Variational autoencoders: graphical models for (semi) supervision

David FAGET, Biel CASTAÑO, Aissa ABDELAZIZ

Abstract

This project investigates the fusion of Variational Autoencoders (VAEs) and semi-supervised learning, with a specific experimental focus on fully supervised tasks. By combining VAEs' unsupervised representation learning capabilities with the discriminative power of supervised classification, we aim to unravel insights and advancements pertinent to diverse challenges. The exploration is driven by the potential synergy between these paradigms in the context of learning from limited labeled data.

1 Introduction

In the realm of machine learning, the quest for more efficient and effective models has fueled the exploration of novel paradigms that seamlessly blend generative capabilities with discriminative tasks. One such remarkable synthesis lies at the intersection of Variational Autoencoders (VAEs) and semi-supervised learning, where the union of unsupervised representation learning and supervised classification gives rise to a powerful framework with the potential to reshape the landscape of artificial intelligence.

Variational Autoencoders, characterized by their ability to learn rich probabilistic representations of complex data, have emerged as a cornerstone in the pursuit of unsupervised learning. The inherent capacity of VAEs to model latent structures within data distributions not only facilitates data compression and reconstruction but also lays the groundwork for generating new, diverse samples from learned representations. This latent space, capturing the underlying structure of the input data, has proven invaluable in various domains, from image synthesis to natural language processing.

On the other hand, the imperative for learning from limited labeled data has spurred the development of semi-supervised learning paradigms. Traditionally, supervised learning relies heavily on labeled examples for model training, often requiring vast amounts of labeled data to achieve high performance. Semi-supervised learning, however, introduces the innovative concept of leveraging both labeled and unlabeled data to enhance model generalization, breaking away from the limitations imposed by data scarcity.

We would like to stress that the aim of this work is not to reformulate the papers, but to look at them from a critical point of view (in particular, explaining points that were not very clear to us and that we had to work on more on our own) and to make experiments by modifying some aspects of the proposed code in [1].

Therefore, this project embarks on an exploration of the symbiotic relationship between VAEs and semi-supervised learning. We will first explain the basics of VAE, which will serve as a starting point for introducing more complex models such as Characteristic Capturing VAE (CCVAE). Finally, we will present the experiments carried out with the attached code.

2 Variational Autoencoders and its extensions

In this comprehensive analysis, we delve deeply into the intricacies of advanced machine learning models, with a primary focus on the Variational Autoencoder (VAE). This pioneering architecture marks a significant evolution in the realm of unsupervised learning. The VAE emerged as a progressive advancement over traditional autoencoders, excelling not only in data compression and reconstruction but also in embedding a sophisticated probabilistic approach. This approach adeptly addresses the complexities and uncertainties inherent in modeling real-world data distributions. We will discuss how we understood these concepts in section 4.

A central innovation of the VAE is its mechanism for transmuting input data into a latent space representation. Each point in this latent space encapsulates a probability distribution, delineating potential data reconstructions. This approach goes beyond just squeezing data into a smaller form. It opens up possibilities for creating new, well-organized samples from the carefully studied distributions. The VAE's encoding-decoding schema navigates a nuanced equilibrium, striving to maintain fidelity to the input data while conforming to the structured constraints of the latent space.

The VAE's probabilistic framework, key for modeling uncertainty often overlooked by deterministic models, is crucial for understanding complex phenomena. Its ability to handle uncertainty shines in semi-supervised learning with limited labeled data. As a foundational element in advanced machine learning, the VAE guides our approach towards innovative breakthroughs, enhancing understanding and accuracy in challenging scenarios.

In the realm of graphical models, particularly within the context of semi-supervised learning, the role of Variational Autoencoders (VAEs) and their advanced variants like Conditional VAEs (CVAEs), M1+M2 models (presented in [2]) or Characteristic Capturing VAEs (CCVAE, presented in [1]) is significant. These models extend the foundational principles of VAEs, adapting and enhancing them for more specific applications and challenges.

Conditional VAEs (CVAEs), for example, introduce a conditional component to the VAE architecture. This modification allows the model to generate outputs conditional on some additional information, such as labels or other data features. This approach is particularly beneficial in tasks where the generation of data needs to be guided by specific conditions or contexts, offering a more controlled and targeted data generation process.

