a purely supervised or unsupervised method, ABC exploits a binary classifier built in an autoencoder. On the one hand, purely supervised methods have been shown to have low resilience to unknown anomalies; on the other hand, purely unsupervised methods are not able to distinguish negative from positive data, limiting their use in anomaly detection tasks. Using a hybrid approach makes it is possible to merge the advantages of the two methods while addressing their limitations.

The architecture of ABC is based on an autoencoder. We indicate with $f_{\boldsymbol{\phi}}$ the encoder and with $g_{\boldsymbol{\theta}}$ the decoder, parameterized by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. ABC assumes that the data set $\mathbf{X} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ is subdivided in two sets: a set of "normal" data ($y = 1$), and a set of anomalous data ($y = 0$). The objective of ABC is to discriminate between normal data and anomalies. To this end, ABC uses the supervision given by y to change dynamically the training objective depending on the class of the example. The main idea is to penalize the reconstruction of the anomalies, while minimizing the reconstruction error on normal data. After the training, the model can be used to discriminate between anomalies and normal data by using the reconstruction error.

The ABC loss function is defined as follows, where the reconstruction term for vector $\mathbf{x}^{(i)}$ is defined as $\mathcal{L}_{AE}(\mathbf{x}^{(i)}) = \|\mathbf{x}^{(i)} - g_{\boldsymbol{\theta}}(f_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}))\|$, and $\|\cdot\|$ is an arbitrary distance function. The ℓ_2-norm was used in the paper.

$$
\begin{aligned}
&-\log p_\theta(y^{(i)}|\mathbf{x}^{(i)}) = \\
&\qquad y^{(i)}\mathcal{L}_{\text{AE}}(\mathbf{x}^{(i)}) - (1-y^{(i)})\log(1-e^{-\mathcal{L}_{\text{AE}}(\mathbf{x}^{(i)})}) \qquad (3)
\end{aligned}
$$

The first term has to be intended as a reconstruction error, while the second term can be interpreted as an anomaly reconstruction penalty. Given this objective, ABC will learn how to reconstruct effectively desired data while returning wrong reconstructions for anomalies. In this paper, we show how this idea of penalizing anomalies can be used to avoid the generation of unwanted data on variational autoencoders.

IV. Proposed method

This section introduces a novel problem formulation for generative methods and presents our V-ABC model.

A. Problem formulation

We describe a novel negative-unlabeled setting for generative methods. Let us assume that we have an unlabeled data set $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$, where $\mathbf{x}^{(i)}$ is a vector of features describing the i_{th} example. Then, suppose that our data set contains some examples that might be anomalous or undesired, for example, explicit songs or scary faces. We refer to these examples as unwanted data, in the sense that our model should avoid modeling and generating them. Figure 1 depicts an instance of the negative-unlabeled setting. The unlabeled examples are depicted in blue circles, while unwanted data is depicted by using red circles. Unwanted data may overlap with unlabeled data, which is what makes this setting challenging. For this reason, we define a *negative concept*, namely an area in the (input) space in which unlabeled data lives together with unwanted data, and a *positive concept*, namely an area of the space in which we have only evidence of unlabeled data. In the figure, the negative concept is surrounded by a red dashed line, while the positive concept is surrounded by a green dashed line. In general, these two concepts are defined by two different probability distributions. We denote with $p(\mathbf{x}|+)$ the probability distribution which models the data set examples in the positive concept, and with $p(\mathbf{x}|-)$ the probability distribution which models the data set examples in the negative concept. The objective of our approach is to learn $p(\mathbf{x}|+)$ and use it for generating desired (i.e., not anomalous) data.

In this paper, we use a label y to differentiate between unlabeled ($y = 1$) and unwanted data ($y = 0$). This leads us to a labeled data set, denoted with $\mathbf{X} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.

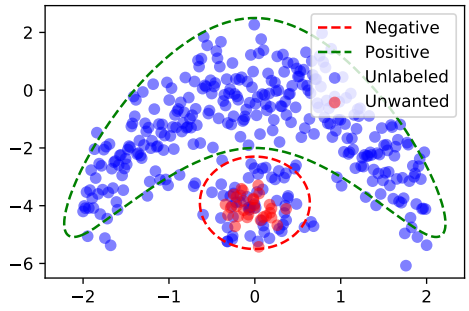

Fig. 1. Possible instance of a negative-unlabeled setting.

