Capturing Label Characteristics in VAEs

Vincent Bardusco
MVA
Télécom Paris
vincent.bardusco@telecom-paris.fr

Quentin Bourbon
MVA
Télécom Paris
quentin.bourbon@telecom-paris.fr

ABSTRACT

This report summarises the main contributions of [2] and offers new ideas to complement them. This article, published in 2021, offers a new architecture derived from common auto-encoders by disentangling image generation and label information. It shows a novel way of handling labels in the context of image generation, which allows to capture dense and complex information from simple binary initial labels. We propose an improvement of this model by using multiple latent characteristic variables to describe each label, and show the advantages of this new architecture. Finally, we apply this model to a radiology dataset to exploit the increased expressiveness of labels to extend from a classification dataset to an image generation task.

1 CONTEXT

Auto-encoders are one of the first types of models to achieve great results in image generation. The main idea was that the general information about a type of image (e.g. faces, numbers, ...) could be represented in a low-dimensional space. An auto-encoder is then a network consisting of two different parts: an encoder that maps the input image to a representation in a latent space of lower dimension, and a decoder that generates an output image from a vector of the latent space. The training process then consists of feeding images through the network, with the aim of obtaining an approximate generated image after encoding the input and decoding the encoded representation. While this network can be used to learn only the relevant image coding, it can also be used to generate images by sampling the latent space and then decoding the sampled representation to generate a new image. This is a general framework that can work with different types of networks to perform encoding and decoding, be they fully connected layers or CNNs for example.

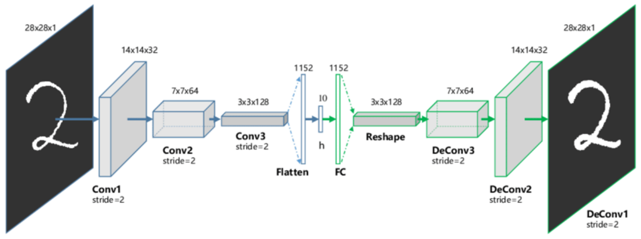


Figure 1: Auto-encoder general scheme

In this architecture, if we denote x the input image, z the latent representation and x' the output generated image, we maximise the likelihood $p_\theta(x|z)$ for the decoder to generate an image. The problem is that it is difficult to estimate the posterior $p_\theta(z|x)$, we do not control $p_\theta(z)$, so when we do sampling in the latent space, we will use a chosen distribution to sample z, but obviously this distribution will not be the same as if the representations were taken from real images, and it is then likely that the generated output will also be affected and not give a great image in the end. A new architecture has emerged to address this problem: Variational Auto-Encoders (VAEs). The idea is simply to force the distribution $p_\theta(z)$ to be Gaussian by modifying the loss used to train the model. As a result, when sampling with a Gaussian distribution, we will sample with the same distribution induced by the input images, and the output generated will then be coherent. To still be able to use backpropagation with this architecture, a reparametrisation trick is used. Specifically, the encoder will output two parameters μ and σ defining the distribution, and the decoder will take as input a Gaussian vector sampled with μ and σ.

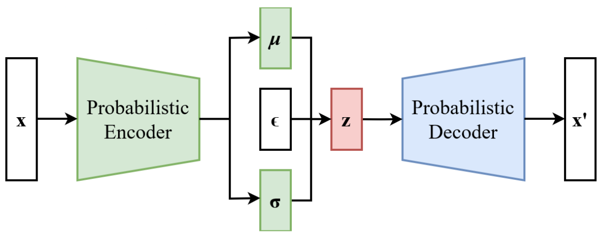


Figure 2: Variational auto-encoder general scheme

An important subfield of image generation is conditional image generation, where we take a label or a set of labels as input and we want to generate an image with respect to those labels. For example, if we take a face generation task, conditional face generation will include labels related to facial features or accessories: a label may indicate whether the generated face should wear glasses, have long or short hair, etc.... Fortunately, VAE can be easily adapted for this new task. During training, we take as input pairs of images denoted x and labels denoted y. x is encoded to give a representation $z = (z^1, z^2, ..., z^p)$, and the decoder takes as input the concatenation of the representation and the labels $z' = (z, y) = (z^1, z^2, ..., z^p, y^1, y^2, ..., y^l)$ to generate an image. As a result, generating an image conditioned on a set of labels is really simple: we sample a random vector z as with usual VAEs, and we concatenate it with the right labels to define the image we want to generate.