The M2 model, part of the broader family of M1+M2 models, represents another intriguing advancement. It combines unsupervised and supervised learning in a stacked manner. The M1 model acts as an unsupervised VAE, learning representations of the data, while the M2 model, layered on top of M1, utilizes these learned representations for supervised tasks. This hierarchical approach enables the model to leverage both labeled and unlabeled data more effectively, making it a powerful tool for semi-supervised learning tasks where labeled data is limited.

These models, like the original VAE, are grounded in the principles of graphical modeling, which allows them to capture complex dependencies and relationships within data. By integrating the strengths of deep learning with probabilistic graphical models, CVAEs, M1+M2 models, CCVAE and other similar variants offer enhanced flexibility and capability in handling diverse and complex datasets. This versatility is crucial in many real-world scenarios where the nature of data and the availability of labels can vary significantly, demonstrating the ongoing evolution and applicability of graphical models in modern machine learning challenges.

3 The Characteristic Capturing VAE (CCVAE)

3.1 Project Scope and Objectives

In the context of our project, we will specifically focus on the application of the proposed Characteristic Capturing Variational Autoencoder (CCVAE) framework [1]. The paper introduces CCVAE as a novel model that explicitly captures label characteristics in the latent space without directly associating label values with latent variables.

In fact, the paper challenges the conventional approach to supervision in Variational Autoencoders (VAEs), where labels are typically associated with a partially observed augmentation of the latent space z_y. The authors contend that this common practice may fall short in capturing the nuanced characteristic information linked to labels, outlining key issues:

Firstly, the assumed z_y may inadequately encapsulate the diverse characteristics associated with labels. There's a recognized challenge in disentangling z_y from other latent variables z, hindering effective representation learning. Additionally, manipulating specific characteristics without altering the categorical label poses a difficulty.

The paper highlights a mismatch between training and generation phases, pinpointing that treating labels as explicit latent variables can disrupt the alignment between the VAE prior $p(z)$ and the pushforward distribution $q(z)$, impacting the generative capacity of latent variables.

We will not explain further the functioning of this model as this would be to reformulate the original paper, which is not the aim of the project.

Our experimental emphasis will be on leveraging CCVAE for supervised learning in Variational Autoencoders (VAEs), and exploring how CCVAE can encapsulate characteristics for each label in an isolated manner.

3.2 Explanation of unclear points in the paper

The aim of this sub-section is to explain in more detail some points in the paper that were not very clear to us and that we have had to investigate further.

We are going to explain the phrase "y has to be imputed" in page 16 of [1]. It relates to handling unlabeled data in a semi-supervised learning setting, particularly in the context of adapting the DIVA model.

The DIVA model is initially focused on domain invariant classification and disentangling various factors such as domain, class, and other generative factors. The authors of the paper are discussing modifying DIVA to suit their problem of encapsulating label characteristics (like in their CCVAE model).

In the original DIVA model, and in many semi-supervised learning models, you have two kinds of data: labeled (where both x and y are known) and unlabeled (where only x is known). The phrase "y has to be imputed" refers to the scenario with unlabeled data, where the label y is not directly observed and must be inferred or estimated.

The authors suggest removing z_d (a latent variable for domain in DIVA) as they are not dealing with multi-domain problems. They also propose incorporating a factorization similar to what's in their CCVAE model, and they define two objective functions, one for supervised learning L_{SDIVA} and one for unsupervised learning L_{UDIVA}. In the supervised case, you have labels for training, but in the unsupervised case, you do not.

For the unsupervised learning part, since y (the label) is unknown for a given x, it needs to be estimated (or "imputed") as part of the learning process. The objective function L_{UDIVA} includes terms that involve estimating y given the latent representation z_c. This imputation is a key step because it allows the model to learn from unlabeled data by making educated guesses about what the labels might be, based on the learned representations.

In summary, "y has to be imputed" means that in the context of adapting DIVA for the authors' purposes, and especially for handling unlabeled data, the model needs to infer or estimate the labels y as they are not directly available. As we have seen, this is a common challenge in semi-supervised learning, where leveraging both labeled and unlabeled data is crucial.

4 Discussion

We would like to mention here that a challenging stage was to understand the functioning of CCVAE. We consider that this paper is dense and difficult to understand without previous knowledge in simpler models like VAE. Therefore, the procedure we followed to understand it was as follows: first we studied how the autoencoder works, then we studied the VAE in depth (and also semi-supervised VAE) and finally we understood the CCVAE. In order to understand VAE, paper [3] was a great help, and https://pyro.ai/examples/ss-vae.html helped us to understand the semi-supervised VAE. This stage was undoubtedly one of the most time-consuming, as none of the three of us had a background in these subjects and the VAE class was already in December, which forced us to be self-taught in this regard.