B. Variational ABC

Variational ABC (V-ABC) is an extension of ABC, based on Variational Autoencoders, which enables generative modeling capabilities of the original ABC model. In particular, we show that V-ABC can be used to avoid the generation of unwanted data. Some practical observations drive the extension of ABC from an autoencoder-based architecture to a VAE-based one:

1) the ABC model is, at its heart, a classical autoencoder. It simply uses a label y to define a different training objective for desired and anomalous data;
2) the conversion from autoencoders to VAEs occurs by first estimating the reconstruction error via a sampled latent vector $\mathbf{z}$ and then adding a KL-divergence regularization term.

Given these observations, we define the loss function of a V-ABC model for a given input $(\mathbf{x}^{(i)}, y^{(i)})$ as:

$$
\begin{aligned}
&\mathcal{L}_{\text{V-ABC}} = \text{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) \\
&\qquad + y^{(i)}\mathcal{L}_{\text{VAE}}(\mathbf{x}^{(i)}) - (1-y^{(i)})\log(1-e^{-\gamma\mathcal{L}_{\text{VAE}}(\mathbf{x}^{(i)})})
\end{aligned}
$$

The first term is the classical KL loss of variational autoencoders, which forces the distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$ returned by the encoder to be similar to the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$, while the second term can be interpreted as a reconstruction error which changes based on the value of $y^{(i)}$. Notice the similarities between this

latter term and Equation (3). In particular, instead of $\mathcal{L}_{\text{AE}}$ we have $\mathcal{L}_{\text{VAE}}$, which is defined as follows.

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}^{(i)}) = \frac{1}{2\sigma^2}||\mathbf{x}^{(i)} - g_{\boldsymbol{\theta}}(f_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}))||^2$$

In this equation, $f_{\boldsymbol{\phi}}$ and $g_{\boldsymbol{\theta}}$ are the encoder and decoder of the V-ABC model, parameterized by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. Then, σ^2 is a hyper-parameter deriving from the instantiation of the ELBO in Equation (2) for a Gaussian decoder [17], i.e., $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \sim \mathcal{N}(g_{\boldsymbol{\theta}}(\mathbf{z}), \sigma^2 * \mathbf{I})$. Another critical difference between Equation (3) and $\mathcal{L}_{\text{V-ABC}}$ is the hyper-parameter γ in the anomaly reconstruction penalty. The purpose of γ is to calibrate the weight of unwanted examples ($y^{(i)} = 0$) for V-ABC. As γ increases, the model becomes more similar to a classical VAE. In particular, with $\gamma \to \infty$ V-ABC becomes a VAE, modeling only positive samples ($y^{(i)} = 1$). In this latter case, V-ABC will not enforce any reconstruction penalty for unwanted data, but the KL term will constraint the latent distributions to be similar to the prior. An advantage of this formulation is the ability to adjust accurately the distance we want to have from the *negative concept* distribution $p(\mathbf{x}|-)$. Higher values of γ lead V-ABC to generate samples closer to unwanted data, while lower values allow generating samples further from unwanted data.

C. Anomaly detection

Although the main focus of the paper is data generation, it is indeed possible to use the proposed method to perform anomaly detection. Following existing methodology [7], a VAE can be used to perform a Monte Carlo estimate using $E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$, i.e., the reconstruction probability, to check whether new data is anomalous or not, according to a threshold α.

V. Experiments

In this section, we present the experiments performed with V-ABC[1]. In particular, we conducted extensive trials on different toy and real-world data sets, seeking to answer the following research questions:

- **Q1** Is V-ABC able to avoid the generation of unwanted data?
- **Q2** How do the different components of the model (e.g., encoder, decoder, labels) work together to avoid generating unwanted data? Does the latent space of V-ABC show any relevant property (e.g., continuity, completeness)?
- **Q3** Are the results consistent across different initial conditions (e.g., seeds, data set sizes, hyper-parameters choice)?

