Visualization Research Center
Allmandring 19
70569 Stuttgart

Master

Deep Learning for Scatterplot Similarity

Maalouly J.

Course of Study:	Computer Science
Examiner:	Prof. Dr. Michael Sedlmair
Supervisor:	MAG. Rene Cutura, M.SC. Cristina Morariu
Commenced:	15.05.2020
Completed:	15.02.2021

Abstract

Through the technology of generative models, one can now learn a distribution of his input data and generate new and similar output where it mimics the behavior of his input. Variational Auto-Encoders (VAE) are generative models where the neural network tries to learn a distribution of the latent vector z. Disentanglement using VAE's tries to decouple the latent vector and render each dimension independent from the other. Using disentanglement we wish to learn the three basic visual properties size, shape, and color. All the different disentangled models used in this paper aim to refine the reconstruction and disentangled representation. After training our models the user will have control over the three properties. Next, wish to create a tool where the user can control the structural properties of a scatterplot. By simply training a disentangled model, we would like to gain control over the different structures of a scatterplot to generate similar data. Furthermore, we will be discussing the limitations and shortcomings of using such models. The code for this paper could be found on https://github.com/jymaalouly/Thesis.

Contents

List of Figures

List of Tables

1 Introduction

Scatterplots have been the main visualization method to present multidimensional data where one specific dimension is paired with another displaying a relation between them. It is the best-known technique for visualizing bivariate data. Visualization of multivariate data is one of the most challenging tasks. When designing a scatterplot graph one must take into account the different visual encodings. A scatterplot matrix (SPLOM) is a symmetric matrix of pairwise scatterplots. A SPLOM contains the cross between all dimensions and their respective relations with a size of $\frac{n(n-1)}{2}$. SPLOMs can be efficiently used to create predictions detect anomaly's and find patterns between the different dimensions where a correlation is not that obvious. However, the sheer size of such matrices can cause problems. Finding an underlying structure becomes more and more difficult [DW14].

Many methods have been proposed for tackling these problems. In Revision, by *Savva et al.* [SKC+11] the system automatically outputs an improved design that aims for better perception. While Scatternet [MTW+20] uses deep learning to capture perception-driven similarities of scatterplots.

Over the last decade, there has been a substantial improvement in the field of deep learning outperforming humans in certain tasks. The Variational Auto-Encoders (VAE) are a bottleneck architecture where we encode our input vector into a latent vector and provide the ability to generate similar data through a Gaussian representation of our latent space. In disentanglement, we have seen that the same latent vector can also be trained to act independently. In other words, our model will try to learn to encode the most important features by projecting the input to a latent vector of a certain size n. By definition this means data loss is inevitable but what if we can learn these underlying structures and even better control them.

This paper is divided into two main parts. In the first part, we propose a method that can detect the varied visual parameters which provide some meaning to the human-perception. We want our system to generalize over a wide range of datasets and extract different features. The three visual factors considered for the first experiment are size, shape, and color. The intention is to give the user the flexibility of using a disentanglement model to control each factor independently. To sum up, our goal is to make the visualization process somewhat easier by extracting features that can increase the human-perception. Moreover, this will also give the user a clear idea of how the visualization will look like, and thus can choose the output that best conveys a specific pattern.

The second part of this paper adds to the work done by *Jo & Seo* [JS19] where they used a data-driven method to find the underlying structure of a scatterplot. We aim to create a tool that uses this model and allows the user to alter and generate new scatterplot images.

In this paper, we will be discussing the different models used to achieve our goal. In the first chapter, we will introduce the different concepts that led to the idea of our paper, we will state our motivation and the related works. The topic of the second section will deal with the extraction of the three visual

parameters. We will explain the different models used as well as the architecture implemented. We will go through the results and comparison of the different models and datasets, then end with our conclusion. The third chapter contains the details for creating a tool that encapsulates the structural components of a scatterplot. This will be achieved by training a disentangled model similar to *Jo & Seo* [JS19]. Last we will discuss the future works and research that address the shortcoming of these models.

1.1 Motivation

It is essential nowadays for users to present their data with a clear idea no matter the background of the user, and the power to interactively change a plot to maximize the perception of a specific pattern is very important.

One of the clear predominant issues in visualization is finding a good representation of your data to easily covey an idea. In experiment 1 we intended to learn the three basic visual features size, shape, and color using a generative model. Our main goal is to see if we can disentangle these features and gain the ability to change them using our generative model.

Many methods have been introduced that can measure the goodness of a graph, and the similarity and dissimilarity between the different scatterplots. We intend to use the VAE model to try to encapsulate the visual and structural features of a graph. Our goal is to create a tool where a user with minimal background and knowledge in visualization, has the ability to generate new and similar data on his own and even have control over these structures. The tool should be used intuitively so that a normal user can understand what is happening.

The tool can be used for both practical and educational reasons. In a practical sense, a user can either test different visual representations and choose which best fits his task, and in a more educational sense it can be used to illustrate The different structures in a visualization. The system could also be used to generate large amounts of similar data where the user can specify the parameters.

In order to achieve our goals, we will be using unsupervised generative models. We decided to go with VAE disentanglement over other generative models. The disentanglement model decouples the latent vector thus increasing the independence in controlling the output. This will give the user control over the encoded features, however, we will be discussing disentanglement in much detail in the upcoming chapters.

1.2 Related Works

To achieve our desired objective we need to acknowledge the work of our predecessors. Various techniques have been implemented to extract relevant information and re-visualizing the data in a way that best fits. This section will be divided into two parts, first, we will discuss the various tools and the different techniques used to achieve re-visualization. Second, we will explain some machine learning concepts that make up the core structure of this project.

1.2.1 Similarity Measures and Graphical Perception

Before we discuss such techniques we must first explain what constitutes a good visualization, or is there a way to differentiate between a good and a bad graph. What is the best way to increase the human perception of a certain figure, chart, plot, etc... In 'Understanding Charts and Graphs' [Kos89] the author not only explains the different types of visualization but also the pitfalls one must avoid in order to relate the information needed to the user, where any violations of the acceptability principles will lead to difficulties in conveying a message in a visualization.

Clevland & McGill [CM84] tries to identify a set of elementary perceptual tasks, and these tasks are then ordered based on how accurately they are performed by humans. Some of the perceptual tasks that humans use to extract qualitative data are position, color, volume, and length, etc... These elementary tasks can be performed simultaneously in order to extract the relevant information.

Nowell et al. [NH97] determines the effectiveness of the three most important data encodings; shape, color, size. These three encodings are capable of visualizing both nominal and quantitative data. For that reason, we decided in our first experiment to try to disentangle the three main visual encodings.

Visual encodings are one thing, however, the structural component of a scatterplot is something entirely different. In 'Graph-theoretic Scagnostics' [WAG05] *Wilkinson et al.* categorizes scatterplots into 9 different man-made features. These features provide a quantitative measure of their structure. The scagnostic features are divided into nine different categories, Outlying, Skewed, Clumpy, Convex, Skinny, Striated, Stringy, Straight, and Monotonic.

In ScagExplorer [DW14] The authors utilize the set of scagnostic features to guide the exploration of high dimensional data. The high complexity of a SPLOM makes outlier detections and pattern recognition an increasingly difficult problem. Using the scagnostic features ScagExplorer help locate anomalies and characterizes scatterplots for easier and interactive exploration.

There exist extensive research on the topic that aims to improve the way we can visualize certain plots. Based on the methods used we can say that these tools are usually a mixture of image and learning-based methods.

Savva et al. Revision [SKC+11] has the objective to allow the users to create more effective Visualization by simply providing an image as an input and then using the combined power of both computer vision and machine learning will generate an output that best describes the input data. Revision works by first classifying the type of chart, second by extracting data and storing them in relational tables, last creating a set of re-designed output that should better communicate the data.

Chen et al. [CKNH20] SimCLR is a framework for contrastive learning of visual representations. The software tries improve upon the learned representations using a contrastive loss between the different representations. SimCLR uses data augmentation to improve the predictive result. The contrastive learning algorithm works by maximizing the agreement between the augmented representation of the same data. The augmented data for example can be taking the original image and applying a rotation, color distortion, Gaussian blurring, and so on.

A reverse process has been implemented by *Poco & Heer* [PH17] where the input is an image and the output is a JSON file specifying the different visual encodings and data. The first step is to detect the text elements in a chart and each element will then be classified based on their roles. The next step is training a convolutional neural network to identify the different mark types. The system then maps the different mark types to the different text elements and labels.

A similar work has been done by *Harper & Agrawala* [HA16]. The aim was to take a D3 chart and extract its structure including the data, their mark types, and mappings. all deconstructed mappings are then ranked on their perceptual effectiveness. Using the previous steps a style template is constructed and later on re-used to create new visuals by mapping(the most important data fields to the most perceptual effective mapping).

Beagle [BDM+18] answers the question of what is the most predominant visualizations and representation by crawling the web and extracting around 41000 SVG-based visualizations. The tool classifies the different visualizations with 85% accuracy with 24 different types of visualizations. The least favored of visualization and the rarest was the pie chart. On the other hand, the most popular were bar charts, line charts, and then scatterplots. Another similar research was done by *Hoque & Agrawala* [HA20]. This paper presents a search engine specifically for D3 visualizations that provides an output based on the specified structure and style given in a query. The engine is built from 7860 deconstructed visualizations. The extracted features were both of data and non-data-encoding. These features were then indexed for each visualization so it can be later called upon and reached.

Jung et al. [JKS+17] presents ChartSense a data extraction tool that uses deep learning for categorizing the different types of charts and then based on the chart types it uses an interactive data extraction mechanism specifically tailored for that specific type.

Similarly, *Mackinlay* [Mac86] creates a tool in which it takes relational data and outputs a graphical representation. It addresses two main problems. First, the output should be able to represent a large amount of information. Second, the graphic design that should be used must convey the data at hand and maximize its effect.

In this paper we will not be using any discriminate models. We chose to implement a generative model where we will depend on our model to encode our features.

1.2.2 Generative Models

In this paper, we use a generative model to reconstruct our input and generate new and similar images. We cannot discuss generative models without the mention of Variational Auto-Encoders (VAE) [KW14]. Let z denote the latent vector that encodes an image $\boldsymbol{x}$. A perfect reconstruction from z results in an image $\boldsymbol{x'}$ that is identical to $\boldsymbol{x}$. What we just described was an Auto-Encoder (AE) [HS06] which is typically a bottleneck neural network consisting of two main parts the encoder and the decoder. The encoder is responsible for reducing the dimensionality of an input to a latent vector z to a specified dimensionality. The decoder is responsible for taking the latent vector z as input and reconstructing an output $\boldsymbol{x'}$ with minimum loss. A VAE unlike AE is a Generative model that will learn a distribution N, that way one can draw a random sample z from a distribution N.

The concept of disentangled representation is explained by *Higgins et al.* [KW14] and we quote from the paper the following explanation:

> *"vector representation is called a disentangled representation with respect to a particular decomposition of a symmetry group into subgroups, if it decomposes into independent subspaces, where each subspace is affected by the action of a single subgroup, and the actions of all other subgroups leave the subspace unaffected." - Higgins et al.* [KW14]

In laymen terms, there is no general explanation for a disentangled representation but what we could sum up is that changing one dimension in z will result in the change of one factor in the reconstructed image without it affecting the other factors.