Running the code was a bit challenging process, because of the number of errors arising from dimension mismatches, incorrect value types, and other issues. Initially, we attempted execution using the MNIST dataset. However, we quickly realized that extensive modifications were necessary, introducing significant complexity to the task. Additionally, we were concerned that utilizing MNIST might compromise the relevance of our experiments. Simultaneously, challenges emerged with the CelebA Dataloader, necessitating some adjustments.

Moreover, importing the CelebA dataset was also a challenge due to its size and indirect accessibility. Furthermore, we encountered issues related to the duration required for the model to load data and perform training because of the extensive model parameters and the computational workload involved.

5 Implementation Details

In this part, we will describe our implementation of the CCVAE model, detailing how we closely followed the mathematical formulas and concepts outlined in the paper.

5.0.1 Code Architecture

Our project was developed in the Pytorch environment, chosen for its extensive range of functions and extensions that simplify implementing standard functionalities. For code execution and result visualization, we utilized Colab Notebook. Furthermore, we have included checkpoint files for our CCVAE model, which has been trained on diverse datasets across multiple scenarios.

5.0.2 Model Architecture

The model is composed of four modules: the encoder, the decoder, the classifier, and the conditional prior module, as represented in Figure 2. First, let's present each of the modules separately.

- The encoder is a convolutional neural network that takes as input an image $x \in \mathbb{R}^{h\times w\times c}$ and outputs the mean μ_ϕ and the variance σ_ϕ^2 of a normal distribution on $\mathbb{R}^m$. From this distribution, it is possible to sample a latent vector $z \in \mathbb{R}^m$ that can be considered as the encoded image. In the actual implementation of the model, the encoder architecture will be the same as in the original CCVAE [1]: five convolutional layers with ReLU activations that allow dimension reduction, followed by a final linear layer. This particular architecture could easily be changed, but we have considered that it would not be of interest for this project.
- The decoder will perform the reverse process. It will take a latent vector $z \in \mathbb{R}^m$ as input and output an image $\bar{x} \in \mathbb{R}^{h\times w\times c}$. The architecture of the decoder mirrors that of the encoder, but in reverse. It begins with a linear layer, followed by five convolutional layers with ReLU activations (the reverse sequence of the encoder), and concludes with a final sigmoid layer.
- The conditional prior module takes as input a vector of labels $y \in [0,1]^L$, and outputs the mean μ_ψ and the variance σ_ψ^2 of a normal distribution on $\mathbb{R}^L$. From this distribution, it is possible to sample a vector of characteristics $z_C \in \mathbb{R}^L$. Note that a latent vector $z \in \mathbb{R}^m$ in the latent space can be split into the latent characteristics z_C and the latent style $z\backslash_C$, so $z = \{z_C, z\backslash_C\}$. Thus, the output of the conditional prior module can be related to the output of the encoder and the input of the decoder. This architecture does not have convolutional or linear layers. Instead, it uses parameter tensors to compute the distribution.
- The classifier receives a vector of characteristics $z_C \in \mathbb{R}^L$ and outputs an image $\bar{x} \in \mathbb{R}^{h\times w\times c}$, in line with the terms used in previous points. This module is composed of a learnable weight tensor and a learnable bias tensor.

In summary, there are four parametrized models to learn, the distributions $q_\phi(z|x)$ and $p_\psi(z_C|y)$, corresponding to the encoder and the conditional prior model, and the functions $p_\theta(x|z)$ and $q_\varphi(y|z_C)$, corresponding to the decoder and the classifier. Figure 1 illustrates these distributions.

Each module of the architecture will be trained simultaneously based on the loss function presented in the next section. Then the different modules of the architecture can be used separately for different tasks. For instance, the conditional prior module and the decoder can be used for image generation, the encoder and the classifier can be used for image classification, and the encoder, the conditional prior model and the decoder can be used together for modifying characteristics.

