Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Anna Rodum Bjøru

The importance of disentanglement when learning representations

Master's thesis in Computer Science
Supervisor: Helge Langseth
September 2021

NTNU
Norwegian University of
Science and Technology

Anna Rodum Bjøru

The importance of disentanglement when learning representations

Master's thesis in Computer Science
Supervisor: Helge Langseth
September 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

The field of disentangled representations has been suggested a promising avenue in search of robust and generalisable machine learning algorithms and increased data efficiency. It has in recent years received a lot of interest.

The key assumption leveraged by disentangling methods is that raw data is generated by a set of factors of variation, referred to as generative or explanatory factors. Such factors are considered to correspond to inherent properties of the data, each factor encoding a unit of information present in the data that is both semantically meaningful and statistically independent of all other factors. Disentangling techniques should attempt to capture and disentangle as many factors as possible describing the data distribution in a compact, low dimensional space.

This thesis provides a thorough review of disentangled representation learning and its theoretical foundation, looking at various methods presented in the field that aim to learn disentangled representations, covering both unsupervised and supervised approaches. Also reviewed are evaluation techniques introduced as to reliably determine the level of disentanglement achieved by the methods, briefly discussing some qualitative approaches and then looking in-depth at the quantitative disentanglement metrics most prevalent in the literature.

It is commonly perceived that disentangled representations would provide an advantage in learning models to solve downstream tasks. It is however not yet clear to what extent enforcing disentanglement results in representations that exhibit such an advantage. This thesis presents results that support the hypothesis that increased disentanglement results in improved downstream model accuracy. Both overall performance as well as limited sample performance of simple downstream tasks are shown to correlate well with disentanglement as measured by both unsupervised and supervised disentanglement metrics.

Sammendrag

I jakten på generaliserbare, robuste maskinlæringsalgoritmer, samt økt dataeffektivitet, har fagfeltet "disentangled representation learning" i de senere år utviklet seg i en lovende retning, og feltet har vært gjenstand for stor interesse.

Nøkkelantagelsen i sentrum for disentanglement går ut på at høydimensjonale rådata genereres fra et sett med generative faktorer, det vil si faktorer som hver for seg forklarer iboende egenskaper ved rådataene, under en antagelse om at hver faktor isolert er semantisk meningsfull og statistisk uavhengig av alle andre faktorer. For et gitt datasett er målet med disentanglement å lære lavdimensjonale datarepresentasjoner bestående av komponenter som tilsvarer de generative faktorene.

Denne masteroppgaven inneholder en grundig studie av disentanglement i representasjonslæring, og de teoretiske ideene som ligger til grunn, med fokus på forskjellige metoder introdusert innenfor feltet som forsøker å lære modeller som lykkes i disentanglement. Her er metoder som benytter både veiledet og ikke-veiledet læring inkludert. I tillegg presenteres en studie av kvalitative og kvantitative evalueringsteknikker som forsøker å fastsette graden av disentanglement oppnådd gitt en modell.

Selv om det innenfor feltet er en felles oppfatning om at data-representasjoner som oppfyller disentanglement vil være godt egnet til å løse nedstrøms læringsoppgaver, er det fortsatt uklart i hvilken grad en slik fordel kan knyttes til disentanglement. I denne oppgaven presenteres resultater som støtter hypotesen om at disentanglement fører til bedre prestasjon hos enkle modeller som løser nedstrøms læringsoppgaver, både overordnet og i tilfeller der lite data er tilgjengelig. Grad av nøyaktighet i løsning av læringsoppgaver vises å korrelere godt med disentanglement målt både av teknikker som krever kjennskap til de generative faktorene og deres verdier, samt av teknikker som ikke gjør antakelser om de underliggende generative faktorene.

Preface

This master thesis was performed at the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU). The thesis is situated in the field of machine learning, and is a continuation of a project report I wrote in 2020.

I would like to thank my supervisor Professor Helge Langseth for inspiring discussions and invaluable guidance through my work.