There are many examples of disentangled models. Generative adversarial network (GAN) [GPM+14] is a generative model consisting of two main parts, the generator g and the discriminator d. A generator is responsible for creating new images from the latent vector z, and it is the job of the discriminator not to be fooled and capture the flaws. InfoGAN by *Chen et al.* [CDH+16] is an extension to GAN, where the author tries to learn a disentangled representation in an unsupervised manner.

In this paper, we opted to use the β-VAE model by *Higgens et al.* [HMP+16] where it extends upon the regular VAE model by adding a hyperparameter β to control the disentangled representation. We chose to work with β-VAE since it outperformed other disentanglement models and showed better results.

2 Experiment 1

Let us discuss what we hope to achieve by the end of this experiment. Our goal is to input an image of a scatterplot, and using our model, we want to encode and disentangle the three main design features size, shape, and color [NH97]. In this section, we will go through the whole process. We will first explain the different variations of disentanglement models. We will explain the different metrics used to measure the accuracy of our disentangled representation. Next, we discuss the different datasets used to train our models as well as the implemented architecture. We will go through the results of each dataset, their latent traversals, and a short discussion about the conducted test. Last we will compare the results of all trained datasets followed by our conclusion.

2.1 Overview

The techniques used in this paper are all extensions of the unsupervised VAE models. Previously we have discussed VAE, similar to an AE but it also learns a distribution $\boldsymbol{N}$, and from that distribution, one can sample a latent vector z. Using z as input to the the decoder we can now reconstruct an image $\boldsymbol{x'}$ [KW14].

Disentanglement models have the objective of encoding an independent latent vector where changing one dimension of that latent vector $\boldsymbol{z}$ will result in the change of that one factor alone. We will now explain three main disentanglement models; β-VAE [HMP+16], Factor VAE [KM19], βTCVAE [CLGD19]. We will look at their corresponding objective functions and get a brief idea of what each model offers.

We will start by the simplest model which is β-VAE model. This introduces a new hyperparameter β which is added to the VAE objective function:

$$L(\theta, \phi; x, z, \beta) = E_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta D_{KL}(q_\phi(z|x)|p(z)) \tag{2.1}$$

As we can see from the function above the β-VAE learns the disentangled representation by applying a penalty over the KL-divergence [Joy11]. Placing more weight on the KL-divergence will enforce our posterior or our latent approximation $q_\phi(z|x)$ to be closer to the prior $p(z)$. What was observed, that although β-VAE will result in a disentangled representation, it does on the other hand fail in providing a good reconstruction. For a disentangled representation β must be chosen greater than one. If $\beta = 1$ the result will be equivalent to a VAE model.

The issue of the blurry reconstructions given by the β-VAE model is addressed in the Factor VAE [KM19] and βTCVAE [CLGD19] where both models set out to penalize the total correlation.

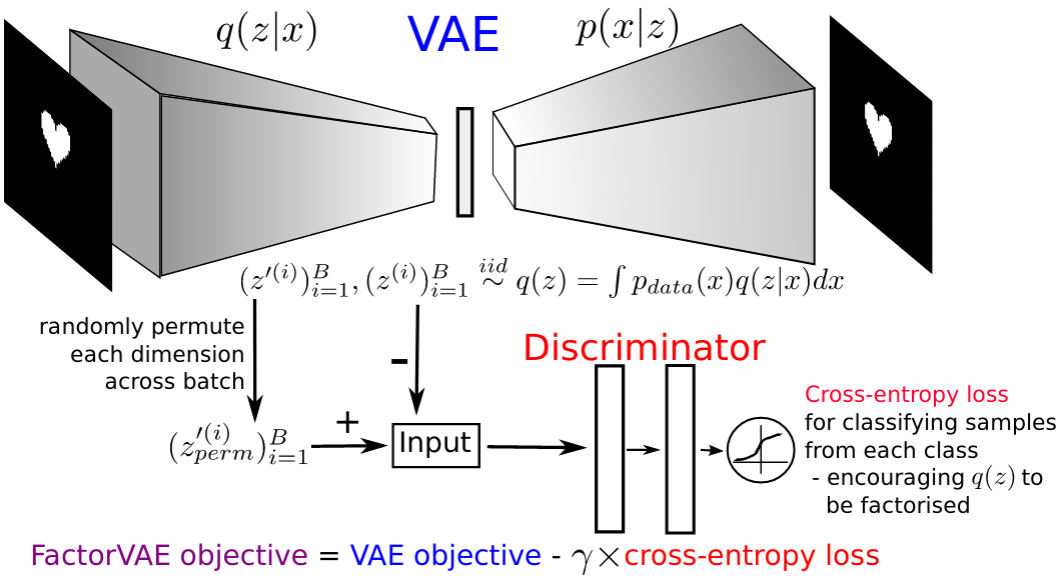


Figure 2.1: Factor VAE architecture as described by *Kim & Mnih* [KM19].

To understand what is causing the blurry reconstruction we look into what we are penalizing in equation 2.1. *Hoffman & Johnson* [HJ] provided a decomposition of the second part of equation 2.1:

$$E_{p(x)}[D_{KL}(q_\phi(z|x)|p(z))] = I(x;z) + D_{KL}(q(z)|p(z)) \tag{2.2}$$

Where $I(x;z)$ is the mutual information between x and z. What this equation implies that the model provided by β-VAE is penalizing both terms. Rather than just disentangling $q(z)$ and $p(z)$ The model is also penalizing the first term, the mutual information, which is not desirable for a good disentangled representation.

$$\frac{1}{N}\sum_{i=1}^{N}[E_{q(z|x^{(i)})}[\log p(x^{(i)}|z)]D_{KL}(q(z|x^{(i)})|p(z))] - \gamma D_{KL}(q(z)|\bar{q}(z)) \tag{2.3}$$

Where $q(\bar{z}) := \prod_{j=1}^{d} q(z_j)$. Equation 2.3 represents the objective function given by the Factor VAE. What the paper tries to explain is that a vector is considered disentangled or independent if the product of the individual marginals of the latent vector is equal to the marginals of the latent vector. The total correlation is described as the KL-divergence [Joy11] between both marginals. Equation 2.3 penalizes the KL-divergence between both probabilities by using the hyperparameter γ.

Figure 2.1 shows the architecture of a Factor VAE model. The figure shows that the Factor VAE uses a discriminator network for the disentanglement process.

$$E_{p(n)}[D_{KL}(q(z|n)|p(z))] = D_{KL}(q(z,n)|q(z)p(n)) + D_{KL}(q(z)|\prod_{j}^{d} q(z_j)) + \sum_{j} D_{KL}(q(zj)|p(zj)) \tag{2.4}$$

Similar to Factor VAE, β-TCVAE also computes the total correlation but rather than using a discriminator it is computed probabilistically. The first part of equation 2.4 is the mutual information between the latent variables z and the data variable x, while the second part describes the total correlation, and the last part is the dimension-wise KL-divergence of the marginal posterior and the prior.

2.2 Dataset

2.2.1 dSprites

In this experiment, we shifted through 4 types of datasets. The first dataset we tested was the dSprites dataset [MHHL17]. dSprites is a set of images each contributing to the following factors color, shape, scale, rotation, x, and y positions. The latent factor values are as follows:

- Shape: square, ellipse, heart,
- Scale: 6 values linearly spaced in $[0.5, 1]$,
- Orientation: 40 values in $[0, 2\pi]$,
- Position x: 32 values in $[0, 1]$,
- Position y: 32 values in $[0, 1]$.

Each latent value will be varied once at a time and ensuring that only one combination is made. The total number of images is relative to the number of factors and their values. dSprites consists of 737280 unique black and white images of size $64 \times 64 \times 1$.

2.2.2 Synthetic Data

As previously discussed, after testing the model with the dSprites dataset we can now start conducting our own research. We started by creating a set of images with the following factors:

- Size: 6 values linearly spaced,
- Shape: circle, square, pentagon, hexagon, Filled x, diamond,
- Color: 5 values linearly spaced in $[0, 1]$.

This will result in 180 unique images of size $64 \times 64 \times 3$. The scatterplot images were generated using Matplotlib [Hun07] with 5 data points guassianly distributed between 0 and 1.

The second iteration of this dataset is a wide variety of different plots with a larger number of samples. After each couple of iterations we would increase the number of data points scattered. The same technique as before was used to populate our scatterplots. Nonetheless by increasing the number of images we had to add one more factor to preserve the condition of having unique images. The factor added was the position, which represented every different plot generated. The position factor included 250 variables resulting in 45000 unique images.

2.2.3 Real World Data

Using synthetic data was not enough. In order for our model to perform better reconstructions and have the capability to learn more we must create a dataset composed of real data. We created our real world dataset using R Dataset Package [R C18]. To create this dataset we have discarded any combination containing categorical, or qualitative data. Any duplicate plots were also discarded in order to ensure synthetically generated plots are not included. All data values were normalized to a value ranging between 0 and 1. The Real World Data contained 56520 unique images consisting of 3 channels with the position, size, shape, and color being 314, 6, 6, and 5 respectively.

2.3 Metrics

There is still no standardized method for measuring the success of a disentangled representation, however, we will discuss some of the metrics that have been considered. In this research, we will assess our disentangled representation using BetaVAE score [HMP+16], FactorVAE score [KM19], and Mutual Information Gap [CLGD19].

The first metric we will use is the BetaVAE metric [HMP+16]. what this metric proposes is to fix a random component of our latent vector and use our generative model to sample two mini-batches from the latent vector. A linear classifier will try to predict the index of each factor. The accuracy of the classifier will determine the value of the disentangled metric. The training is done on 10 000 points and test on 5000.

The second metric considered is the FactorVAE [KM19]. This metric is almost inversely equivalent to the BetaVAE score. This metric measures the disentangled representation by predicting the index of the ground truth factor from the latent vector with least variance. Training is done by using a majority vote classifier. We train over 10 000 points and evaluate over 5000.

The Mutual Information Gap (MIG) [CLGD19] is the last metric considered in this experiment. Both the above metrics are not general and fail to be unbiased since they both depend on some hyperparameters. We compute the mutual information gap which is the average of the normal difference between the highest and second highest mutual information of each factor.

As suggested by *Locatello et al.* [LBL+19] we will traverse our latent dimensions, that way we can relate each dimension to the factor it represents.

Encoder	Decoder
$64 \times 64 \times C$	Latent Vector
4×4 conv, 32 ReLU, stride 2	FC, 256 ReLU
4×4 conv, 32 ReLU, stride 2	FC, $4 \times 4 \times 64$ ReLU
4×4 conv, 32 ReLU, stride 2	4×4 upconv, 32 ReLU, stride 2
4×4 conv, 32 ReLU, stride 2	4×4 upconv, 32 ReLU, stride 2
FC, 256 RELU	4×4 upconv, 32 ReLU, stride 2
	4×4 upconv, 32 ReLU, stride 2

Table 2.1: Architecture used for experiment 1 as defined by *Locatello et al.* [LBL+19].

2.4 Model Training

In this section, we will discuss the different models used to achieve our goal. We will be using the three main disentanglement models, β-VAE, Factor VAE, and βTCVAE. Our tests will be conducted on four different datasets.

We start our research by first setting up the disentangled library provided by [LBL+19]. The library makes the disentangling process much easier and feasible to control. It also permits us to conduct our own research. After all needed dependencies were installed, it was time to run the first test.