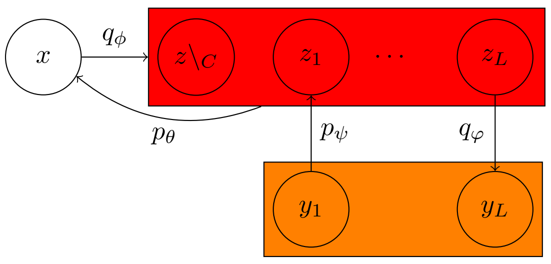


Figure 1: CCVAE graphical model.

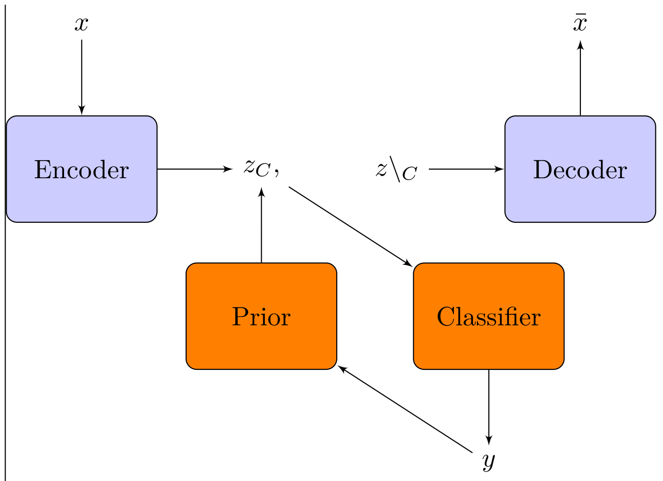


Figure 2: Model architecture.

5.0.3 Loss Function

As defined in [1], the loss function of the supervised CCVAE can be defined as

$$\begin{aligned}
L_{CCVAE} &= \mathbb{E}_{q_\phi(z|x)}\left[\frac{q_\varphi(y|z_C)}{q_{\varphi,\phi}(y|x)} log \frac{p_\theta(x|z)p_\psi(z_C|y)}{q_\varphi(y|z_C)q_\phi(z|x)}\right] + log\ q_{\varphi,\phi}(y|x) + log\ p(y) \\
&= \mathbb{E}_{q\phi(z|x)}\left[\frac{q_\varphi(y|z_C)}{q_{\varphi,\phi}(y|x)}\left(log\ p_\theta(x|z) - log\ q_\varphi(y|z_C) - log \frac{p_\psi(z_C|y)}{q_\phi(z|x)}\right)\right] + log\ q_{\varphi,\phi}(y|x) + log\ p(y) \\
&= \mathbb{E}_{q\phi(z|x)}\left[exp\left(log\ q_\varphi(y|z_C) - log\ q_{\varphi,\phi}(y|x)\right)\left(log\ p_\theta(x|z) - log\ q_\varphi(y|z_C) - D_{KL}(q_\phi(z|x)|p_\psi(z_C|y))\right)\right] \\
&+ log\ q_{\varphi,\phi}(y|x) + log\ p(y)
\end{aligned}$$

Where $q_\phi(z|x), p_\psi(z_C|y), p_\theta(x|z), q_\varphi(y|z_C)$ are the distributions and functions presented in the previous section and the prior distribution $p(y)$ is defined as the distribution of the attributes in the training set. It is needed for calculating the loss, and for our implementation, we simply calculate it once, computing the average value of each attribute taking into account the whole training set.

5.0.4 Datasets

We tested our model using three distinct types of datasets to assess its performance, identify limitations, and gain insights for potential enhancements:

- CelebA Dataset (Celebrities Attributes): It is a large-scale face attributes dataset containing over 200,000 celebrity images. Each image is annotated with 40 attribute labels.

- MNIST-Fashion: It consists of grayscale images of 10 different fashion categories.

- Medical MNIST Dataset: This dataset is a collection of medical images, including 6 different classes of medical images(Computed tomography of Abdomen, Head, ..etc).

5.0.5 Data Loader

We have introduced a data loader class for the CelebA dataset to fetch the images from their folder with their attributes from *attr.txt* file as a one-hot vector. then applied some transformations and we resized all the images to have the same shape (3x64x64) , we used a *partition.txt* file to split the data set into Train, Test, and Validation sets. However, we can do the split randomly. After that, we defined a DataLoader class with a batch size of 200 to iterate over the datasets. For the other dataset, we applied one-hot encoder on the labels and we used a standard data loader provided by PyTorch with batch size of 32 for MNIST fashion and Medical MNIST datasets.