A. Data sets

We trained V-ABC on different toy and real-world data sets, summarized in Table I. In the table, the number of examples in the positive ($\oplus$) and negative ($\ominus$) concepts are presented, as well as the number of unlabeled and unwanted examples, following the nomenclature proposed in Section IV. Specifically, p is the proportion of negative examples considered as unwanted data during the training of V-ABC, denoted with $y = 0$ in our problem formulation. The remaining negative examples are put in the unlabeled fold, together with the positive examples. The unlabeled examples are denoted with $y = 1$ instead. Our method is then trained on these two latter folds, namely the unlabeled and unwanted folds. The *class* column in the table indicates which class from the data set has been chosen as negative concept. In the following, we refer to negative data as the examples in the negative concept, and to positive data as the examples in the positive concept.

TABLE I
DATA SETS INFORMATION

Data set	class	p	#$\oplus$	#$\ominus$	#unlabeled	#unwanted
`Moons1`	0	1	5000	5000	5000	5000
`Moons2`	0	0.8	5000	5000	5983	4017
`MNIST`	1	0.1	53258	6742	59328	672
`MNIST`	1	0.2	53258	6742	58686	1314
`MNIST`	1	0.3	53258	6742	57986	2014
`MNIST`	7	0.1	53735	6265	59375	625
`MNIST`	7	0.2	53735	6265	58777	1223
`MNIST`	7	0.3	53735	6265	58125	1875
`MNIST`	8	0.1	54149	5851	59414	586
`MNIST`	8	0.2	54149	5851	58858	1142
`MNIST`	8	0.3	54149	5851	58276	1724

a) Moons1: this synthetic data set is composed of two interleaving half circles, also referred as *moons* by the clustering research community. The lower moon contains only positive data, while the upper moon contains negative data. This data set presents a minimal, but existing, overlapping between the two distributions (i.e., lower and upper moons). `Moons1` constitutes an *easy* setting, in the sense that the unlabeled and unwanted folds contain exactly the positive and negative data, respectively.

b) Moons2: the previous toy data set is extended to a harder formulation, where the unlabeled fold contains a portion (20%) of negative data, while the rest (80%) is moved on the unwanted data fold.

c) MNIST: to show the versatility of V-ABC on real-world scenarios, we employ a simple computer vision data set comprising images of handwritten digits. The data set is already subdivided into training and test sets. The training examples related to one digit have been considered as negative data, while all the others as positive data. Then, 10% of the negative data has been selected as unwanted data for V-ABC, while the remaining 90% make up the unlabeled fold, together with the positive examples. In our experiments, we tried different values for p (i.e., $0.1, 0.2, 0.3$) and selected '1', '7', and '8' as negative data in three different trials.

B. Model architecture

Depending on the data set, we used different architectures for V-ABC.

a) Moons: We used a simple Multi-Layer Perceptron (MLP) as encoder and decoder, with two hidden layers with 20 units each. The latent space dimension has been set to 1. `ReLU` has been used as activation function for the hidden layers. For

[1]We make our source code available here: tinyurl.com/2uh5xjzz

the output of the encoder, we used a linear activation. The same is done for the output of the decoder. The final output has been chosen to be linear since we interpret it as the mean of the Gaussian distribution modelled by the probabilistic decoder. A sample can be then visualized by sampling from the aforementioned distribution, using the predicted mean and by using 1 as variance, as we will see in our qualitative results. During training, we fixed hyper-parameter $\sigma^2 = 1$.

b) MNIST: We increased the number of hidden units to 300 and 100 for the outer and inner layers, respectively. Here, a sigmoid function has been applied to the output of the network to compute pixel values between 0 and 1. We fixed $\sigma^2 = 2.5$.

C. Training procedure

During training, we balanced the unlabeled and unwanted data in the mini-batches to mitigate the effect of unbalanced data sets, by ensuring to sample an equal number of unwanted and unlabeled examples for each mini-batch. We used a sigmoidal annealing schedule [18] for hyper-parameter γ, described as follows. (1) At the beginning of the training, we set γ to a relatively high value (e.g., 4). This helps the model in accurately learn the positive distribution without penalizing it for reproducing negative samples in the early stages of training. (2) During training, we decrement γ by using a sigmoidal scheduling, forcing the model to consider progressively negative examples and learn how to not generate them. (3) γ is reduced until it reaches its desired final value (e.g., 1).

For the annealing of the KL term, we followed the procedure described by [18]. The annealing for the KL term and γ has been performed only on the epochs of training. We refer to this number of epochs as E'. In addition, for `MNIST` we used early stopping as a regularization technique, using the test data set to compute a validation score. The test loss function has been used as validation metric. It is important to highlight that the test set has been exclusively used for validation purposes.