This section is divided into five parts. First, we will walk through the architecture and the general hyperparameters. The next sections will be dedicated to the different tests conducted on the different datasets, where each dataset will be given its own section. Moreover, we will be comparing the results of our tests and finally, we will arrive at the conclusion of Experiment 1.

2.4.1 Architecture

We followed the architecture done by *Locatello et al.* [LBL+19]. The encoder takes an input of $64 \times 64 \times C$, where C is the number of channels. This is followed by two layers of 4×4 convolutions with 32 filters and ReLU as the activation function with stride 2. The next two layers are the same as before but have 64 filters. The last layer is a fully-connected(FC) layer of 256 neurons. We extract from the last layer the mean representation and the log variance. The decoder part is very similar except it is the reverse process of the encoder model.

The Discriminator network needed for the Factor VAE consists of 6 FC layers of 1000 neurons and a leaky ReLU activation function. This is followed by an FC layer of size 2. Table 2.1, 2.2, and 2.3 shows the model architecture, the discriminator architecture, and the list of fixed hyperparameters.

Discriminator
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 2

Table 2.2: Discriminator architecture as suggested by *Locatello et al.* [LBL+19]

Hyperparameter	Value
Latent dimension	10
Batch size	16
Optimizer	Adam
Decoder	Bernoulli
Learning rate	0.0001

Table 2.3: Fixed hyperparameters

2.4.2 dSprites Dataset

Even though the dSprites dataset does not help to achieve our main goal, however, it was the first step in testing the disentanglement library [LBL+19], the different models, and traversals. We ran our test with both the β values of 16 and 1. For $\beta = 16$ we expect a trade-off between the quality of the reconstruction and the disentanglement score, while for $\beta = 1$ we would end with a fully dependent latent vector but with better reconstructions.

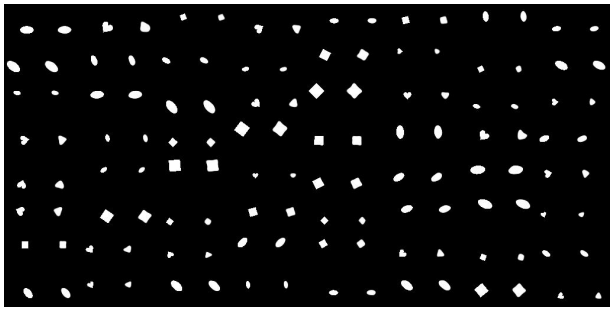
We already specified the fixed hyperparameters in table 2.3, the only varied parameter is the β value. Our first test was done for $\beta = 1$, the reconstruction loss was 23.45 while the KL loss was at 27.14. On the other $\beta = 16$ showed a much higher reconstruction loss of 72.217 and a lower KL loss of 6.45.

	Reconstruction loss	BetaVAE score	FactorVAE score	MIG
$\beta = 1$	23	0.771	0.49	0.019
$\beta = 4$	50	0.968	0.8476	0.102
$\beta = 16$	72	0.983	0.771	0.111

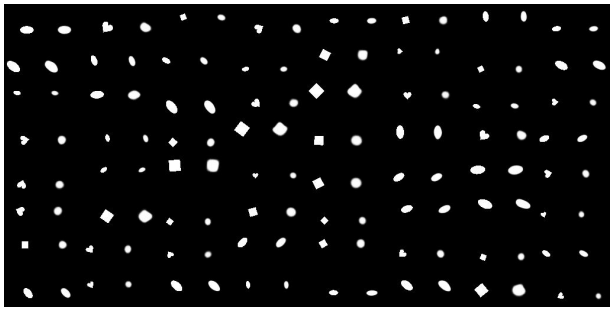
Table 2.4: Results of the dSpirtes dataset using a β-VAE model.

Table 2.4 shows the detailed results of our tests. We will not go into further details for this experiment since this is just for testing purposes. The β hyperparameter controls the trade-off between a proper reconstruction and a good disentangled representation. The β hyperparameter is a penalizer in the objective function. If we set the β value just right can we get a good disentangled representation while also preserving our reconstruction. For the exact purpose of conserving both reconstruction and disentanglement, we set the β hyperparameter to 4.

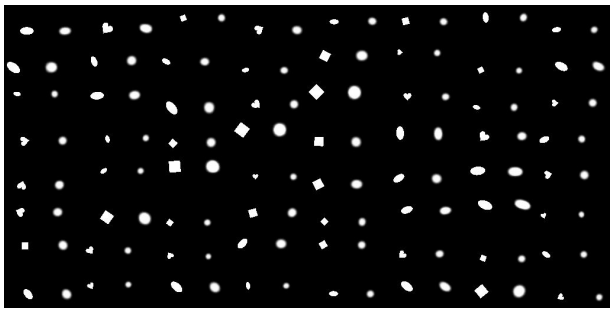
The $\beta = 4$ performed as expected. The KL loss was 13.07 while a reconstruction loss of 50.78, this clearly demonstrates the trade-off. The reconstruction loss was higher than $\beta = 1$ but lower than $\beta = 16$.



(a) $\beta = 1$



(b) $\beta = 4$

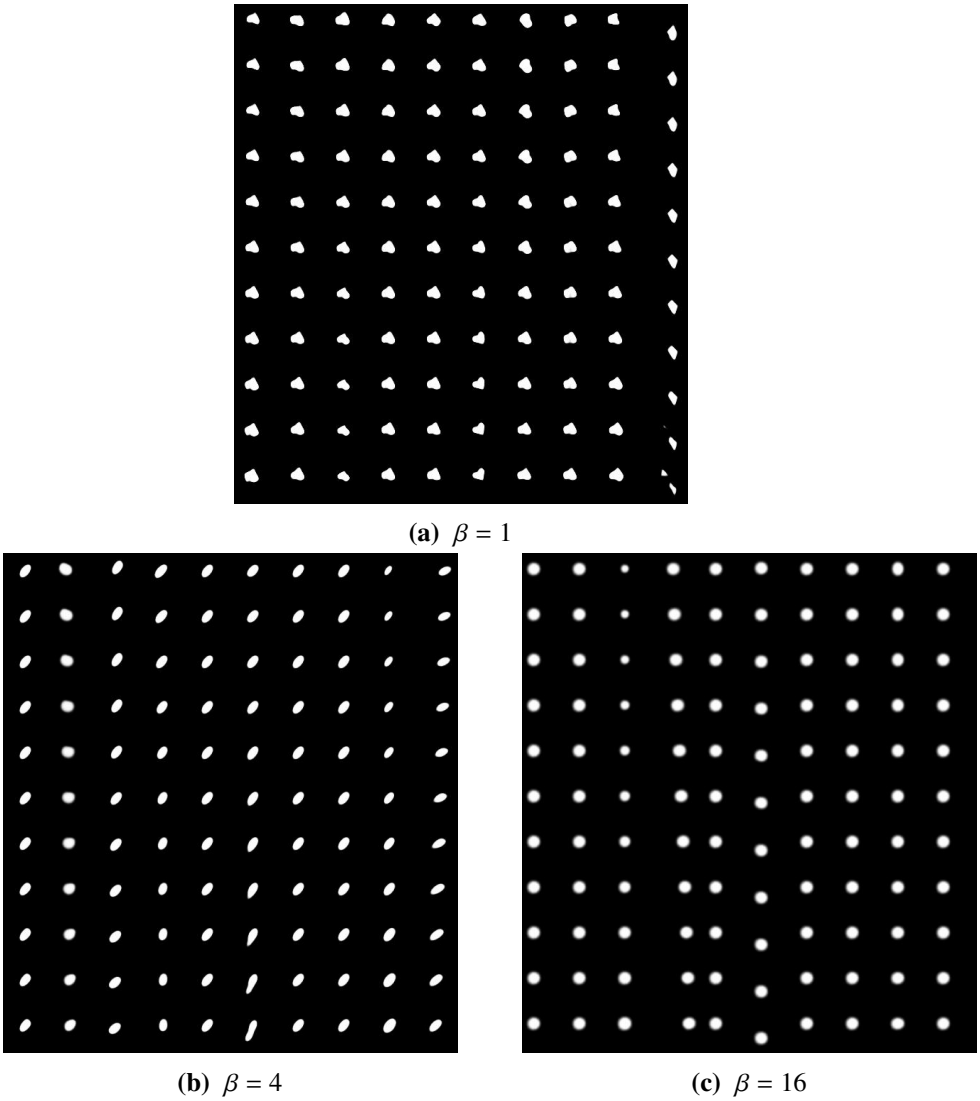


(c) $\beta = 16$

Figure 2.2: Each row contains a set of images where every two columns represent the original sample and its reconstruction. (also for the next figures)

Latent Traversals

In this section we traverse our latent vector with a range $[-2.5, 25]$. Each dimension is traversed once a time while the other dimensions remain fixed. As seen in Figures 2.2 and 2.3 the model with the parameter $\beta = 1$ gave much better reconstructions. Figure 2.2a shows more details and objects were distinguishable while the reconstruction of $\beta = 16$ shows blurry objects. For $\beta = 1$ the model did provide a good reconstructions, however, at a cost. For $\beta = 16$ we see how the objects preserve their shape while traversing the latent vector which is not evident for $\beta = 1$. The model $\beta = 4$ showed an average result with respect to the two other models.



(a) $\beta = 1$

(b) $\beta = 4$

(c) $\beta = 16$

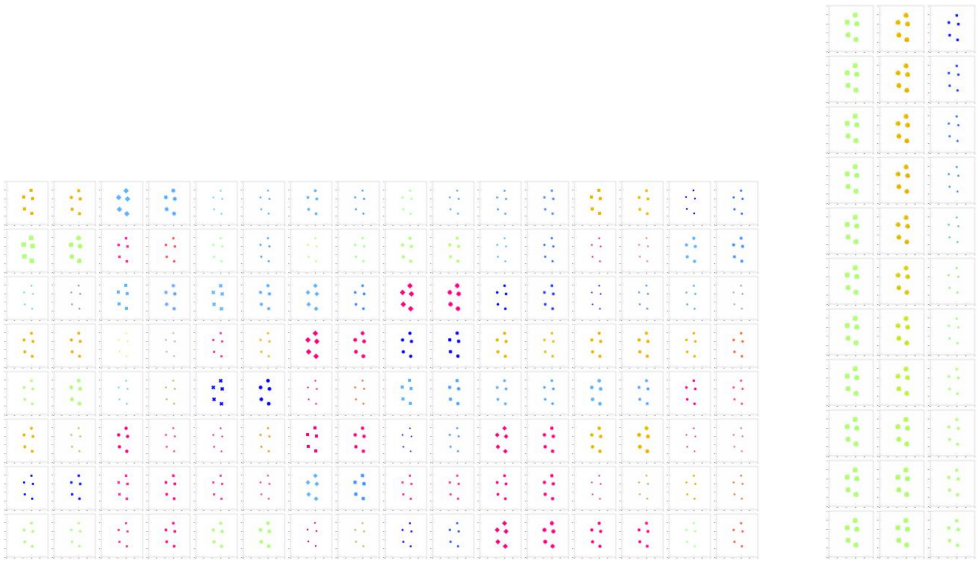
Figure 2.3: Columns represent the factors in the latent vector; Rows are the traversal from $[-2.5, 2.5]$.

Discussion

This test has proved successful but on a dataset that is not ours. We did not go into detail about the measures of our metrics since this is not the purpose of this dataset. For now, we can continue further in our research. Of course improvements in our model, especially for the reconstruction, will be discussed in the following tests. Let us conclude our first test, and continue on to the next steps where we will test our model on a single set of scatterplots.