5.0.6 Experimental Setup

We have trained the model many times over 20 epochs on the different data sets: CelebA, Mnist-Fashion, Mnist, and Medical MNIST to observe the change of the latent space. During this process, we track the performance of our model by calculating the loss and the accuracy of our classifier. We also feet an example to observe the changes in the reconstruction process after each epoch. We have used Adam as an optimizer for our model with a learning rate of 0.002, and we have fixed the size of the latent space to 45.

5.1 Results

We plotted the loss and the accuracy of our model after the training as illustrated in Figure 3), where we can observe the stability of the convergence of the model after each epoch such that it reaches the minimum after 8 epochs.

To visualize the performance of the reconstruction, we defined a function latent-walk such that we combine some attributes that affect the z_c in our latent space and observe the change in the images generated by the decoder as illustrated in the Figure 4.

Also, we plotted the heat map (Figure 5) of the combined confusion matrix for all the selected attributes to evaluate the performance of the model over each attribute.

Limits and Further Work

We have tested the model on some big datasets with many attributes (more than 35 attributes) where we have reached its limits such that we tried many times with different hyperparameters to train the model but it was not able to learn how to generate images even after more than 20 epochs.

As a suggestion for future work, we propose developing this model to perform effectively in a semi-supervised manner for classification tasks, such as data labeling. Additionally, further enhancements could focus on ensuring the model's capability to handle datasets with a high number of attributes.

Conclusion

During this project, we learned about the probabilistic properties of the Variational Autoencoder (VAE) and delved deeply into the paper [1], where the author introduced a new approach aimed at capturing the characteristics of the input data. Moreover, we successfully re-implemented this approach based on our understanding of the paper and the theoretical concepts covered in the Probabilistic Graphical Models course lectures.

In conclusion, we would like to note that our project yielded results comparable to those presented in the paper [1], and we look forward to further development in the future.

Author Contributions

Theory and code were studied and understood by all three authors. However, Biel and Aissa focused more on writing the notebooks while David focused more on theory and report writing.

References

[1] T. Joy, S. M. Schmon, P. H. S. Torr, S. Narayanaswamy, and T. Rainforth, "Capturing label characteristics in VAEs," in Proc. Int. Conf. Learn. Representations, 2021.

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

[3] C. Doersch, "Tutorial on variational autoencoders," arXiv:1606.05908, 2016.

[4] Tameem Adel, Zoubin Ghahramani, and Adrian Weller. Discovering interpretable representations for both deep generative and discriminative models. In International Conference on Machine Learning, pp. 50–59, 2018.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell., 35(8):1798–1828, August 2013. ISSN 0162-8828.

[6] Yang Li, Quan Pan, Suhang Wang, Haiyun Peng, Tao Yang, and Erik Cambria. Disentangled variational auto-encoder for semi-supervised learning. Information Sciences, 482:73–85, 2019.

A Appendix

Other interesting latent walk results can be observed in Figure 6 and Figure 7.