Despite its simplicity, this architecture has yielded great results in terms of image generation. It is still used today, with improvements and variations, but keeping the same general architecture, as it often performs better in certain tasks than its image generation counterparts.

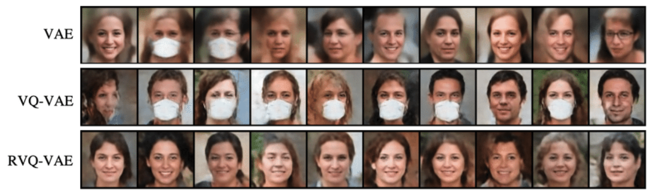


Figure 3: Examples of face generation with different types of VAEs

However, this architecture also has important shortcomings, particularly in the way it handles labels for conditional image generation. To begin with, it only allows the use of binary labels, for example the label 'smiling' is either true or false. In fact, there are often many variations for the same label: a face can be more or less smiling, or there can be different hair lengths for 'long hair', and such complexity cannot be described by binary labels. This means that when generating an image, we can only set a label to one and hope for a specific result, it is not possible to tune a label so that the characteristic is more or less expressed in the generated image.

More importantly, with this architecture, the label information is completely separated from the image coding as it is concatenated. During generation, the latent representation is sampled and the labels are added. This method is arguably suboptimal, because when encoding images during training, the information associated with the labels is obviously also present in the image and will thus affect the latent representation. This shared information is then lost, especially during generation where we sample the representation without conditioning on the labels.

As a result, several ideas have emerged to propose new architectures that allow for more complex handling of labels, which would help to improve the expressiveness of those labels. This can be useful, on the one hand, for improving the overall quality of generation by allowing much more specific images to be generated by tuning the value of each label rather than simply enabling or disabling features, but also for more complex tasks. For example, if we take a medical dataset with a pathology label that indicates whether an image corresponds to a healthy patient or a sick patient, we could use the VAE architecture to encode the image and then tune the pathology label to predict the evolution of the disease, for example to see in which areas it might spread.

We will focus on one of the most recent approaches in VAE architectures focusing on capturing rich label characteristics: Characteristic Capturing VAE (CCVAE), presented in [2].

2 CHARACTERISTIC CAPTURING VAES

2.1 Disentangling labels information and general information

The central idea of CCVAEs, presented in [2], is to capture dense information about the labels directly in the latent space after encoding, instead of concatenating it manually. Since the information about the labels is indistinguishable from the general information about the image in the latent space, the main idea is to disentangle these two parts. The goal is to separate the latent encoded representation z of an image x as follows:

$$z = \{z_{\backslash c}, z_c\}$$

with $z_{\backslash c}$ containing general encoded information on the image unrelated with the labels, and z_c containing dense information on the characteristics we want to focus on.

This disentangling is ensured by the training process. Given an input image x with associated labels $y = (y^1, ..., y^l)$ (where each component of y is a single label), we pass x through the encoder to obtain a latent representation z. This representation is then split into $z_{\backslash c}$ and z_c, the dimensions of which are pre-determined (we will discuss the effect of representation size on the expressiveness of features later). As with an ordinary auto-encoder, an output image x' is generated by decoding the entire representation $z = \{z_{\backslash c}, z_c\}$. What is new is that a classifier is added in parallel, which only takes z_c as input and is optimised to predict y. This means that in this architecture the labels y are never shown directly to the model during training, it is trained to extract relevant information from z_c to predict the labels directly from x, which is much more powerful.

In particular, this means that each component of z_c is sampled from a Gaussian distribution, and is thus a continuous variable with much more expressive power than binary labels. For image generation, we will then be able to tune the components of z_c, giving us more control over the features we want to appear. We can also note that this architecture allows the model to learn dense variations from simple binary labels as training data, which is quite impressive and can also have extended applications.