For each experiment, we used batch size equal to 80, and `Adam` as an optimizer, with parameters set to default values. For `Moons`, we trained V-ABC for 30 epochs with $E' = 10$ and $\gamma = 3$, while for `MNIST` we used $E' = 5$ and $\gamma = 0.05$.

D. Evaluation procedure

We evaluated the results on the `Moons` data sets in a qualitative manner, by plotting the sampled data over the original data. For the `MNIST` data set, we used a quantitative evaluation. In particular, a Convolutional Neural Network (CNN) classifier has been trained on the original positive and negative labeled data. Then, this classifier is used to measure the correctness of the generated samples after the training of V-ABC. For each `MNIST` data set in Table I we repeated the training procedure using 3 different random seeds to investigate possible sensitivity to initial conditions.

We compared V-ABC to a vanilla VAE. We trained the VAE on the unlabeled fold of the data sets by using the same architecture as for V-ABC.

VI. Results

This section presents the results of the experiments conducted, subdividing them by research question and data set.

A. Negative concept avoidance (*Q1*)

In this section, we analyze the ability of V-ABC to avoid generating negative data by evaluating the output of our model in both a qualitative and quantitative way.

a) Moons: As shown in Figure 3 (`Moons1`), a VAE can faithfully reproduce the selected distribution, but it occasionally creates data that is similar to the negative concept. Additionally, it can create outliers. V-ABC increases the distance of the generated samples from the negative distribution by avoiding generating data near to the upper left of the lower moon, since it is surrounded by negative data. In addition, it also reduces the generation of outliers on the right side of the lower moon. Despite unsupervised models are feasible approaches for sufficiently separated data sets, the negative-unlabeled setting is unfeasible to solve for a classic VAE. In fact, as shown in Figure 4 (`Moons2`), the VAE spans the whole unlabeled distribution. The V-ABC, instead, has a preferable behavior, since it learns how to avoid the upper moon and similar unlabeled data in the process. We observe that the probability mass modeled by V-ABC is simply moved to a safer place, increasing the frequency of generating data points on the left. This is noteworthy, since one may be interested in generating data that is similar as much as possible to the unlabeled data distribution, without actually creating new negative data.

b) MNIST: The performances on `MNIST` have been quantitatively evaluated by using a simple CNN classifier (which has a test accuracy equal to 99.16%) to assess the frequency of unwanted data generated by V-ABC. We denote this metric as *negative generation error*. The classifier used shows an average negative generation error equal to $0.14\% \pm 0.0016$ for V-ABC, as opposed to $8.31\% \pm 0.0181$ for VAE, as reported in Figure 2.

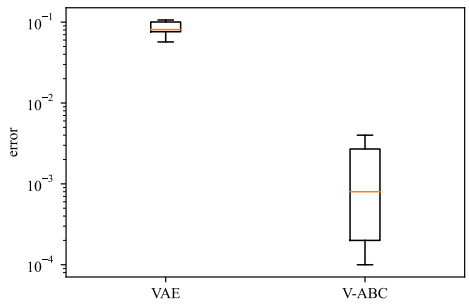

Fig. 2. Negative generation error for `MNIST` data set ($p = 0.2$); VAE vs. V-ABC.

B. Model analysis (*Q2*)

We exploited different visualization techniques to better understand the model components.