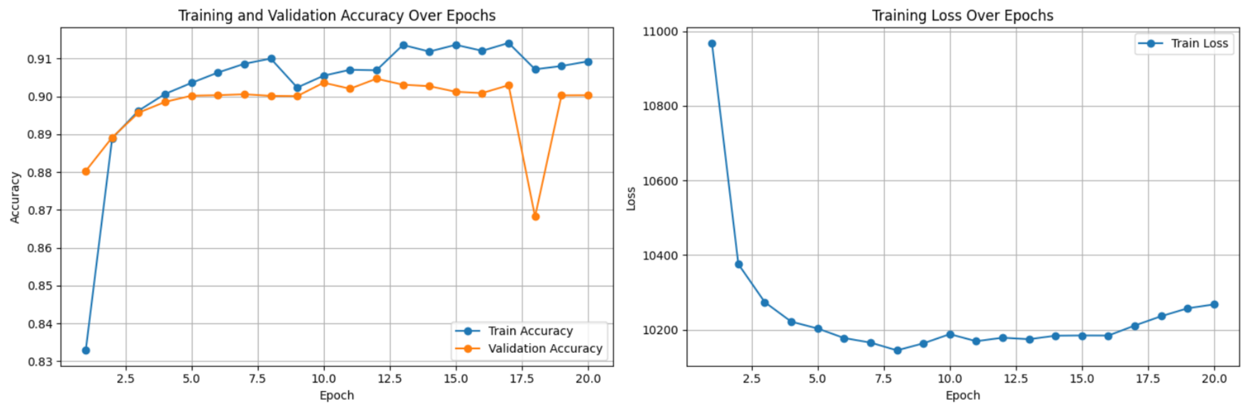


Figure 3: The performance of our model over CelebA Dataset

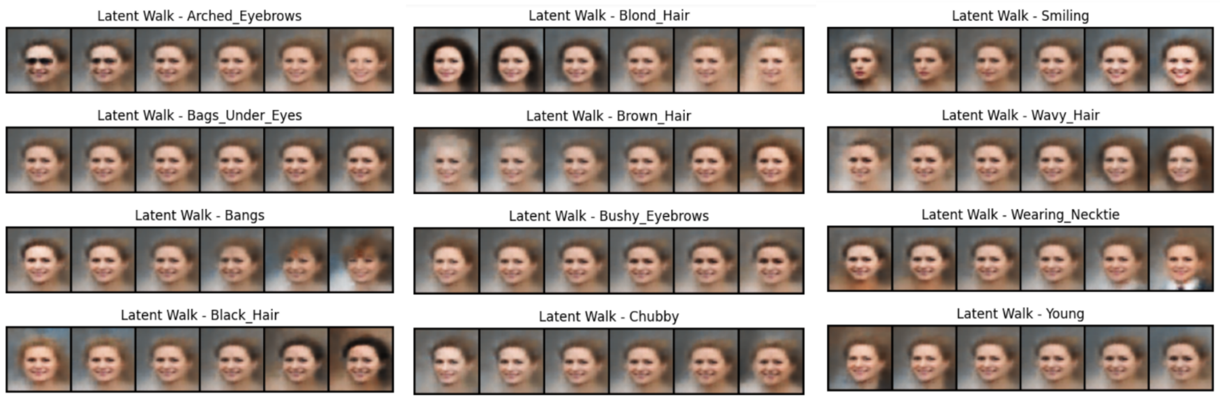
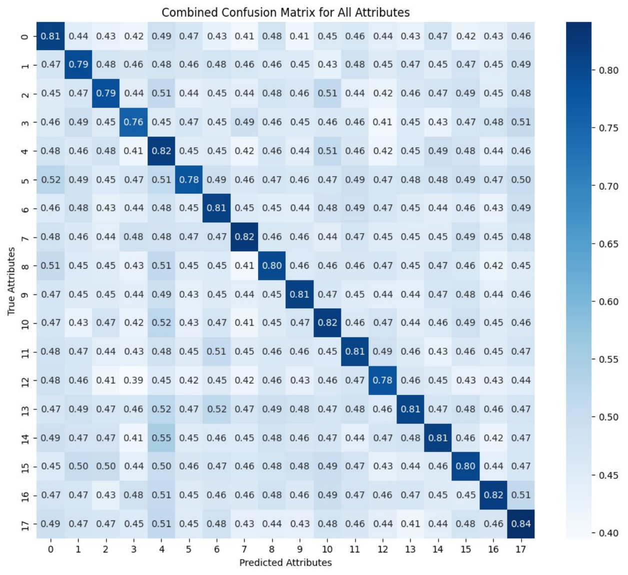