2.2 Disentangling different labels information

One problem that remains with the architecture is that even if label information is disentangled from general information, we could tune each component of z_c to see its impact on the characteristics of the generated image, but information on different labels will still be entangled in z_c, as the model is only asked to be able to retrieve y from z_c. To give more control over the expressiveness of each label, [2] offers to use a diagonal classifier, which means that each component of y (thus each single label) has to be predicted only by a certain number of components of z_c, and each component of z_c is only used to predict a single component of y, ensuring a complete disentangling of different label information in z_c.

In particular, the authors in [2] recommend giving z_c the same dimension as y, so that each component of z_c is associated with a single label, which means on the one hand that during training each y_i must be predicted only by z_i, and on the other hand that during generation each component z_i will affect its corresponding label. In fact, it provides a much simpler way of dealing with the expressiveness of the label by tuning each component of z_c to more or less express each corresponding label. More formally, it gives us $p(z_c|y) = \prod_i p(z_c^i|y^i)$. However, representing each label with a single latent component can have some drawbacks, and we discuss the implications of using multiple latent components to express a single label later in our experiments.

This CCVAE architecture can be represented as a graphical model as we can see in figure 4.

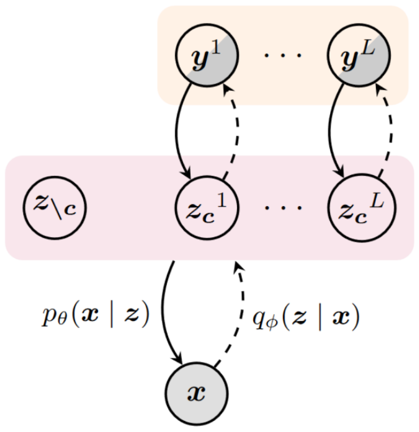


Figure 4: CCVAE graphical model

2.3 Objective function

The final objective function presented in [2] is as follows:

$$\mathbb{L}_{CCVAE}(x,y) = \mathbb{E}_{q_\phi(z|x)}\left[\frac{q_\varphi(y|z_c)}{q_{\varphi,\phi}(y|x)}\log\frac{p_\theta(x|z)p_\psi(z|y)}{q_\varphi(y|z_c)q_\phi(z|x)}\right] + \log q_{\varphi,\phi}(y|x) + \log p(y)$$

We can especially note a global classification term $q_{\varphi,\phi}(y|x)$ showing that y must be predicted from the input image x directly without any additional information, and a usual reconstruction term $p_\theta(x|z)$.

It is also important to note that CCVAEs can also be used in unsupervised learning, with the loss adjusted accordingly, as shown in [2]. The loss becomes:

$$\mathbb{L}_{CCVAE}(x) = \mathbb{E}_{q_\phi(z|x)q_\varphi(y|z_c)}\left[\log\frac{p_\theta(x|z)p_\psi(z|y)p(y)}{q_\varphi(y|z_c)q_\phi(z|x)}\right]$$

In semi-supervised training, the total loss is the sum of the supervised loss and the unsupervised loss for the data points without labels.

3 MULTIPLE CHARACTERISTICS CAPTURING VARIATIONAL AUTO-ENCODERS

In this section we extend the model defined by Joy et al. [2] by considering not only one latent characteristic variable z_c^i per label y^i, but several latent characteristic variables $(z_c^j)_{j\in J_i}$ per label y^i.

In this section we will be careful to separate the notion of label from the notion of characteristic. We call the label y^i the class available to us during our training. For example, it could be the height of a person. And we call features everything that makes up a broader description. For example, it could be all the sizes of the limbs in the human body. In this example we can see that the features make up the label: the height of a person is directly dependent on the size of his limbs. More than that, it's a partition of the label, because the size of a human depends only on the size of its limbs, and the limb sizes don't intersect.

3.1 Motivations

Some labels are too complex to be described by a single latent variable. Thus, the reconstruction of the label would not be very good because the model lacks information about it.

In fact, the CCVAE model encapsulates the labels y in characteristic latent variables z_c, while disentangling the characteristics of different labels by partitioning the latent space so that a given label y^i only has access to a given latent z_c^i. In practice, however, [2] have trivially partitioned the space so that they use a *one to one* strategy: a particular label y^i only has access to a particular latent z_c^i. Intuition tells us that the model would work even better if there were multiple latent variables at the origin of a label, as this would add more information to the description of a label.