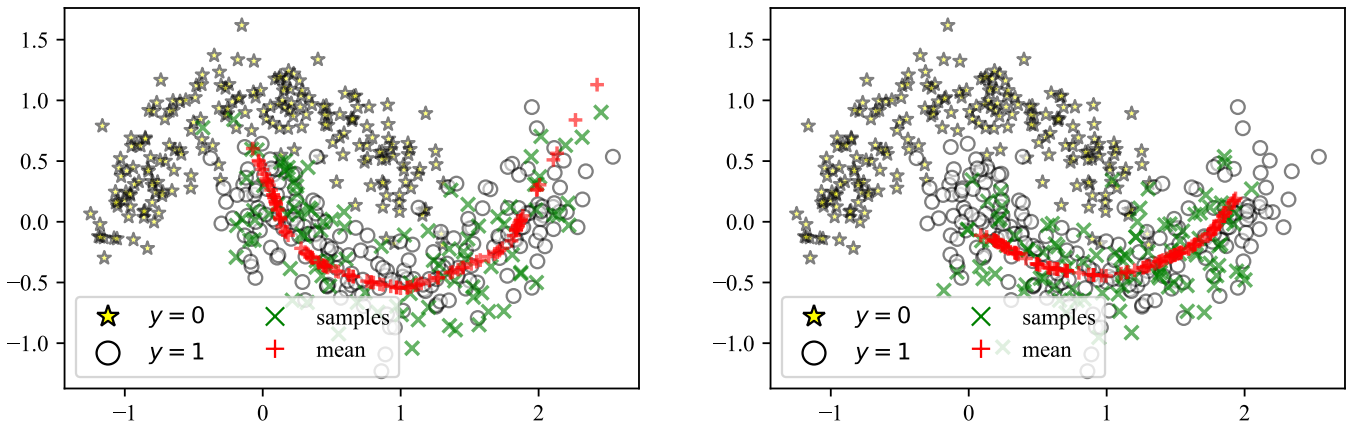

Fig. 3. VAE (left) and V-ABC (right) model samples; `Moons1` data set. Green crosses are sampled data. Red plus signs are the mean of the distributions used for sampling. $y = 0$ is unwanted data, $y = 1$ is unlabeled data. We display only 100 data points for each class to avoid clutter.

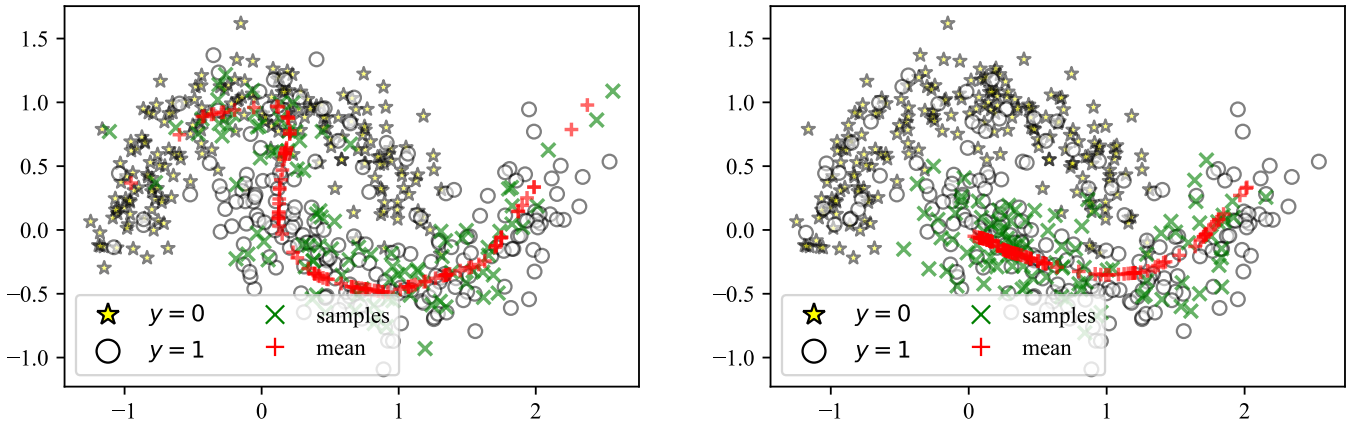

Fig. 4. VAE (left) and V-ABC (right) model samples; `Moons2` data set. Green crosses are sampled data. Red plus signs are the mean of the distributions used for sampling. $y = 0$ is unwanted data, $y = 1$ is unlabeled data. We display only 100 data points for each class to avoid clutter.