Scores	Value
Reconstruction loss	469
BetaVAE	0.758
MIG	0.044

Table 2.5: Results of the initial dataset using a β-VAE model.



(a) Reconstructions **(b)** Traversals

Figure 2.4: Reconstructions and traversals of the initial dataset.

2.4.3 Initial Dataset

The premise of the second test is a simple one. We previously created a very simple dataset consisting of a set of varying factors, 6 for size, 6 for shape, and 5 for color. We wanted to test the disentanglement library on a dataset that we provided. After a thorough inspection of how the library works, we were able to insert the dataset or any other dataset for that matter into the library. The inputs for the following test were 180 images of size $64 \times 64 \times 3$, and a list containing the factors with their corresponding index value for each image.

As previously discussed this experiment is only made to test the feasibility of the library on a foreign dataset. Therefore we only trained our model once with 100 000 steps and a $\beta = 16$.

Since this is a very small dataset we changed one of the fixed hyperparameter, the latent dimension, to a size of 3. Table 2.5 shows the results of the reconstruction.

	Reconstruction loss	BetaVAE score	FactorVAE score	MIG
β-VAE	1304	0.575	0.509	0.008
Factor VAE	1274	0.791	0.616	0.179
βTCVAE	1292	0.77	0.547	0.189

Table 2.6: Results of the synthetic dataset using the three models with 100 000 training steps.

	Reconstruction loss	BetaVAE score	FactorVAE score	MIG
β-VAE	1341	0.579	0.590	0.01
Factor VAE	943	0.77	0.691	0.2
βTCVAE	987	0.739	0.528	0.176

Table 2.7: Results of the synthetic dataset using the three models with 300 000 training steps.

Latent Traversals

Looking at figure 2.4b we see the latent traversals for three latent dimensions. We can clearly see the second column representing the color factor while the third column representing the size. However, the shape factor seems to be missing. The first column must be the shape factor but the traversal over that particular dimension was not distinguishable.

Discussion

The traversals show that only two factors, size, and shape, were disentangled while the size factor theoretically must be disentangled. What we concluded is that the quality of reconstruction was affecting our traversals rendering the shapes blurry and indistinguishable. For the next test, we thought of addressing the issue with the blurry reconstructions by adding the Factor VAE and βTCVAE models. Now since we have tested our initial dataset and the disentanglement library, we can now start training on a larger dataset.

2.4.4 Synthetic Dataset

In this section, we will discuss the tests applied for the synthetic dataset, its main issues and problems that arise from our implementation. We try to address two different problems. The first issue is the need for generalization. Second, the lack of good reconstructions. The problem with generalization seemed to be simple at first. Our solution was to create a new dataset with varying samples and positions. However, this solution had a major flaw which we will discuss later on. As for the second issue we will start training our data on 2 new models that exist to solve the blurry reconstructions caused by the hyperparameter β.

Going through this test we realized that our dataset needs one more factor. The result from our initial datasets showed some promising results. The model was able not only to distinguish the different factors but also to disentangle our latent dimension. What our previous dataset fails to

do was generalize over a bigger range of scatterplot images. So we thought in order to solve this problem we should create more samples and iterate the styling factors over these samples. In our initial dataset, we had a total of 180 unique images each representing a different combination of those factors. We created a dataset of 250 different samples and we iterated each one of those samples over the three original factors size, shape, and color. The resulting dataset was 45000 different images. As for the factors, we added a position dimension that represents the different shapes of our samples. The factors position, size, shape, and color had the values 250, 6, 6, and 5 respectively.

For the following test, we used a β value of 16 and we ran it over 100 000 steps. We used the hyperparameters specified in table 2.3. Now let's take a look at the results of this test and the metrics. We examine first the reconstruction loss where the value sits at 1304.93 and the KL loss 2.41. For the BetaVAE score, we have a training accuracy of 0.59 while the evaluation accuracy is 0.575. Moreover, results for the FactorVAE showed a score of 0.509 for the training accuracy and 0.521 for the evaluation accuracy while the MIG score has a value of 0.008.

Previously we have discussed how β-VAE works. As β grows as do the improvements with the disentangled representation, however at a cost. While penalizing the KL divergence of $q(z)$ and $p(z)$ we are also penalizing the mutual information $I(x;z)$.

In order to try and fix this issue, we have considered two models. The Factor VAE and βTCVAE. The main goal of both models is to compute and penalize the total correlation, and while the Factor VAE uses a discriminator network, βTCVAE uses a probabilistic method. We tested both models with similar hyperparameters as before. We recall the latent dimension being 10 and the reconstruction loss for the decoder being a Bernoulli loss function. The only difference is that we omit the β hyperparameter. We include a new hyperparameter γ for Factor VAE set to 40 and a β_{TCVAE} hyperparameter set to 8 for the βTCVAE model.

The Factor VAE model gave a result of 1274 for the reconstruction loss while the KL loss was 10.726. The BetaVAE score did improve with a training accuracy of 0.81 and an evaluation accuracy of 0.791. The FactorVAE score also showed a bit of improvement with the training accuracy scoring at 0.62 and the evaluation accuracy at 0.616. The last metric, MIG, showed a value of 0.179.

The βTCVAE model showed a reconstruction loss of 1292.15 and KL loss of 3.954. The BetaVAE score showed a slightly lower value of training and evaluation accuracy than the Factor VAE model. The resulting training accuracy is 0.792 and evaluation accuracy at 0.77. The FactorVAE score was at 0.551 for training accuracy and 0.547 for evaluation accuracy. The MIG score showed a slightly higher value than the previous model with 0.189 for the discrete MIG. Table 2.6 gives a summation of all results for the three models over 100 000 steps.

Next, we thought of increasing the number of training steps from 100 000 to 300 000 steps. The hyperparameters all remained the same as the previous test. we first start by checking the results for the β-VAE model.

The reconstruction loss was 1341 and KL loss at 2.47 which is higher than the previous test. We can say that increasing the training steps for the β-VAE model did not help with our reconstructions. The BetaVAE score has a value of 0.594 for the training accuracy and 0.579 for the evaluation accuracy, while the FactorVAE score showed a value of 0.516 and 0.59 for the training and evaluation accuracy respectively. The last metric, the MIG metric showed a value of 0.01 for the discrete MIG.

Now we look at the results for the Factor VAE Model. The reconstruction loss sits at 943.8 while we have a score of 20.22 for the KL loss. The BetaVAE score showed a slightly lower training and evaluation accuracy than before with a score of 0.768 and 0.77 respectively. The FactorVAE score was slightly higher than before with a score of 0.688 and 0.691 for the training and evaluation accuracy. The MIG showed some improvement going from 0.179 from the previous test to 0.2.

The last model will be βTCVAE model. The reconstruction loss was at 987.116 which is lower than the Factor VAE reconstruction loss. As for the BetaVAE score, we have a training accuracy of 0.754 and 0.739 for the evaluation accuracy. The FactorVAE score got a result of 0.543 for training accuracy against a score of 0.528 for the evaluation accuracy. Last the MIG got a score 0.176 being outperformed by Factor VAE.

Latent Traversals

Now that we have the results from our test, let us check if the results match the reconstructions and traversals. Figure 2.6 shows the latent traversals of our latent dimensions while figure 2.5 shows the reconstructions of certain images from our dataset. You might notice that we see both the disentangled size and color factors, however, the position and shape are not represented in any of the models. What we concluded was that both the shape and position are heavily influenced by the quality of reconstruction. If the model is not able to adequately reconstruct the different shapes and positions it will not be able to disentangle these factors. Another reason for the failed disentanglement of the position factor might be attributed to the way the dataset images were populated.

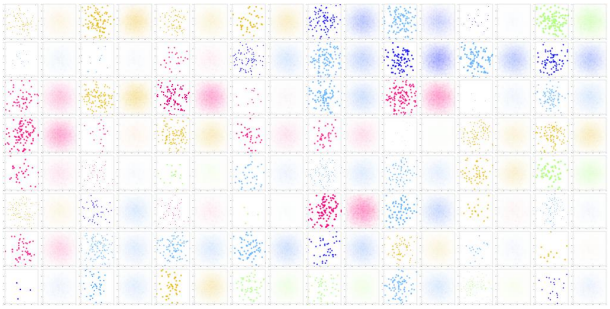
The reconstruction loss for the β-VAE model scored a very high value with figure 2.5a evident of this score. The β-VAE model shows a very blurred reconstruction of the original images and almost all reconstructions look exactly the same but in a different color. Based on first impressions both figures 2.5b and 2.5c show better result than the β-VAE model but not quite so. We can see a bit more details in our reconstructions but still can't get distinguishable shapes and structures. While it is really hard to say based on the reconstructions if the Factor VAE outperformed the βTCVAE model, nearly both models resulted in the same quality of reconstructions and both were able to provide a good disentangled representation.

Figures 2.7 and 2.8 show the reconstructions and traversals of our models but with 300 000 training steps. We can clearly see that the β-VAE model was massively outperformed by the other two models. The Factor VAE and βTCVAE gave pretty good reconstructions where now structures can be distinguished. We can now see a significant improvement in our reconstructions.

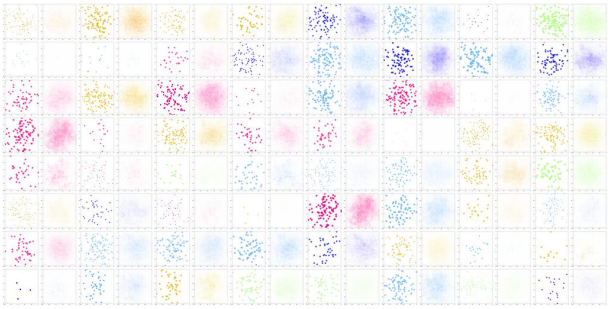
As per the traversals the βTCVAE is the clear winner. The color size and position factors are all clearly represented while the Factor VAE falls a bit short.

Discussion

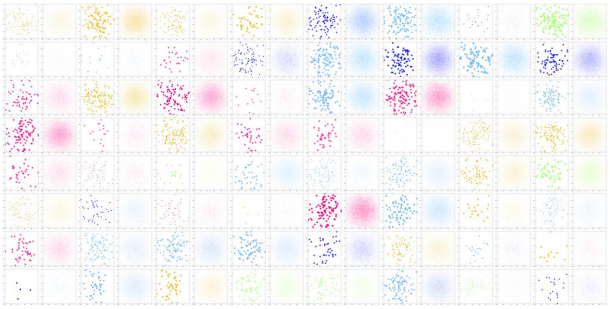
Now that we have completed our tests, is there a model that outperformed all others. It is easy to say that the β-VAE was outperformed by both the Factor VAE and the βTCVAE. The β-VAE model failed to give us a reconstruction of our data due to the nature of its objective function. In our case, it is clear that the β-VAE is not the correct model to consider.