Figure 4: Latent Walk results over CelebA Attributes



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0.81	0.44	0.43	0.42	0.49	0.47	0.43	0.41	0.48	0.41	0.45	0.46	0.44	0.43	0.47	0.42	0.43	0.46
1	0.47	0.79	0.48	0.46	0.48	0.46	0.48	0.46	0.46	0.45	0.43	0.48	0.45	0.47	0.45	0.47	0.45	0.49
2	0.45	0.47	0.79	0.44	0.51	0.44	0.45	0.44	0.48	0.46	0.51	0.44	0.42	0.46	0.47	0.49	0.45	0.48
3	0.46	0.49	0.45	0.76	0.45	0.47	0.45	0.49	0.46	0.45	0.46	0.46	0.41	0.45	0.43	0.47	0.48	0.51
4	0.48	0.46	0.48	0.41	0.82	0.45	0.45	0.42	0.46	0.44	0.51	0.46	0.42	0.45	0.49	0.48	0.44	0.46
5	0.52	0.49	0.45	0.47	0.51	0.78	0.49	0.46	0.47	0.46	0.47	0.49	0.47	0.48	0.48	0.49	0.47	0.50
6	0.46	0.48	0.43	0.44	0.48	0.45	0.81	0.45	0.45	0.44	0.48	0.49	0.47	0.45	0.44	0.46	0.43	0.49
7	0.48	0.46	0.44	0.48	0.48	0.47	0.47	0.82	0.46	0.46	0.44	0.47	0.45	0.45	0.45	0.49	0.45	0.48
8	0.51	0.45	0.45	0.43	0.51	0.45	0.45	0.41	0.80	0.46	0.46	0.46	0.47	0.45	0.47	0.46	0.42	0.45
9	0.47	0.45	0.45	0.44	0.49	0.43	0.45	0.44	0.45	0.81	0.47	0.45	0.44	0.44	0.47	0.48	0.44	0.46
10	0.47	0.43	0.47	0.42	0.52	0.43	0.47	0.41	0.45	0.47	0.82	0.46	0.47	0.44	0.46	0.49	0.45	0.46
11	0.48	0.47	0.44	0.43	0.48	0.45	0.51	0.45	0.46	0.46	0.45	0.81	0.49	0.46	0.43	0.46	0.45	0.47
12	0.48	0.46	0.41	0.39	0.45	0.42	0.45	0.42	0.46	0.43	0.46	0.47	0.78	0.46	0.45	0.43	0.43	0.44
13	0.47	0.49	0.47	0.46	0.52	0.47	0.52	0.47	0.49	0.48	0.47	0.48	0.46	0.81	0.47	0.48	0.46	0.47
14	0.49	0.47	0.47	0.41	0.55	0.45	0.46	0.45	0.48	0.46	0.47	0.44	0.47	0.48	0.81	0.46	0.42	0.47
15	0.45	0.50	0.50	0.44	0.50	0.46	0.47	0.46	0.48	0.48	0.49	0.47	0.43	0.44	0.46	0.80	0.44	0.47
16	0.47	0.47	0.43	0.48	0.51	0.45	0.46	0.46	0.48	0.46	0.49	0.47	0.46	0.47	0.45	0.45	0.82	0.51
17	0.49	0.47	0.47	0.45	0.51	0.45	0.48	0.43	0.44	0.43	0.48	0.46	0.44	0.41	0.44	0.48	0.46	0.84

Figure 5: Confusion Matrix for the Selected Attributes

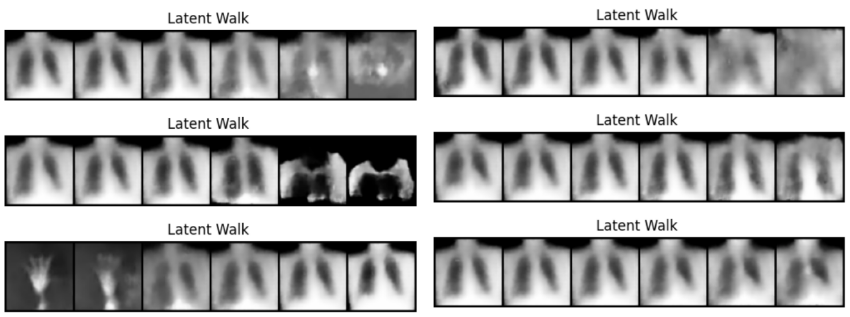


Figure 6: Latent Walk results over MNIST Medical Attributes

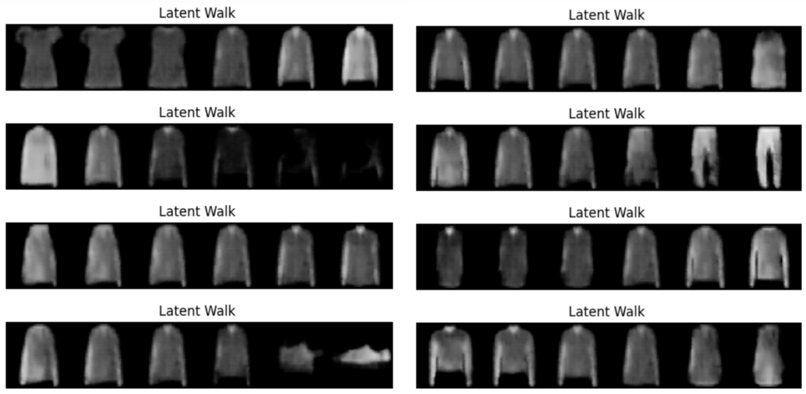


Figure 7: Latent Walk results over MNIST Fashion Attributes