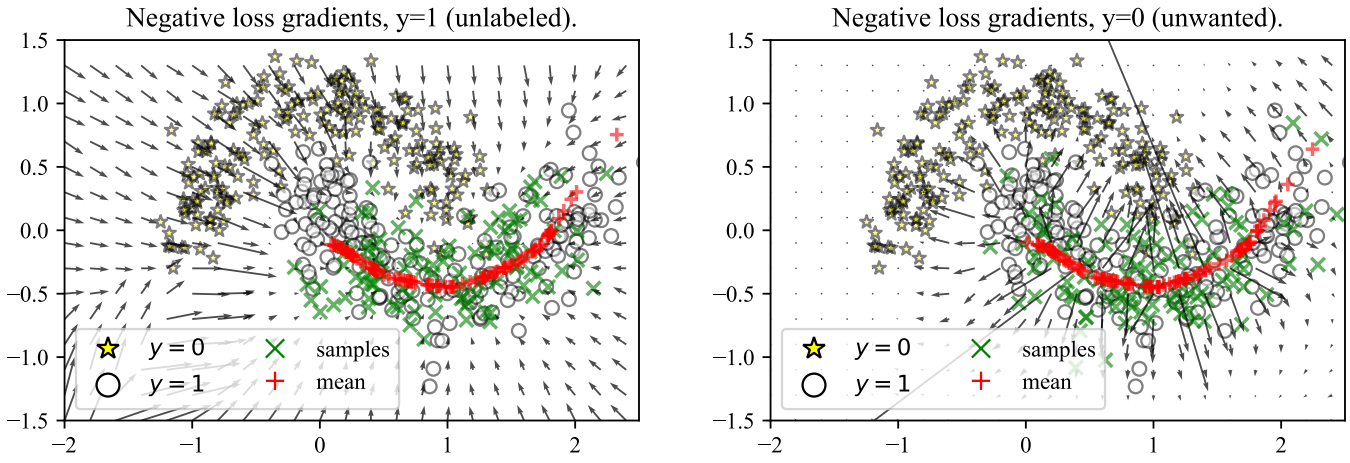

Fig. 5. Vector field of the gradient of $\mathcal{L}_{\text{V-ABC}}$ w.r.t. the input grid, given unlabeled input $y = 1$ (left), and unwanted input $y = 0$ (right). We display only 100 data points for each class to avoid clutter.

a) Moons: A recent paper on GAN [19] shows an effective way to visualize better the impact of a loss function on a 2D data set. We applied this method to a VAE using a simplified formulation. In particular, we computed the gradient $-\frac{\partial \mathcal{L}_{\text{V-ABC}}(\mathbf{x})}{\partial \mathbf{x}}$ and found the direction of the steepest descent of the loss function. By computing the gradient of the loss function w.r.t. an input grid, we created a *vector field*, depicted in Figure 5. To compute the gradient, we skipped the stochastic step in the model by choosing $\mathbf{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$. Avoiding sampling makes the gradients behave as they would do on average.

The computation of the gradient allows to give an interpretation of how the model expects the data points to be positioned in the space. High gradients show the model expects a data point to be positioned in another location, while low gradients show the model is confident that a point is correctly positioned. As we can see in the figure, the gradient pushes unlabeled data to be positioned near the lower moon, while unwanted data has high gradients only for those data points that the model is extremely confident they belong to the positive distribution.

b) MNIST: By using two latent dimensions, we can easily represent the latent space and create a two-dimensional manifold for `MNIST`. To plot the manifold, we selected different values from a grid in the interval $[-3, 3]$ for each dimension of $\mathbf{z}$, then we applied the decoder to each $\mathbf{z}$ value. By using this method, it is possible to inspect the behavior of the decoder, giving an intuitive way to reason on $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. For example, we may assess the area associated with each digit, e.g. evaluating whether a digit is more frequent than another, in a visual way. We can also get a general idea of the sample quality. Figure 6 shows how the V-ABC latent space exhibits a structure which is similar to the classic VAE counterpart. By traversing it along any direction, digits slowly transform in shape, stroke, or angle. A classic VAE can generate only the digits supplied, making it useful and effective for fully labeled data sets. In the negative-unlabeled setting, this model has a clear disadvantage: the model cannot discern unwanted digits, limiting itself to imitate the input distribution that contains all the digits. The V-ABC manifold displays an important difference: the unwanted digit '8' is not generated.

Another way to assess an autoencoding model is to visualize how data is reconstructed. An autoencoder reconstructs data given as input, even if occluded or corrupted. Figure 7 shows, in practice, how a VAE and V-ABC reconstruct the inputs. For the VAE, the digits are reconstructed correctly, except whenever a digit is misread; here, a different, plausible digit is reconstructed. The V-ABC can reconstruct input data just as well, but with a fundamental difference: it reconstructs unwanted digits as different, plausible digits. The positive data is recostructed, instead, correctly, similarly to the VAE.