(a) β-VAE



(b) Factor VAE

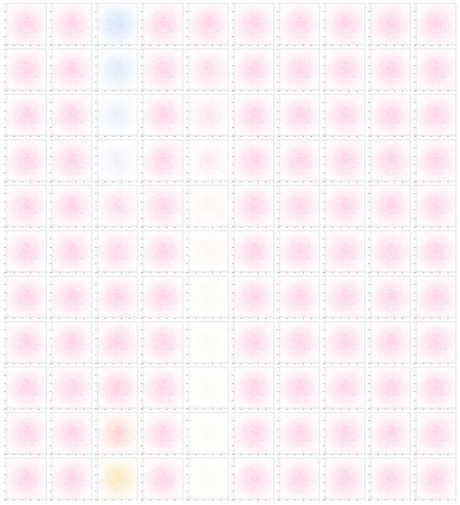


(c) βTCVAE

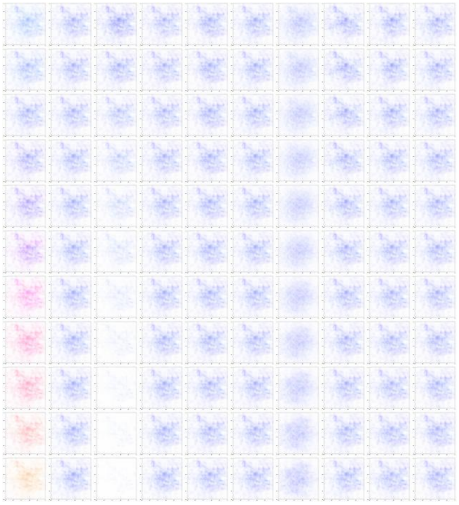
Figure 2.5: Reconstructions from the three different models with 100 000 training steps on synthetic dataset.

It is tough to say which of the two remaining models outperformed the other. Although based on the metrics alone we can see that the Factor VAE model outperformed the βTCVAE model, however, these metrics are not standardized metrics and multiple tweakings can be done with hyperparameters that can achieve better results. However, judging by the traversals alone the βTCVAE created superior traversal, where the different dimensions were better represented a defined. The difficult part about tweaking the hyperparameters is that, as mentioned by *Locatello et al.* [LBL+19], there is no specific combination that will give us the best results.

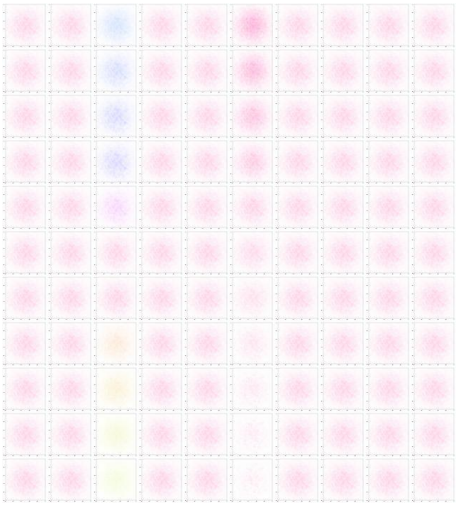
Nevertheless, we have a very big problem with our model. This issue is still related to our data and still concerns generalization. Our model cannot encode and provide a good reconstruction for all types of samples. In this dataset all samples have the same distribution, but what if we have another type of distribution, for example, a linear distribution. This leads us to consider other types of data. For now, we will end this section where further explanations will be provided in the next one.



(a) β-VAE



(b) Factor VAE



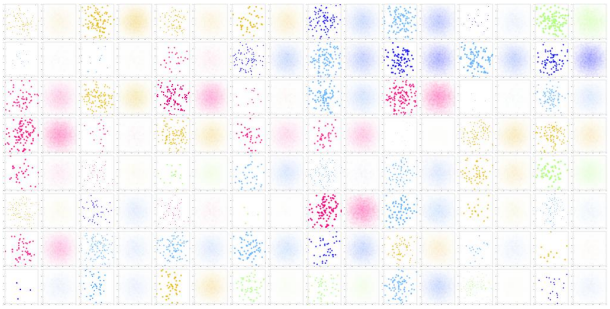
(c) βTCVAE

Figure 2.6: Traversals from the three different models with 100 000 training steps on synthetic dataset.

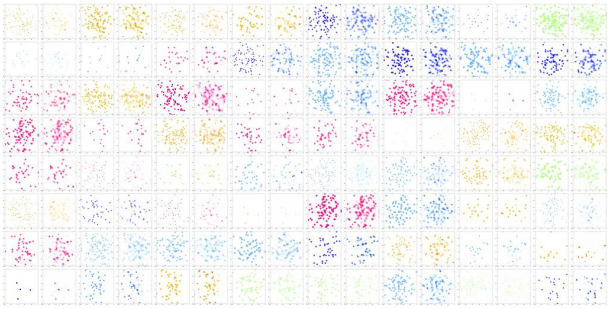
2.4.5 Real world Dataset

Based on our previous results, we will not be using the β-VAE model since it was outperformed by both Factor VAE and βTCVAE models. For this test, we will only be looking at both Factor VAE and βTCVAE models.

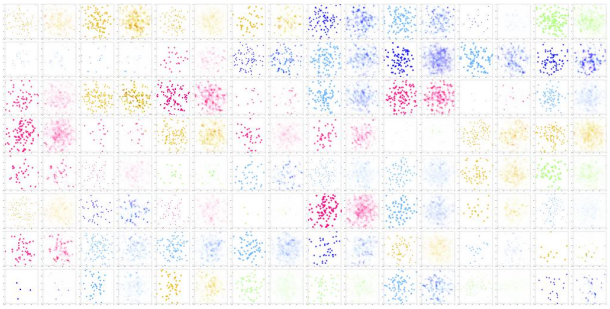
This dataset was created from the R dataset package [R C18]. We chose this dataset because of its diverse samples. One of the main reasons we think the previous dataset failed to produce a wide variety of reconstructions is because it has been trained on a uniformly distributed dataset. These datasets host a wide variety of samples that we think can help with the reconstruction phase. The position factor in this dataset has changed to 314 making the total number of unique images 56520 of size $64 \times 64 \times 3$. In these tests we have trained all our models on 300 000 training steps with the latent vector of ten dimensions.



(a) β-VAE



(b) Factor VAE



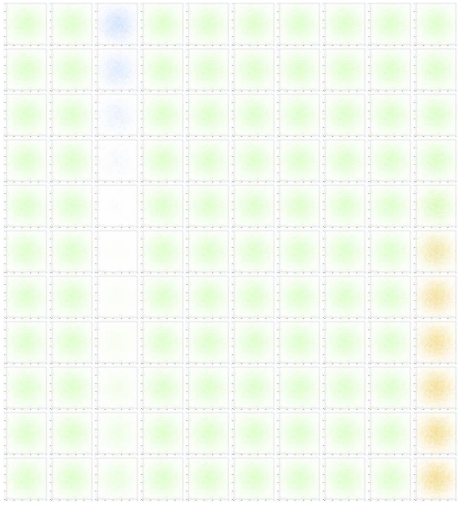
(c) βTCVAE

Figure 2.7: Reconstructions from the three different models with 300 000 training steps on synthetic dataset.

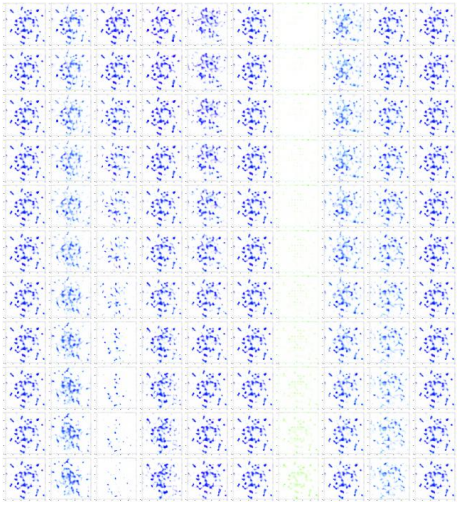
	Reconstruction loss	**BetaVAE score**	**FactorVAE score**	**MIG**
Factor VAE	668	0.79	0.631	0.192
βTCVAE	666	0.767	0.617	0.185

Table 2.8: Results of the real world dataset using the Factor VAE and βTCVAE models.

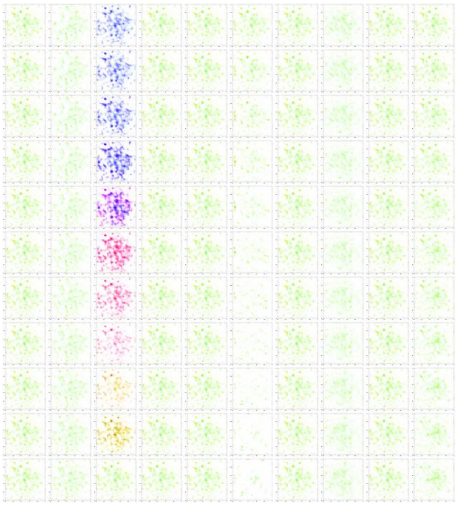
We start by testing our Factor VAE model with the γ hyperparameter set to 40. The reconstruction loss was 688.799 with 18.11 as our KL loss. The BetaVAE training score was 0.79 and 0.779 as the evaluation score. The FactorVAE score had a training accuracy of 0.645 and 0.631 as test accuracy. The last metric we considered is the MIG where we got a value of 0.192.



(a) β-VAE



(b) Factor VAE



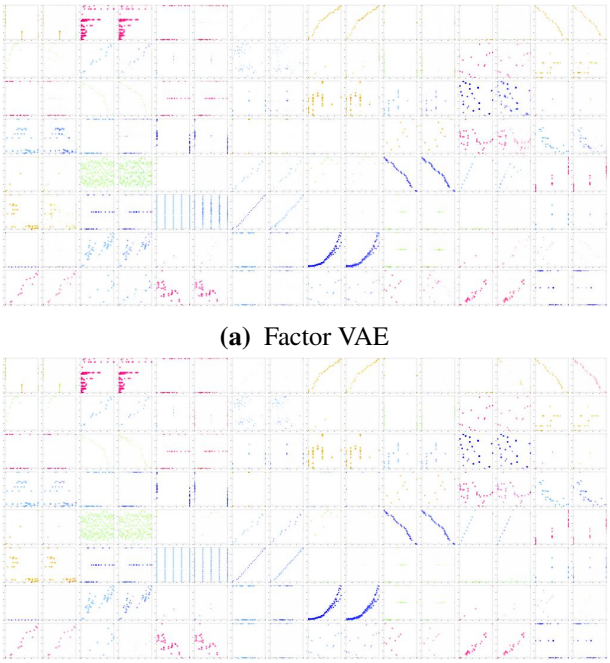
(c) βTCVAE

Figure 2.8: Traversals from the three different models with 300 000 training steps on synthetic dataset.

The next test is done using the βTCVAE model. We again start by checking our reconstruction loss which sits at 666.9 and the KL loss at 11.43. The BetaVAE score has a training accuracy of 0.772 and 0.767 as evaluation accuracy. The FactorVAE score was at 0.617 for the training accuracy and 0.61 for the evaluation accuracy. Last we got 0.185 for the discrete MIG score.