Moreover, as we have seen, sometimes a label is composed of several features, and so adopting a *one to one* strategy supposes that a single latent variable z_c^i is able to capture all the features present in the label y^i.

Furthermore, it could also be a way of learning features that capture features that we don't have labels for. Indeed, if we take the human height example again, we could imagine a dataset where we have access to the categorical label *tall*/*small*, but not to the limb sizes. If we use the right number of characteristic latent variables $(z_c^j)_{j\in J_i}$ to describe y^i, we can hope that the model has learned a separate characteristic in z_c^j. For example, z_c^0 could be associated with the size of the arms, z_c^1 with the size of the legs...

So we will use a *many to one* strategy where we use multiple latent variables $(z_c^j)_{j\in J_i}$ to describe y^i.

3.2 The Multiple Characteristic Capturing VAE

To improve the CCVAE model using *one to many* strategy, we keep the model objective and the only thing that changes is $p(z_c|y)$. As it is described in [2], we can introduce a factorised set of generative models $p(z_c|y) = \prod_i p(z_c^i|y^i)$. But here, z_c^i is a set of variables and not just one variable: $z_c^i = (z_c^{i,1}, z_c^{i,2}, ..., z_c^{i,k})$. We also consider that y^i depends only on the set z_c^i: $p(y|z_c)\prod_i p(y^i|z_c^i)$.

This MCCVAE architecture can be represented as a graphical model as we can see in figure 5.

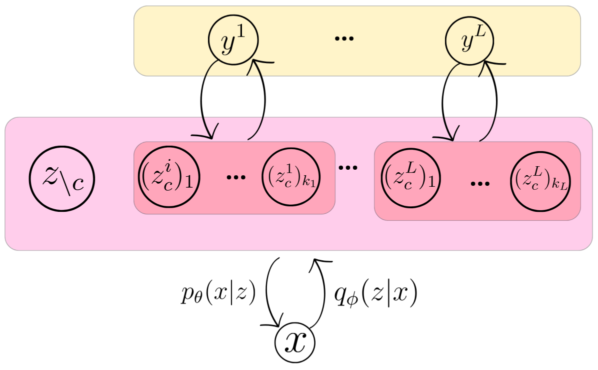


Figure 5: MCCVAE graphical model

4 EXPERIMENTS

We will perform our experiments on three different datasets: MNIST, Chest X-Ray, CelebA, which we will describe in the following subsections.

The experiments aim to show the efficiency of MCCVAE over CCVAE, since the efficiency of CCVAE over other models has already been seen in [2]. To do this, we will evaluate models on classification, intervention and generation. We will use this to vary the number of latent style or trait variables.

We use the architectures from Higgins et al. [1] for the encoder and decoder. The label prediction distribution $q\phi(y|z_c)$ is defined as $Ber(y|\pi_\phi(z_c))$ with a block diagonal transformation $\pi_\phi(.)$ enforcing $q_\phi(y|z_c) = \prod_i q_{\phi^i}(y^i|z_c^i)$. The conditional prior $p_\psi(z_c|y)$ is defined as $\mathcal{N}(z_c|\mu_\psi(y), \text{bloc-diag}(\sigma^2_\psi(y)))$, and both have parameters derived by MLP.

In practice, the classifier and the conditional prior are just a MLP with block diagonal weights. Using pytorch, we can force the block diagonal shape of the weights by initialising them as block diagonal and making the gradient vanish after each backward step by multiplying (the pointwise multiplication of) the gradient of the weights by a block diagonal mask. The i^{th} block of weights are in the form $(w_i^1, w_i^2, ..., w_i^k)$ (or its transpose), where k is the number of latent trait variables we want to get y_i. So if $k = 1$ for each i, we find CCVAE from Joy et al. [2].

We will try three configurations for each data set. One with a base number of style latent variables $z_{\backslash}c$ and $k = 1$ for each i. This is the CCVAE model. Another where we increase the number of style latent variables $z_{\backslash}c$. This is also a CCVAE model. And the last one is to test our model MCCVAE, where we take the original number of style latent variables $z_{\backslash}c$ and we set $k = 2$ for each i.

We use Adam with a learning rate of $2e - 4$, and the batch size depends on the data set, but does not change between models. We use supervised loss for 1 batch over 2 (and unsupervised loss for the other batch).