In Figure 8, we visualize the encoder behavior by computing $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$ and $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x})$ on a training data subset. Hence, we give an intuitive way to reason on the $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ distribution. These values are visualized by creating an ellipse for each data point considered; its mean locates its center, and its variance gives the width and the height to the ellipse. The reparameterization step in a VAE-based model selects with high probability[2] a $\mathbf{z}$ inside this ellipse. The distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ tries to be similar to the Gaussian prior $p_{\boldsymbol{\theta}}(\mathbf{z})$, having mean $\mathbf{0}$ and diagonal co-variance matrix $\mathbf{I}$. In the figure, it is possible to observe that different latent regions encode different digits, with some overlapping because of ambiguous data. Negative digits instead have a different behavior: in this case, the distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is much closer to the prior, as it covers the whole distribution. As long as the positive data covers well the whole latent space, the decoder cannot reconstruct negative data, and instead picks up a random $\mathbf{z}$ vector, drawn by the uninformative prior.

C. Sensitivity to initial conditions (*Q3*)

To test the limits of V-ABC, we trained many model instances, assessing their robustness to initial conditions (**Q3**).

a) Moons: V-ABC is resilient to the different initial conditions tested, since it gives similar results by changing the random seed and p. By increasing γ, the model reduces its distance from the negative data distribution, while by decreasing it, the model increases the distance. A very low γ leads to distortion, changing the shape of the positive moon to further increase the distance from negative data.

b) MNIST: We investigated how the performance varies using different sizes for the unwanted data set. The first plot in Figure 9 shows that the use of more unwanted samples leads to lower negative generation error, while a lower number of unwanted samples leads to a much higher error and variance, showing that there is a clear trade-off between data set size and the model capacity to avoid generating unwanted data.

Finally, we show how the performance changes using different values for γ. As shown in the second plot in Figure 9, a higher γ increases the overall error. Instead, a lower γ can reduce the average error. However, it is more sensitive to initial conditions, as shown by the higher scores for outliers. This behavior decreases the reliability of the model found.

VII. Conclusions

In this paper, we showed it is possible to control the generative capabilities of VAEs. In particular, we repurposed the autoencoding-based ABC anomaly detector, extending it by regularizing its latent space. This novel model, named V-ABC, has proved to be effective in tasks dominated by uncertainty, as we showed in our experimentation in the negative-unlabeled setting. We also contributed to the ABC model by proposing to weight negative samples differently using γ, and to use annealing techniques for training. This resulted in better performances and training behavior for the V-ABC model. As experimentally shown, the value of γ is critical to the overall sample quality. Hence, it is needed to adopt a rigorous hyper-parameter selection process.

We believe that further exploration of the negative-unlabeled setting is imperative. Future work will focus on experimenting with other state-of-the-art VAE-based anomaly detectors to illustrate their generative capabilities.

[2]The area of each ellipse shows 2σ, corresponding to about 95% of data.

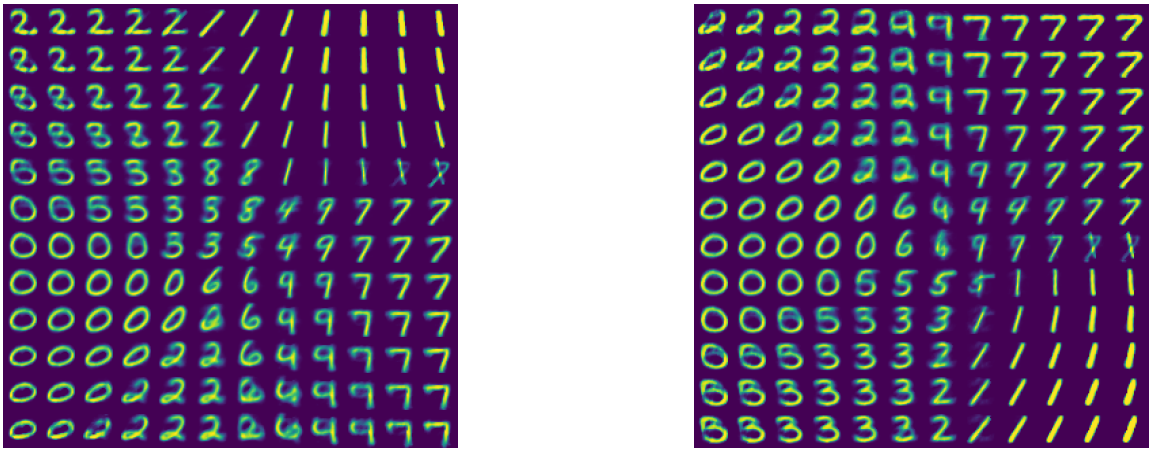

Fig. 6. Manifold: VAE (left) vs. V-ABC (right); `MNIST` data set ($p = 0.2$). The unwanted digit '8' is removed by V-ABC.