Latent Traversals

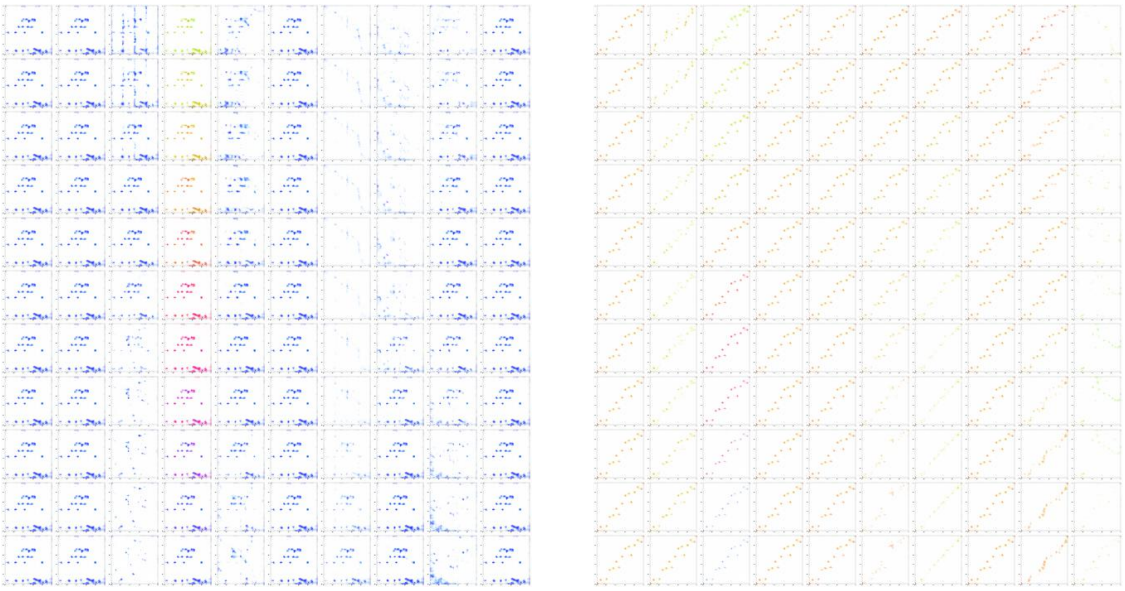
We will now look at the latent traversals and reconstructions for both our Factor VAE and βTCVAE model. Each dimension is traversed one at a time from $[-2.5, 2.5]$ of step size 0.5. Before checking the traversals, let's first check the quality of our reconstructions.



(a) Factor VAE

(b) βTCVAE

Figure 2.9: Traversals from Factor VAE and βTCVAE models on real world dataset.



(a) Factor VAE

(b) βTCVAE

Figure 2.10: Traversals from Factor VAE and βTCVAE models on real world dataset.

We start by checking the reconstructions given by the Factor VAE model. What we can identify from Figure 2.9 that the model was able to portray a reasonable reconstruction. We still get the blurriness of a reconstruction but still, the Factor VAE provides a sharper reconstruction than its β-VAE counterpart.

After checking the reconstructions, we now check the latent traversals. Figure 2.10 shows the traversals of the ten dimension. from the traversals, we can clearly see the color factor represented by the fourth dimension while the size factor is represented by the seventh dimension. Just as before the shape factor lacks representation.

Next, we will check the reconstruction given by the implementation of the βTCVAE model. As predicted, The model did provide a good representation of the original input images, maybe even surpassing the quality given by the Factor VAE model.

We traverse the latent dimension z for the βTCVAE model. From figure 2.10b we can see the traversal of our color factor represented by the third dimension while the size factor is being represented by the seventh dimension. Judging solely by the traversals we can conclude that the Factor VAE outperformed the βTCVAE model.

Discussion

We have seen our model's reconstructions and traversals. Additionally, we have computed the different metrics for both models. Now let us combine both the results presented by our traversals and reconstruction coupled with the results from the disentanglement metrics. Going for the quality of reconstruction, we previously said that the βTCVAE gave us a better reconstruction than that of Factor VAE and that is further verified by their reconstruction loss where the βTCVAE showed a reconstruction loss lower than that of Factor VAE. To check the quality of the disentangled representation we check for the BetaVAE score, the FactorVAE score, and the MIG. For all considered metrics the Factor VAE had a higher score than the βTCVAE therefore the Factor VAE did present us with a slightly better disentangled representation.

2.5 Comparison

Tables 2.9, 2.10, and 2.11 show the different metrics and losses for the different models. The two datasets dSprites and initial dataset were the first datasets we tested our disentangled library with. Although not too much to say about these datasets, they did pave the way to set up our library and familiarize ourselves with its different models. It is really difficult to compare the 4 datasets and almost seems unfair since the initial dataset consists of only 180 and the dSprites dataset has a different number of channels. Therefore we will only be comparing the synthetic dataset against the real world dataset. Both of these datasets have the same number of factors and a relatively equivalent number of samples. The Factor VAE represented in table 2.10 shows that the real world dataset outperformed the synthetic with a lower reconstruction loss by a huge margin and scored higher on all disentanglement scores except the MIG. Table 2.11 shows βTCVAE model which shows consistent results from table 2.10. What we can conclude is that the real world dataset in a general sense seemed to generate better results.

	Reconstruction loss	BetaVAE score	FactorVAE score	MIG
Synthetic dataset	1341	0.579	0.590	0.01
Real world dataset	–	–	–	
dSprites	72	0.98	0.49	0.019
Initial dataset	469	0.758	–	0.044

Table 2.9: Results of all datasets from β-VAE model.

	Reconstruction loss	BetaVAE score	FactorVAE score	MIG
Synthetic dataset	943	0.77	0.691	0.2
Real world dataset	668	0.79	0.631	0.192
dSprites	–	–	–	–
Initial dataset	–	–	–	–

Table 2.10: Results of all datasets from Factor VAE model.

2.6 Conclusion

In this chapter, we have described and implemented three different models. The β-VAE model, the Factor VAE model, and the βTCVAE model. We have tested these three models on four different datasets of which three we have created. Although all models were able to achieve a good disentangled representation the β-VAE failed to deliver a good reconstruction and that's when the two other models come into play.

The main problem that we encountered was the lack of generalization that we addressed with the synthetic dataset and the real world dataset. But even with the real world dataset the position factor had only 314 values, which means there are only 314 different samples. Given the nature of how disentanglement works, we realized that we need a very large dataset in order to generalize. And if we add another factor or even a variation in a factor, the number of samples increases by a huge margin.

Our goal was to see if using disentanglement we can encode the visual properties of a scatterplot. By the end of this experiment, we can clearly say that yes it is possible to encoded and control these properties. Although we were not able to control the size property but we did get a close enough reconstruction to give a good idea about the distribution of our data.

This will conclude experiment 1. In the next experiment, we will be building upon *Jo & Seo* [JS19] work and extract the structural features of a scatterplot. We will create a tool where the user can change these structures and generate his own data.

	Reconstruction loss	BetaVAE score	FactorVAE score	MIG
Synthetic dataset	987	0.739	0.528	0.176
Real world dataset	666	0.767	0.617	0.185
dSprites	–	–	–	–
Initial dataset	–	–	–	–

Table 2.11: Results of all datasets from βTCVAE model.

3 Experiment 2

The second part of this paper will be creating a tool where we can generate images based on the structural properties of a scatterplot. We want the user to have the ability to input an image, and from a trained model control the different features of a scatterplot graph and generate similar data. This experiment will cover the entire process. We will discuss the dataset used, the training process, the results, and the problems and issues faced.

This experiment will be split into two sections. The first section involves gathering data, training our model, and discussing the results. The second section will be dedicated to the construction of the tool that takes our trained model and allows the user to control the latent vector and generating new data.

3.1 Overview

The main goal of this experiment is to create a tool where we can control the structural features of a scatterplot. The user should be able to change each dimension while changing only one factor at a time. We add on the study conducted by *Jo & Seo* [JS19] where the authors train a β-VAE model trying to encode the scagnostic features in a latent dimension of size 32. These dimensions will capture the underlying structures, and traversing these dimensions will hopefully give control over the high-level semantics. The features encoded can be summed up into 8 structural features; Correlation, density, x position, y position, variance, scatter, skew x, skew y. We first try to replicate the results using the β-VAE model, then we try to improve the reconstruction by training with a βTCVAE model.

3.2 Dataset

The dataset in this study is taken from *Jo & Seo* [JS19]. The data gathered is 2 101 990 datasets from different repositories including Plotly [Inc15] and UCI Machine Learning [DG17]. The datasets are then filtered to satisfy the following criteria.

The datasets must contain only quantitative data with floating values and exceeding 20 unique numbers. These criteria must be fulfilled to ensure that our data does not include any categorical fields with numerical names. A dataset that fails to provide a pair of quantitative fields were discarded. Additionally, if the dataset contains more than 30 quantitative fields, the dataset will be discarded. The reason being is to avoid synthetically generated data resulting in a large number of synthetic scatterplots. Any duplicate datasets were also discarded. The number of remaining datasets was 477 177 datasets. All pairs of quantitative fields from each dataset were then combined. All data were normalized and plotted on a $[0, 1]$ axis. Each data sample occupied 1 pixel of white

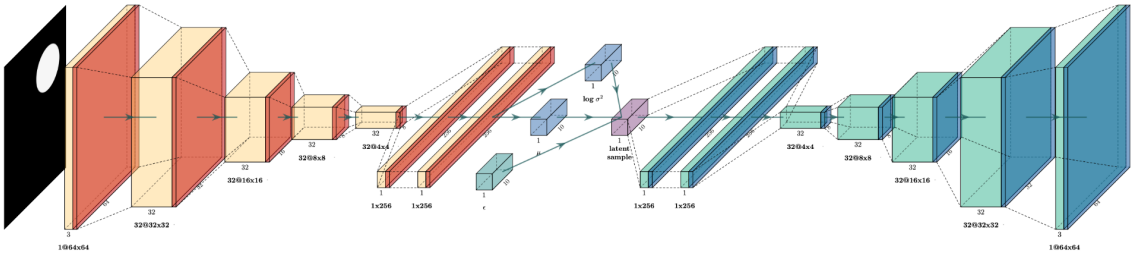


Figure 3.1: Experiment 2 architecture as described by *Burgess et al.* [BHP+18].

color. All data samples were plotted on a black background with no axes and legends to reduce clutter. The resulting image has a size of $64 \times 64 \times 1$. This dataset holds 1 189 039 unique scatterplot images.

3.3 Architecture

For this experiment we used a different library than Experiment 1. The Disentangled library by *Locatello et al.* [LBL+19] did not allow the disentanglement of continuous factors, therefore we decided to change the library and work with the Disentangled VAE library [20118]. In this experiment we used both β-VAE model and the βTCVAE models.

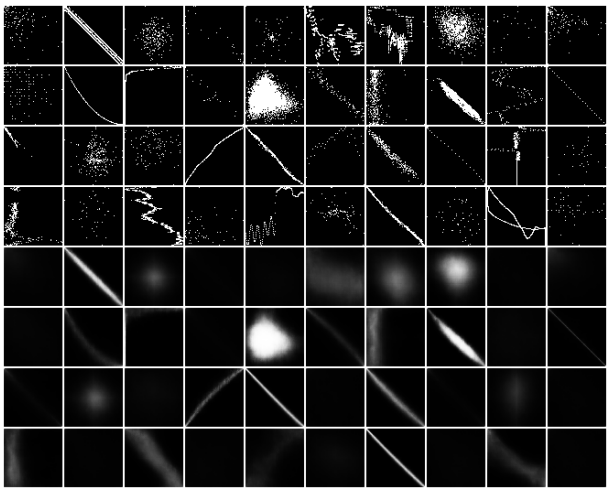
We used the same architecture for both models for a fair comparison. The architecture used is the same implemented in 'Understanding disentangling in β-VAE' [BHP+18]. Figure 3.1 shows The architecture of both encoder and decoder. The encoder includes 4 convolutional layers with 32 channels each, the kernel has a size of 4×4 and a stride of 2. This was followed by two fully connected layers of 256 neurons. The size of the latent dimension for this test was chosen to be 32. The decoder architecture is equivalent to the transpose of the encoder architecture. For both tests, we used an Adam Optimizer with a learning rate of 0.00005 and a Bernoulli loss function for the reconstruction.