Anna Rodum Bjøru Trondheim, September 17, 2021

Contents

List of Figures

List of Tables

Notation

x	-	scalar
$\boldsymbol{x}$	-	vector
X	-	matrix
$\boldsymbol{X}$	-	tensor
x_i	-	element of vector $\boldsymbol{x}$ located in row i
$x_{i,j}$	-	element of matrix X located in row i, column j
$x_{i,j,k}$	-	element of 3D tensor $\boldsymbol{X}$ located in row i, column j, and at depth k
$\boldsymbol{x}_{\setminus i}$	-	all elements of $\boldsymbol{x}$ except for element in row i

x	-	scalar random variable
$\mathbf{x}$	-	vector random variable
$p(\mathrm{x})$	-	probability distribution over random variable x
$\mathrm{x} \sim p$	-	random variable x with distribution p
$\mathbb{E}_{\mathrm{x}\sim p}[q(\mathrm{x})]$	-	The expected value of $q(\mathrm{x})$ with x having distribution p
$D_{\mathrm{KL}}(q\|p)$	-	Kullback-Leibler divergence of probability distributions q and p
$\hat{p}_{\mathbb{X}}$	-	The empirical distribution given by a dataset
$p^*_{\mathbb{X}}$	-	The true generating distribution of the data in the dataset
$p_{\boldsymbol{\theta}}$	-	The distribution function given by a model with parameters $\boldsymbol{\theta}$
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	-	Gaussian distribution over random variable $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

$\mathbb{X}$	-	The dataset with examples
$\boldsymbol{x}^{(i)}$	-	The i'th example of a dataset $\mathbb{X}$
$\boldsymbol{y}^{(i)}$	-	The target attached to the i'th example of a supervised dataset $\mathbb{X}$
$\hat{\boldsymbol{y}}$	-	Prediction made by a function f from input $\boldsymbol{x}$, $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$

Chapter 1

Introduction

1.1 Motivation

Learning useful data representations is considered one of the main contributions to the success of machine learning over the last decades, and the potential of machine learning algorithms is to a large extent determined by how the data they receive as input is represented [Bengio et al., 2014]. In general terms, representation learning is the process of learning what features best describe the data, and a good data representation is one that proves useful as input to a learning algorithm solving some learning task. If the task in question is solved satisfactorily on the given input, the representation learning algorithm has succeeded in extracting useful information from the data and making it accessible for the algorithm solving the task.

Traditional techniques for creating good data representations typically require human involvement, where features are explicitly designed leveraging domain knowledge about the data at hand. Such techniques are referred to as feature engineering, and can quickly get time consuming and even infeasible in cases concerning large sets of complex data. Consequently, representation learning algorithms that receive raw, often high dimensional data as input and produce good data representations for any subsequent tasks to be solved is desirable in the search for artificial intelligence, removing the need for human involvement through feature engineering and arriving at increasingly flexible and efficient solutions, possibly applicable to a diverse set of problems

Particularly interesting perhaps is unsupervised representation learning, where one attempts to learn representations for data through unsupervised learning using unlabeled data. Labeled data is then preprocessed by applying the trained representation model before a supervised learning model is trained on the resulting representations. As unlabeled data is typically much easier to collect than labeled data, semi-supervised learning has the potential to take advantage of far more extensive information found in possibly vast unlabeled datasets. Such techniques can help in preventing overfitting, resulting in models that generalize better to unseen data and in turn increase accuracy of predictions. It is generally considered that unsupervised representation learning has potential to drive machine learning toward significant progress. This belief is also supported by a hypothesis suggesting that learning conducted by the human brain is largely unsupervised, in that hu-

mans learn the structure of the world mainly from observation [LeCun et al., 2015], requiring only a few labeled examples.

Deep learning provides examples of highly successful representation learning algorithms, often taking images, video, sound or text as input and creating representations through multiple layers in deep neural networks, separating out features and recombining them in new ways in order to improve performance in the last layer of the network, where a learning task is solved [Goodfellow et al., 2016]. This way, deep networks implicitly leverage an assumption that useful features are gradually built on top of each other, allowing for hierarchies of increasingly abstract features approaching the last layer.

Taking advantage of the assumption that new, more useful features can be build on top of less abstract features starting from raw data input has led deep neural networks to be able to solve a range of complex tasks. Similarly, other general assumptions about the world can be expressed in representations, providing general representations that can be used in a wide range of problem solving tasks [Bengio et al., 2014]. Such assumptions are also referred to as priors.

Generative factors of variation, also referred to as explanatory factors, is an interesting example of such priors. This is an assumption that raw data, or more precisely the world state described by the raw data, is generated by a set of underlying mutually independent factors, where recovering all or most of these factors in a disentangled structure would provide an efficient, flexible representation [Bengio et al., 2014]. Each factor is expected to change its value independently of the configuration of the remaining factors, and learning a representation is reduced to learning the distribution of each of the generative factors given any input. Such a representation would allow for generalisation to factor combinations not necessarily encountered during training, capturing the full data distribution, from potentially less data.