4.1 Loss computation

We model the posterior $q_\phi(z|x)$ as a normal distribution with parameters given by the encoder. Therefore, we need to sample z, but in order not to destroy the computational graph to do the backpropagation, we will use the reparameterisation trick described in the original article by VAE [3]. The conditional prior $p_\psi(z_c|y)$ also follows a normal distribution with parameters given by the conditional prior layer. The classification losses $q_\varphi(y|z_c)$, $q_{\varphi,\phi}(y|z_c)$ follow Bernoulli laws, as does the prior $p(y)$. The reconstruction error $p_\theta(x|z)$ depends on the data set we are dealing with.

We compute the log probability instead of the probability, so that when the log probability of a Bernoulli law is exactly the opposite of the Binary Cross Entropy. We use Pytorch's BCEWithLogits function to take advantage of the log-sum-exp trick.

4.2 MNIST

The MNIST dataset consists of greyscale images of handwritten digits (0-9). We upsample the images to a size of 60 by 60 while preserving the grey scale. We use a uniform prior so that each logit has a probability of 0.1 of appearing. Then the prior $p(y)$ has no real influence on the class here. We can model the pixels of the image to follow the Bernoulli law, so we can use the binary cross entropy to get the logit of the reconstruction error. We use a batch size of 128. The base model contains 5 style latent variables. And we increase this number to 15. We train for 50 epochs.

4.3 Chest X-Ray

The Chest X-ray dataset contains high-quality chest x-ray images of paediatric patients (one to five years old), categorised into pneumonia (bacterial or viral) and normal classes, obtained from the Guangzhou Women and Children's Medical Center, annotated and validated by expert physicians for AI training. These images are then greyscale. We downsample them to 60*60 to preserve the grey scale. From a quick study of the dataset we get a non-uniform prior, so the prior is important here. As with MNIST, we can model pixels to follow a Bernoulli law, so using BCELoss for reconstruction is justified. We use a batch size of 128. The base model contains 7 style latent variables. We increase this number to 27. We train for 50 epochs.

4.4 CelebA

As in the original paper [2], we decided to use a subset of CelebA's labels, focusing on attributes that are visually discernible in the reconstructions (e.g., arched eyebrows, bags under eyes, bangs, black hair, blond hair, brown hair, bushy eyebrows, chubby, glasses, heavy makeup, male, no beard, pale skin, receding hairline, smiling, wavy hair, wearing a tie, young). We resize these images to 64*64 and preserve the RGB channels. We use a uniform prior. Each image has 0.5 to get each label. Again, as in [2], we model the reconstruction to follow a Laplacian distribution. We use a batch size of 256. As in Joy et al. [2], we use a latent variable of style 27. And we have not increased it for this dataset. We train for 20 epochs.

4.5 Classification

We use classification accuracy as our metric. We perform classification accuracy on train sets and test sets in 1, 2

	CelebA	MNIST	Chest X-Ray
CCVAE	**0.92**	**1**	**1**
CCVAE more style		**1**	**1**
MCCVAE	0.90	**1**	0.98

Table 1: Train classification accuracy

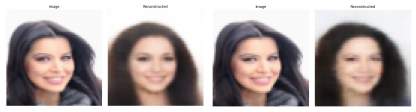
	CelebA	MNIST	Chest X-Ray
CCVAE	**0.90**	**1**	0.78
CCVAE more style		**1**	**0.79**
MCCVAE	**0.90**	**1**	0.77

Table 2: Test classification accuracy

We observe a significant difference in performance in the classification task between CCVAE and MCCVAE, with CCVAE outperforming MCCVAE. This is surprising because with MCCVAE we have several main variables (here 2) that allow us to predict y. The main hypothesis would be that MCCVAE is better for reconstructions, so that backpropagation favours reconstruction over classification. Therefore, we can predict that with more training, MCCVAE would reach the same level as CCVAE for the classification task.