3.4 Model Training

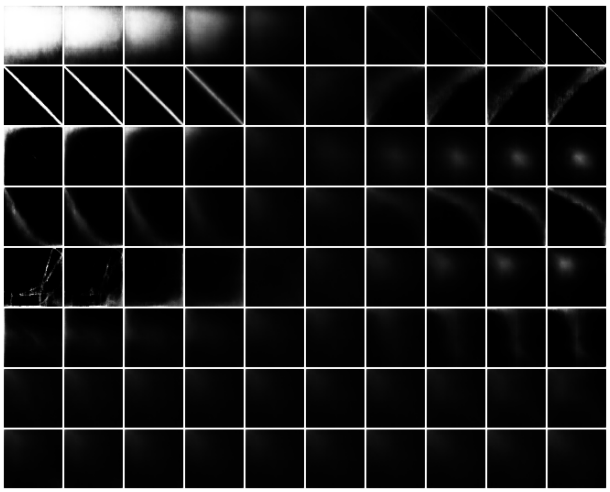
3.4.1 β-VAE Model

We first start our test using the standard β-VAE model. We set the β hyperparameter to 16 and the batch size to 124. This architecture basically projects the input data to latent vector and tries to learn the most important factors. Trying to mimic the behavior of the model in *Jo & Seo* [JS19] we chose our latent vector to be 32. Increasing the size of our latent dimension will allow our model to learn more and minimize the data loss.

The training was implemented over 40 epochs. The β-VAE model is known for its failure to produce a good reconstruction as β increases. For this model, we had a reconstruction loss of 0.365. The next step is to traverse our latent vector.



(a) Top 4 rows representing the original samples; Bottom 4 rows representing the reconstructions.

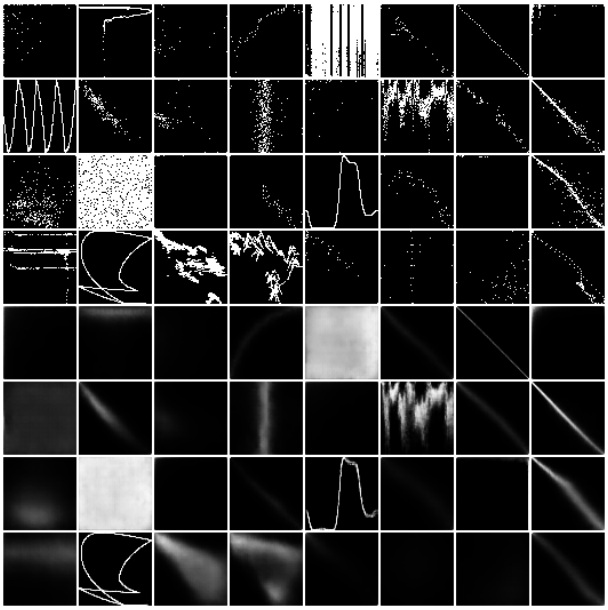


(b) Columns are the traversals; Rows are the factors in the latent dimension.

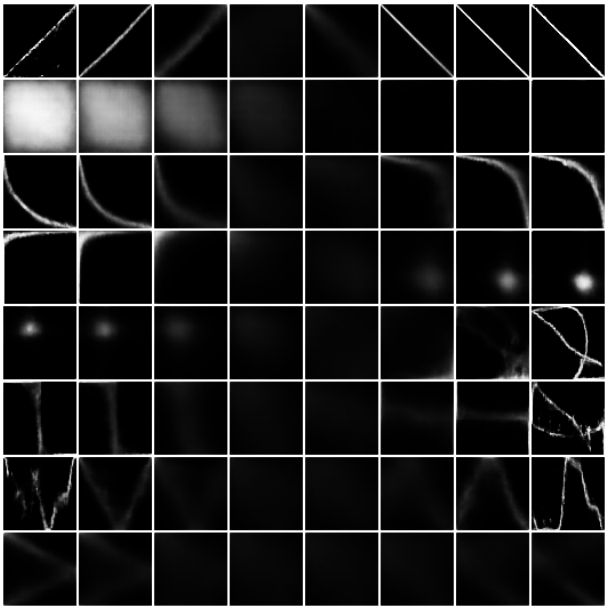
Figure 3.2: Reconstructions and traversals for β-VAE model.

As we can see from the traversals from Figure 3.2b we can formulate the following factors that have been disentangled by our model: correlation, Density, x position, y position, variance, scatter, skew x, skew y. we have chosen the eight dimension with the highest KL divergence. Figure 3.2a shows the reconstructions performed by our model. For this experiment, it is not necessary to get a perfect reconstruction. Again our goal is to build a tool that can provide the user with the flexibility to control the latent dimension and generate similar data. These reconstructions will later serve as intensity maps to create similar data.

We have now achieved results similar to *Jo & Seo* [JS19]. Nonetheless, we want to find out if we can achieve a better reconstruction so we considered improving our model using a βTCVA model.



(a) Top 4 rows representing the original samples; Bottom 4 rows representing the reconstructions.



(b) Columns are the traversals; Rows are the factors in the latent dimension.

Figure 3.3: Reconstructions and traversals for βTCVAE model.

3.4.2 βTCVAE Model

In this test, we opted to use the βTCVAE for the exact purpose stated in experiment 1. The β-VAE model results in a blurry reconstruction due to the nature of its objective function penalizing the mutual information $I(x;z)$, which is not desirable for a good reconstruction. We will now test our model with βTCVAE which in experiment 1 improved the reconstruction quality while maintaining its disentangled function.

The βTCVAE model requires a β_{TCVAE} hyperparameter which we will set to 8. Again we used an Adam Optimizer [KB17] with a learning rate of 0.00005. There is no definitive method to determine the best hyperparameter value. Therefore we use the same hyperparameters used in experiment 1.

We train our model over 40 epochs. The βTCVAE resulted in a reconstruction loss of 0.318, the minimum information loss was 0.185, and the total correlation loss was -0.176. The results show a slight improvement than the β-VAE model.

In Figure 3.3a we see the reconstructions our model was able to achieve. The quality of reconstruction is slightly better than the quality of the β-VAE model. And similarly as before, Figure 3.3b shows the traversals from a range of $[-2.5, 2.5]$ of the eight dimensions with the highest KL divergence. The eight encoded factors are the same as the factors of the β-VAE model and as seen in figure 3.3b was able to disentangle these factors. As stated before we are satisfied with the performance of our model and an ideal reconstruction is not necessary.

3.5 Tool

In this section, we will take the trained model from the previous section and build a tool that allows the user to control the 8 latent dimensions and generate more data based on the desired reconstruction. We will also discuss the use cases of such a tool, and the resulting images it was able to generate.

3.5.1 Use Cases

The tool allows the user to input an image and using our trained model outputs the appropriate reconstruction. The tool has eight sliders each controlling one disentangled factor, and the user can see the changes in run-time. When the user is satisfied with the reconstruction the user has the ability to generate random samples based on the reconstruction, where the black and white image acts as an intensity map.

This tool can be very useful for generating similar data. This tool allows full control over the following factors: Correlation, density, x position, y position, variance, scatter, skew x, skew y. The user has the ability to make us of our disentangled model to generate different plots of different shapes with the added flexibility given by our disentanglement factors.

This tool can also be viewed as a visualization tool where the user can see the result of the traversal of each factor. Whether for educational or personal purposes this tool can be helpful in visualizing the encoded features and studying their behaviors through their traversals.

3.5.2 Pipeline

In order to build our tool, we had to combine the different aspects of image processing, visualization, and machine learning. The tool was built using python where we used the efforts of multiple libraries to acquire a proper functionality. This section covers the different libraries and the steps required to build our tool.

First of all our model is trained on an input of $64 \times 64 \times 1$. Any image provided by the user must be treated to become compatible with our model. Therefore, the first step was to take an input image of any size and convert it to a 64×64 black and white image. Since our model is trained on images with a black background and white sample points we converted any input image to a black and white image of one channel where we converted the background to a black color and the sample points to a white color. To transform the images to the necessary size and color we used the Pillow library [Cla15].

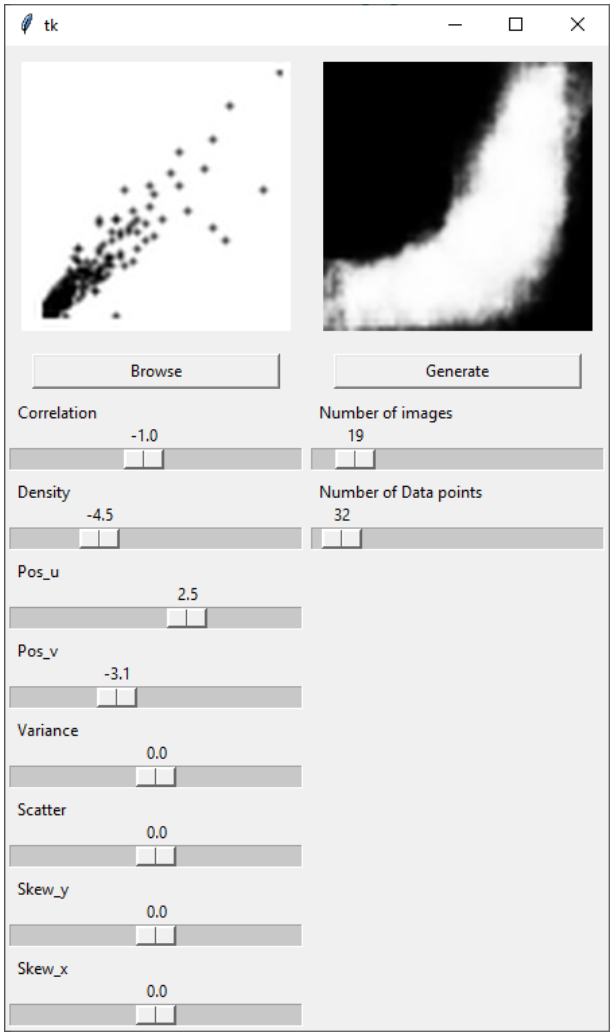
After the necessary adjustments, we can now use our trained model to provide a reconstructed image of our input. Our model's encoder will take the input image and output a latent vector of size 32. As seen in the previous section we select the factors that have the highest dimension-wise KL divergence. Each of these factors can then be traversed by an x value that the user can control. Our model can then take the traversed latent vector and output its respective image. The output image will be saved and displayed to the user.

Now that we have our reconstruction we can use the output image as an intensity map to create other plots and images. Each pixel in the reconstructed images has a value between $[0, 1]$. This gives us a distribution where darker areas or lower values will have a lower probability for a sample to exist than a lighter area. Using simple python code we were able to create such a behavior. The number of sample points can be controlled by the user as well as the number of images to generate. The plots are created using the Matplotlib [Hun07] library.