Learning representations that disentangle the generative factors of data has been suggested to be a robust approach to representation learning [Bengio et al., 2014; Higgins et al., 2018; Locatello et al., 2019], with the goal being to capture and disentangle as many factors as possible describing the data. A disentangled representation should contain all or most of the information present in the original data, encoded in an interpretable structure that axis aligns with the generative factors, which are assumed to correspond to semantically meaningful concepts.

The approaches to recovering disentangled representations presented in the literature are diverse and range from purely unsupervised methods to methods requiring different levels supervision. Overall, representing the data by disentangled representations is suggested beneficial in enabling machine learning to solve problems more efficiently and flexibly, producing general representations that could potentially serve as useful input to many different learning tasks, unknown at the time of representation learning.

1.2 Goals and Research Questions

A review of the goal of this masters project is presented in the following, with two research questions highlighted.

Goal To explore advantages of disentangled representations when solving downstream prediction tasks

Representing high-dimensional raw data by disentangling generative factors is suggested beneficial. As mentioned in Section 1.1, one main motivation behind attempting to learn disentangled representations is the hypothesis that such representations will result in improved performance in downstream learning tasks.

Research question 1 Do existing disentanglement evaluation techniques reliably quantify disentanglement?

Quantifying the impact of disentangled representation learning on the performance of downstream prediction tasks requires being able to reliably measure disentanglement. In order to examine the effect of disentanglement, metrics should preferably be unambiguous and precise in determining level of disentanglement.

Research question 2 Will unsupervised disentanglement improve downstream performance on simple prediction tasks?

Unsupervised models enable general learning of disentanglement, where representations are learned independently of the downstream tasks in question. Given reliable measures of disentanglement, impact of disentanglement on the ability of a subsequent model to solve learning tasks can be examined.

1.3 Thesis Structure

The thesis consists of six chapters, where Chapter 1 provides an introduction.

Chapter 2 contains an overview of background theory useful in understanding the material discussed in the remaining chapters. This includes a brief review of deep learning models in Section 2.2, as well as an introduction to variational inference in Section 2.3.

Chapter 3 provides a literature review of disentangled representation learning. Section 3.2 discusses representation learning in general and Section 3.3 continues the discussion by introducing the disentanglement property and the variational autoencoder. The rest of chapter 3 gives an overview of some approaches attempting to learn disentangled representations, with Section 3.4 focusing on unsupervised methods and Section 3.5 focusing on supervised methods.

Chapter 4 provides a review covering the evaluation methods used to measure how well a representation learning model disentangles. Section 4.1 gives a brief introduction to disentanglement evaluation, Section 4.2 presents some qualitative evaluation techniques commonly encountered in the disentanglement literature, and

Section 4.3 presents quantitative evaluation techniques, covering a set of metrics designed to measure model disentanglement.

Chapter 5 presents the results of experiments designed to test whether increasing disentanglement results in representations that improve prediction accuracy of downstream tasks. The experiment plan and setup is detailed in Sections 5.1 and 5.2, and Section 5.3 describes the experiment results.

Chapter 6 concludes the thesis, with Section 6.1 presenting an evaluation of the results presented in Chapter 5, Section 6.2 detailing the contributions of the master's project and Section 6.3 discussing future work.

Chapter 2

Background Theory

2.1 Introduction

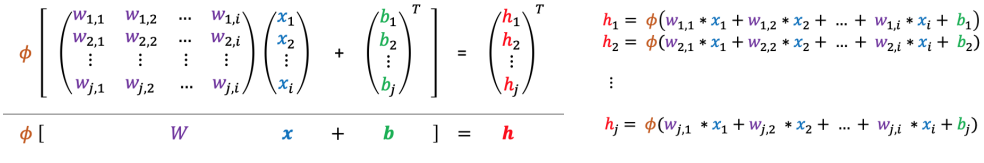
This chapter presents an overview of background theory, with Section 2.2 covering deep learning and Section 2.3 covering variational inference. The material presented in Chapter 2 is based on [Goodfellow et al., 2016]. An earlier version of the chapter was included in [Bjøru, 2020].

2.2 Deep Learning

Machine learning attempts to learn useful patters from a set of data points $\mathbb{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(n)}\}$, such that these patterns can be generalized to new, unseen data. Developing powerful machine learning algorithms is considered to be one of the most promising ways of achieving artificial intelligence. Deep learning is a sub-field of machine learning that covers a set of flexible, powerful algorithms and models designed to solve complex learning problems, both supervised and unsupervised. This section attempts to give a brief review of deep learning and some useful related concepts.