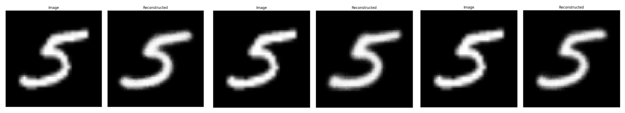
4.6 Diversity of generations

It is often difficult to define what makes a good reconstruction. It has to be faithful to the original image, while showing some interesting variations. Here we see a lot of blurring, certainly due to the loss of reconstruction. Therefore, the main challenge here is not only the diversity of the generations, but also the resemblance to the original image. We present the reconstructions obtained for the different datasets with the different models in 6, 7, 8.



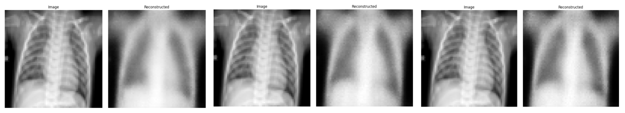
(a) CCVAE **(b) MCCVAE**

Figure 6: Reconstructed image from CelebA



(a) CCVAE **(b) CCVAE more style** **(c) MCCVAE**

Figure 7: Reconstructed image from MNIST



(a) CCVAE **(b) CCVAE more style** **(c) MCCVAE**

Figure 8: Reconstructed image from Chest X-Ray

On these images, the reconstructions seem to be slightly better with MCCVAE than with others. This is confirmed by the training loss, where MCCVAE outperformed CCVAE in 3.

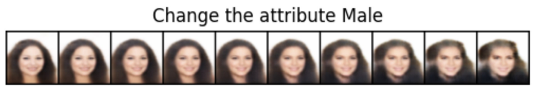
We clearly observe better results with MCCVAE than with CCVAE. We also observe that increasing the latent dimension of the style improves the performance of the reconstruction, which was predictable. In fact, the loss gives us directly a metric of the reconstruction, since the classifier of the CCVAE works better, it means that the classifier loss of the CCVAE is lower. Classification cannot therefore explain the low loss value for MCCVAE. And it remains only the reconstruction loss and the fidelity of the latent space.

	CelebA	MNIST	Chest X-Ray
CCVAE	9676	2298	2419
CCVAE more style		**2242**	2401
MCCVAE	**8657**	2289	**2385**

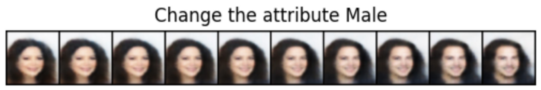
Table 3: Train loss

4.7 Intervention

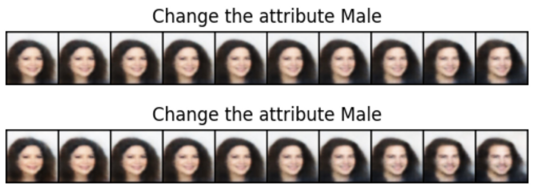
Since we preserve the factorisation $p(z_c|y) = \prod_i p(z_c^i|y^i)$, changing a z_c^i would only produce a change in the label y^i. So we could do a walk in the latent space and display the reconstruction. When we are dealing with CCVAE, only a z_c^i has an effect on y^i. However, when we are dealing with MCCVAE, several latent variables $(z_c^j)_{j\in J_i}$ have an impact on y^i, so we can choose to change one of them or all of them. In the context of MCCVAE, if we change one variable at a time, we can discover new features, some subfeatures of the label. However, the label must be complex in order to be decomposed. We present the latent walk we made in 9, 10, 11, 12



(a) CCVAE



(b) MCCVAE, all changes



(c) MCCVAE, one change at a time

Figure 9: Latent walk with the attribute *Male*

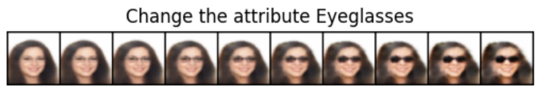
We see some very interesting things.

First of all, all three allow the attribute to be changed quite easily. As you might expect, there is not much difference between CCVAE and CCVAE with more style, as the style does not interfere with the label. On a more global level, there isn't much difference between all the models.

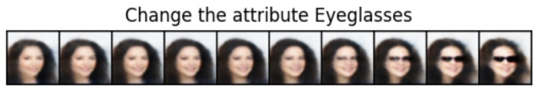
However, when we want to change a complex and large label such as the attribute *Male* in the CelebA dataset, we observe with MCCVAE, when we change one $z_c^{i,j}$ at a time, that $z_c^{i,j}$ are responsible for certain different changes, so the model seems to discover different features that make up the large attribute *Male.*

The same is true for the other changes, but overall we see that one variable does almost all the work.