Fig. 7. Reconstruction: VAE (upper) vs. V-ABC (lower); `MNIST` data set ($p = 0.2$). Original and reconstructed data are alternated. Threshold filter applied. The unwanted digit '8' is reconstructed to the same digit only for the VAE.

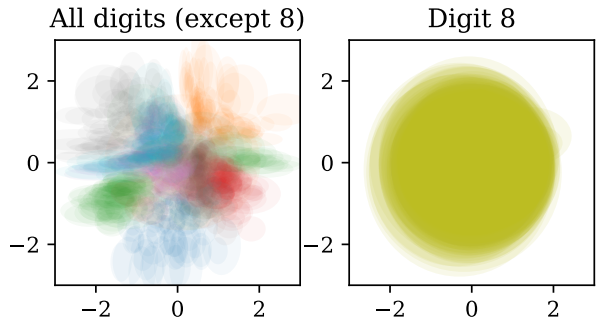

Fig. 8. $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$ and $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x})$ visualization, V-ABC; `MNIST` data set ($p = 0.2$). Each color represents a different digit. Each ellipsis represents the area in $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}) \pm 2\boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x})$. '8' is the unwanted digit.

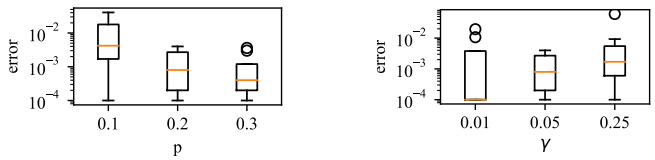

Fig. 9. Negative generation error for different unwanted data set sizes (left), and γ values (right).

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[3] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther, "Biva: A very deep hierarchy of latent variables for generative modeling," *ArXiv*, vol. abs/1902.02102, 2019.

[4] A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever. (2019) Better language models and their implications. [Online]. Available: https://openai.com/blog/better-language-models/

[5] I. J. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *ArXiv*, vol. abs/1701.00160, 2017.

[6] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[7] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[8] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," *Proceedings of the 2018 World Wide Web Conference*, 2018.

[9] Y. Yamanaka, T. Iwata, H. Takahashi, M. Yamada, and S. Kanai, "Autoencoding binary classifiers for supervised anomaly detection," *ArXiv*, vol. abs/1903.10709, 2019.

[10] C. C. Aggarwal, "Outlier analysis," in *Springer New York*, 2013.

[11] X. Ma, J. Wu, S. Xue, J. Yang, Q. Z. Sheng, and H. Xiong, "A comprehensive survey on graph anomaly detection with deep learning," *ArXiv*, vol. abs/2106.07178, 2021.

[12] A. A. Pol, V. Berger, C. Germain, G. Cerminara, and M. Pierini, "Anomaly detection with conditional variational autoencoders," *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1651–1657, 2019.

[13] Y. Kawachi, Y. Koizumi, and N. Harada, "Complementary set variational autoencoder for supervised anomaly detection," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2366–2370, 2018.

[14] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *NIPS*, 2015.

[15] S. Hanneke, A. T. Kalai, G. Kamath, and C. Tzamos, "Actively avoiding nonsense in generative models," *ArXiv*, vol. abs/1802.07229, 2018.

[16] T. Basile, N. Mauro, F. Esposito, S. Ferilli, and A. Vergari, "Generative probabilistic models for positive-unlabeled learning," in *Workshop on NFMCP held with ECML/PKDD*, 2017.

[17] C. Doersch, "Tutorial on variational autoencoders," *ArXiv*, vol. abs/1606.05908, 2016.

[18] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio, "Generating sentences from a continuous space," in *CoNLL*, 2016.

[19] H. Thanh-Tung, T. Tran, and S. Venkatesh, "On catastrophic forgetting and mode collapse in generative adversarial networks," *ArXiv*, vol. abs/1807.04015, 2018.