The simplest deep learning models are based on taking affine transformations of the input $\boldsymbol{x}$ and applying a nonlinear mappings ϕ to the result. An affine transformation is made up of a linear transformation $W\boldsymbol{x}$ and a constant term $\boldsymbol{b}$, such that if $\boldsymbol{a} = W\boldsymbol{x} + \boldsymbol{b}$ for vector-valued $\boldsymbol{x}$, a matrix W and a vector $\boldsymbol{b}$, then $\boldsymbol{a}$ is an affine transformation of $\boldsymbol{x}$. For $\boldsymbol{x} \in \mathbb{R}^i$ and $\boldsymbol{a} \in \mathbb{R}^j$, where i and j are integers, $W \in \mathbb{R}^{i \times j}$ and $\boldsymbol{b} \in \mathbb{R}^j$. Applying a nonlinear mapping to $\boldsymbol{a}$ gives a function $f(\boldsymbol{x}) = \phi(W\boldsymbol{x} + \boldsymbol{b})$. The details of the matrix calculations for such vector-valued inputs $\boldsymbol{x}$ are outlined in Figure 2.1. The calculations extend to tensor-valued inputs of 3 or more dimensions.

The functions $f(\boldsymbol{x}) = \phi(W\boldsymbol{x}+\boldsymbol{b})$ are also called layers, or simply transformations of the input $\boldsymbol{x}$. Several such transformations can be stacked by function composition



(a) (b)

Figure 2.1: a) Details of the matrix calculations of the transformation between input $\boldsymbol{x}$ and $\phi(W\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{h}$ in a deep learning model. $\boldsymbol{x} \in \mathbb{R}^i$ and $\boldsymbol{h} \in \mathbb{R}^j$ for positive integers i, j. b) The mapping ϕ is applied elementwise to each entry of $W\boldsymbol{x} + \boldsymbol{b}$ to produce the entries of $\boldsymbol{h}$.

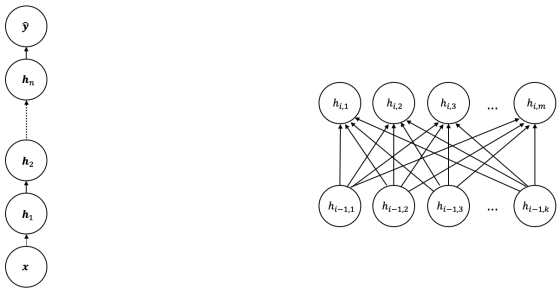
to create a deep model:

$$
\begin{aligned}
\boldsymbol{h}_1 &= \phi_1(W_1\boldsymbol{x} + \boldsymbol{b}_1) \\
\boldsymbol{h}_2 &= \phi_2(W_2\boldsymbol{h}_1 + \boldsymbol{b}_2) \\
&\vdots \\
\boldsymbol{h}_n &= \phi_n(W_n\boldsymbol{h}_{n-1} + \boldsymbol{b}_n) \\
\hat{\boldsymbol{y}} &= \phi_o(W_o\boldsymbol{h}_n + \boldsymbol{b}_o)
\end{aligned}
$$

Here each $\boldsymbol{h}$ denotes an internal layer of the model, also called a hidden layer, and the $\hat{\boldsymbol{y}}$ denotes the output of the model. The type of each of the nonlinear mappings in $\{\phi_1, \phi_2, ..., \phi_n, \phi_o\}$ is usually chosen ahead of training, while the tensors $\boldsymbol{W} = (W_1, W_2, ..., W_n, W_o)$ and $\boldsymbol{B} = (\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n, \boldsymbol{b}_o)$ constitute the parameters of the model that are assigned values during training of the model. The parameters of $\boldsymbol{W}$ are usually called the weights of the model, and those of $\boldsymbol{B}$ are called biases. Sometimes these parameters are also denoted jointly as $\boldsymbol{\theta}$, with $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{B}\}$.

A model made up of an arbitrary number of such layers is called a feedforward neural network, also known as a multi-layer perceptron (MLP). Figure 2.2a shows an example. It is referred to as fully connected, meaning that each entry in a layer is connected to each entry in the next layer, such that every entry in $\boldsymbol{h}_{i-1}$ contributes information to the calculation of every entry in $\boldsymbol{h}_i$. Figure 2.2b illustrates this. The term neural network is also used more generally to refer to any deep learning model.