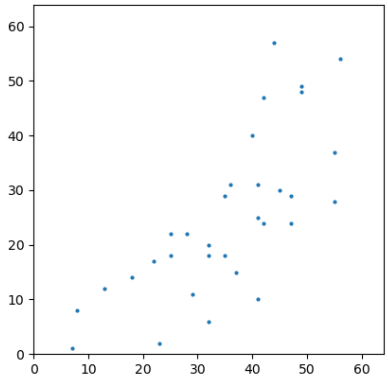
Finally, we need a Graphical User Interface (GUI) to bind all these components together and allow the user to interact with the trained model. The library used is a python library, Tkinter [Van20]. Using this library we were able to show the input image as well as the reconstructed image. The traversals are updated at run-time to increase our tool's practicality. Not only does the GUI allows control over the latent factors but also the different parameters for the plot configuration.

3.5.3 Results

As can be seen from Figure 3.4a we can see the result of the different components. First, the browse button allows the user to input an image file from the local machine. When the user selects an image the proper transformations will be applied directly after. The Browse button will also trigger our trained model to take our transformed image and provide us with its reconstruction. At this point, we have not yet tempered with any of the latent factors. We have provided eight sliders each of which controls a different factor. Each of the sliders is labeled for the user's convenience. Changing the value of a slider will trigger our trained model to update the reconstructed image based on the values provided by the user using our the sliders. When the user is satisfied with the results of the reconstruction he can then use the slider on the right to control the number of samples



(a) On the left side starting from the top we have the input image then browse button to input image then the sliders to control the latent vector; On the left side starting from the top we have the output image then image generator button then slider to control the image parameters.



(b) Image generated

Figure 3.4: Scatterplot tool

and number of images to create. The Generate button will then create the plots and images based on the parameters chosen by the user. The generated images are then saved locally on the user's machine.

4 Future Work and Conclusion

4.1 Future Work

There are a lot of improvements that can be done for experiment 1. We recall the main issue being the lack of generalization. Using our technique one must create a huge dataset and annotate each image with the factors it contains. One solution could be using *Locatello et al.* [LTB+20] model where we use a semi-supervised model that needs only a few labels. The idea is that in a practical sense one does not always have access to the labels, and manually generating these labels is very costly. In our case, we use web crawling to gather data and images, then using the semi-supervised techniques we can maybe achieve better results.

Another idea might be using transfer learning to solve this issue. We could train a model to learn the different structures and distribution. Then we can combine the results with our disentanglement model. But this poses a big question of how can we combine both models. How can we use the results of experiment 2 as a structure for experiment 1.

In experiment 2 we learned 8 different factors that defined our structure. However, the initial aim was to disentangle the scagnostic features. It's a very difficult task since the factors are of a continuous nature rather than discrete.

Another idea is to disentangle both continuous and discrete factors. Is it possible to combine both experiments where we can disentangle the visual and structural features. *Dupont et al.* [Dup18] presents a new disentanglement framework, the Joint VAE, that shows both continuous and categorical variations can be disentangled.

4.2 Conclusion

This will conclude our research of disentanglement for scatterplot images. For this experiment we used *Locatello et al.* [LBL+19] disentanglement library. In this paper, we were able to disentangle the visual properties of a scatterplot, size, shape, and color. We refined the blurry reconstructions given by the β-VAE model. We implemented two different models, the Factor VAE and βTCVAE models, which both showed improvements in disentanglement and reconstruction. We discover that even with our real world dataset we still fall short with the lack of generalization. However, both the encoding and disentanglement proved to be successful.

The work *Jo & Seo* [JS19] gave us the idea to extend the research and create a tool where one can use this model to alter the encoded factors. In experiment 2 we try first to replicate the work done by *Jo & Seo* [JS19] and even try to improve the reconstruction quality by using the βTCVAE model. Using python Tkinter library, we created a tool where the user can input an image and traverse its

latent vector. The tool proved to have a successful functionality generating both reconstructions based on the varied parameters and then generating scatterplot images based on the number of images and number of samples parameters.

In this paper, we set out to understand the structural and visual properties of a scatterplot. Additionally, we gained control over these properties and got a better understanding of the underlying structure that defines scatterplot similarities.

Bibliography

[20118] 2019 Yann Dubois, Aleco Kastanos, Dave Lines, Bart Melman. *disentangling-vae*. 2018. URL: https://github.com/YannDubs/disentangling-vae/blob/master/LICENSE (cit. on p. 38).

[BDM+18] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, M. Stonebraker. "Beagle: Automated Extraction and Interpretation of Visualizations from the Web". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–8. ISBN: 9781450356206. URL: https://doi.org/10.1145/3173574.3174168 (cit. on p. 14).

[BHP+18] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, A. Lerchner. *Understanding disentangling in β-VAE*. 2018. arXiv: 1804.03599 [stat.ML] (cit. on p. 38).

[CDH+16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *CoRR* abs/1606.03657 (2016). arXiv: 1606.03657. URL: http://arxiv.org/abs/1606.03657 (cit. on p. 15).

[CKNH20] T. Chen, S. Kornblith, M. Norouzi, G. Hinton. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG] (cit. on p. 13).

[Cla15] A. Clark. *Pillow (PIL Fork) Documentation*. 2015. URL: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf (cit. on p. 42).

[CLGD19] R. T. Q. Chen, X. Li, R. Grosse, D. Duvenaud. *Isolating Sources of Disentanglement in Variational Autoencoders*. 2019. arXiv: 1802.04942 [cs.LG] (cit. on pp. 17, 20).

[CM84] W. S. Cleveland, R. McGill. "Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods". In: *Journal of the American Statistical Association* 79.387 (1984), pp. 531–554. ISSN: 01621459. URL: http://www.jstor.org/stable/2288400 (cit. on p. 13).

[DG17] D. Dua, C. Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml (cit. on p. 37).

[Dup18] E. Dupont. *Learning Disentangled Joint Continuous and Discrete Representations*. 2018. arXiv: 1804.00104 [stat.ML] (cit. on p. 45).

[DW14] T. N. Dang, L. Wilkinson. "ScagExplorer: Exploring Scatterplots by Their Scagnostics". In: *2014 IEEE Pacific Visualization Symposium*. 2014, pp. 73–80. DOI: 10.1109/PacificVis.2014.42 (cit. on pp. 11, 13).

[GPM+14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML] (cit. on p. 15).

[HA16] J. Harper, M. Agrawala. “Converting Basic D3 Charts into Reusable Style Templates”. In: *CoRR* abs/1609.05283 (2016). arXiv: 1609.05283. URL: http://arxiv.org/abs/1609.05283 (cit. on p. 14).

[HA20] E. Hoque, M. Agrawala. “Searching the Visual Style and Structure of D3 Visualizations”. In: *IEEE Transactions on Visualization and Computer Graphics* 26 (2020), pp. 1236–1245 (cit. on p. 14).

[HJ] M. D. Hoffman, M. J. Johnson. “Elbo surgery: yet another way to carve up the variational evidence lower bound”. In: (cit. on p. 18).

[HMP+16] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner. “beta-vae: Learning basic visual concepts with a constrained variational framework”. In: (2016) (cit. on pp. 15, 17, 20).

[HS06] G. E. Hinton, R. R. Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (2006), pp. 504–507. ISSN: 0036-8075. DOI: 10.1126/science.1127647. eprint: https://science.sciencemag.org/content/313/5786/504.full.pdf. URL: https://science.sciencemag.org/content/313/5786/504 (cit. on p. 14).

[Hun07] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55 (cit. on pp. 20, 42).

[Inc15] P. T. Inc. *Collaborative data science*. 2015. URL: https://plot.ly (cit. on p. 37).

[JKS+17] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, J. Seo. “ChartSense: Interactive Data Extraction from Chart Images”. In: ACM, May 2017. URL: https://www.microsoft.com/en-us/research/publication/chartsense-interactive-data-extraction-chart-images/ (cit. on p. 14).

[Joy11] J. M. Joyce. “Kullback-Leibler Divergence”. In: *International Encyclopedia of Statistical Science*. Ed. by M. Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327. URL: https://doi.org/10.1007/978-3-642-04898-2_327 (cit. on pp. 17, 18).

[JS19] J. Jo, J. Seo. “Disentangled Representation of Data Distributions in Scatterplots”. In: *2019 IEEE Visualization Conference (VIS)*. 2019, pp. 136–140. DOI: 10.1109/VISUAL.2019.8933670 (cit. on pp. 11, 12, 35, 37–39, 45).

[KB17] D. P. Kingma, J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG] (cit. on p. 41).

[KM19] H. Kim, A. Mnih. *Disentangling by Factorising*. 2019. arXiv: 1802.05983 [stat.ML] (cit. on pp. 17, 18, 20).

[Kos89] S. M. Kosslyn. “Understanding charts and graphs”. In: *Applied Cognitive Psychology* 3.3 (1989), pp. 185–225. DOI: https://doi.org/10.1002/acp.2350030302. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/acp.2350030302. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/acp.2350030302 (cit. on p. 13).

[KW14] D. P. Kingma, M. Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML] (cit. on pp. 14, 15, 17).

[LBL+19] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, O. Bachem. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *International Conference on Machine Learning*. 2019, pp. 4114–4124 (cit. on pp. 20–22, 29, 38, 45).

[LTB+20] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, O. Bachem. "Disentangling Factors of Variations Using Few Labels". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SygagpEKwB (cit. on p. 45).

[Mac86] J. Mackinlay. "Automating the Design of Graphical Presentations of Relational Information". In: *ACM Trans. Graph.* 5.2 (Apr. 1986), pp. 110–141. ISSN: 0730-0301. DOI: 10.1145/22949.22950. URL: https://doi.org/10.1145/22949.22950 (cit. on p. 14).

[MHHL17] L. Matthey, I. Higgins, D. Hassabis, A. Lerchner. *dSprites: Disentanglement testing Sprites dataset*. https://github.com/deepmind/dsprites-dataset/. 2017 (cit. on p. 19).

[MTW+20] Y. Ma, A. K. H. Tung, W. Wang, X. Gao, Z. Pan, W. Chen. "ScatterNet: A Deep Subjective Similarity Model for Visual Analysis of Scatterplots". In: *IEEE Transactions on Visualization and Computer Graphics* 26.3 (2020), pp. 1562–1576. DOI: 10.1109/TVCG.2018.2875702 (cit. on p. 11).

[NH97] L. T. Nowell, D. Hix. "Graphical encoding for information visualization: using icon color, shape, and size to convey nominal and quantitative data". In: 1997 (cit. on pp. 13, 17).

[PH17] J. Poco, J. Heer. "Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images". In: *Computer Graphics Forum (Proc. EuroVis)* (2017). URL: http://idl.cs.washington.edu/papers/reverse-engineering-vis (cit. on p. 14).

[R C18] R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2018. URL: https://www.R-project.org/ (cit. on pp. 20, 30).

[SKC+11] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, J. Heer. "ReVision: Automated Classification, Analysis and Redesign of Chart Images". In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. Santa Barbara, California, USA: Association for Computing Machinery, 2011, pp. 393–402. ISBN: 9781450307161. DOI: 10.1145/2047196.2047247. URL: https://doi.org/10.1145/2047196.2047247 (cit. on pp. 11, 13).

[Van20] G. Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020 (cit. on p. 42).

[WAG05] L. Wilkinson, A. Anand, R. Grossman. "Graph-theoretic scagnostics". In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* 2005, pp. 157–164. DOI: 10.1109/INFVIS.2005.1532142 (cit. on p. 13).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Druck-Exemplaren überein.

Datum und Unterschrift:

Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted hard copies.

Date and Signature: 15.02.2021