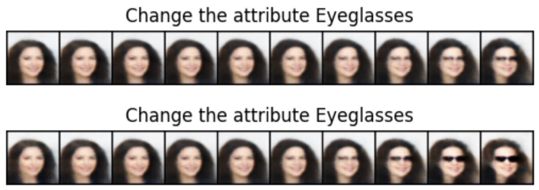
However, this is a big step forward, and we show that the model can learn characteristics of the label using this method.



(a) CCVAE

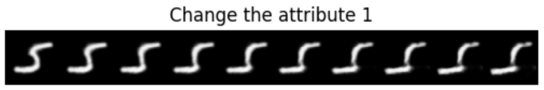


(b) MCCVAE, all changes

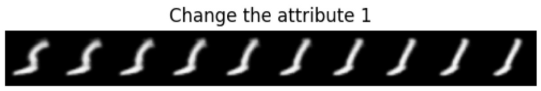

Change the attribute Eyeglasses
Change the attribute Eyeglasses

(c) MCCVAE, one change at a time

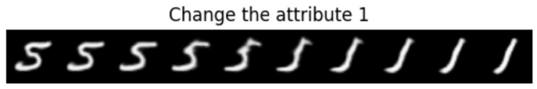
Figure 10: Latent walk with the attribute *Eyeglasses*



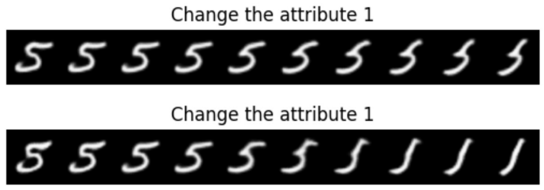
(a) CCVAE



(b) CCVAE with more style



(c) MCCVAE, all changes



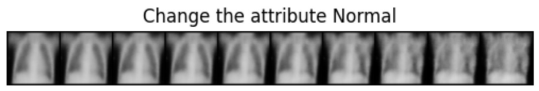
(d) MCCVAE, one change at a time

Figure 11: Latent walk: changing *5* to *1*

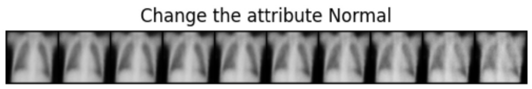
5 DISCUSSION

In addition to presenting CCVAE from Joy et al. [2], we extend the method to capture more features. From our results, our methods seem to work well and could allow to capture more characteristics than what the labels of a dataset can give us.

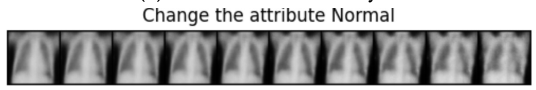
However, a new issue with MCCVAE is the number of characteristic latent variables per label. A good perspective should be to take advantage of the confusion matrix to see where the model struggles to classify and reconstruct. We can then imagine placing more feature latent variables where the reconstruction is poor to improve the reconstruction.



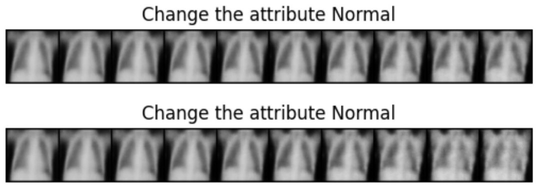
(a) CCVAE



(b) CCVAE with more style



(c) MCCVAE, all changes



(d) MCCVAE, one change at a time

Figure 12: Latent walk: *Bacteria* to *Normal*

All in all, this method offers a great perspective to deal with the variability of more complex labels.

REFERENCES

[1] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Sy2fzU9gl

[2] Tom Joy, Sebastian Schmon, Philip Torr, Siddharth N, and Tom Rainforth. 2021. Capturing Label Characteristics in {VAE}s. In *International Conference on Learning Representations*. https://openreview.net/forum?id=wQRlSUZ5V7B

[3] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. arXiv:http://arxiv.org/abs/1312.6114v10 [stat.ML]