The neural networks act as function approximators that can approximate any function with accuracy depending on the depth and width of the network. Depth here refers to the number of hidden layers, while width refers to the number of entries in the layers. The universal approximation theorem states that a network with one hidden layer can approximate any continuous function $\boldsymbol{y} = f(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$ for arbitrary positive integers n and m, such that increasing the with of the hidden layer increases the accuracy of the approximation. That is, a network $\hat{\boldsymbol{y}} = g(\boldsymbol{x}) = W_2(\phi_1(W_1\boldsymbol{x} + \boldsymbol{b}_1)$ can be constructed such that for $\boldsymbol{x} \in \mathbb{X}$, $|f(\boldsymbol{x}) - g(\boldsymbol{x})| < \epsilon$ for arbitrarily small ϵ. However, this may not always be feasible in practice due to computational limitations, and more layers are often introduced as an attempt at improving the approximation while restricting the network width.



(a) (b)

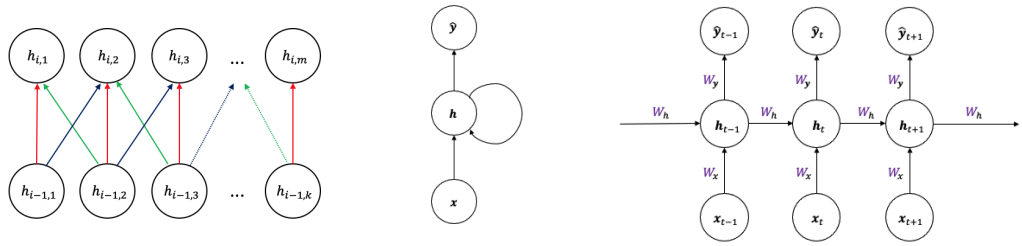
Figure 2.2: a) A common way to draw a graph of the layers of a feedforward neural network. The arrows represent the transformations $\boldsymbol{h}_i = \phi_i(W_i\boldsymbol{h}_{i-1} + \boldsymbol{b}_i)$ from layer to layer. b) This graph shows the connections between entries in two subsequent layers. Each entry in layer $i-1$ contributes information in the calculation of each entry in layer i, weighted by the corresponding entry in W_i. Analogous to the calculations in Figure 2.1b, for $h_{i,1}$ the calculation is $h_{i,1} = w_{i;1,1}h_{i-1,1} + w_{i;1,2}h_{i-1,2} + ...w_{i;1,k}h_{i-1,k} + b_{i;1}$. Here, $\boldsymbol{h}_{i-1} \in \mathbb{R}^k$ and $\boldsymbol{h}_i \in \mathbb{R}^m$, where m and k can be any positive integer chosen independently as part of the model architecture.

Other important types of deep learning models include convolutional neural networks (CNN) and recurrent neural networks (RNN). Convolutional networks make use of simplified connections between layers, removing some connections and using shared parameters for other connections. Figure 2.3a illustrates this concept. These types of models work well with structured data such as images, where pixel location, as well as pixel value, contain information about the data it represents.

Recurrent networks uses backwards connections, which are connections from layers back to themselves or from deeper layers in the model back to preceding layers. This allows the network to remember information from previous input. This is useful when modelling time series data etc. Figures 2.3b and 2.3c illustrates this.

If the dataset $\mathbb{X}$ contains data examples with attached targets $\boldsymbol{y}$, i.e. the task is to perform supervised learning, the output $\hat{\boldsymbol{y}}$ of the network can be modelled to solve both regression and classification tasks. In the case of regression, the output can be interpreted directly, and in the case of classification, if the length of the output vector $\hat{\boldsymbol{y}}$ is set to match the number of classes, it can be passed through an elementwise softmax function to produce normalised probabilities of how likely it is that the input belongs to each class. Regression and classification are both examples of predictive learning tasks.

The network is trained by optimizing a suitable objective function. In the supervised case where $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^n$, where n is the size of the dataset, maximum likelihood estimation is a common way of performing such optimization. The log of the likelihood of the data is maximised in order to learn the parameter values



(a) (b) (c)

Figure 2.3: a) This graph shows sparse connections between layers, where only a few nodes in a layer contributes information to each node in the next layer. Parameters are shared for the remaining connections, indicated by the different coloring of the arrays. In this graph, all red arrays share the same weight parameter, and similarly the green and blue arrows. b) A graph of a recurrent neural network, where there is an additional parametric connection from a hidden layer $\boldsymbol{h}$ back to itself. c) An illustration of the parametric connections in 2.3b when unfolded over several time steps. For each time step t, W_h carries information from $\boldsymbol{h}_{t-1}$ into $\boldsymbol{h}_t$

$\{\boldsymbol{W}, \boldsymbol{B}\}$, which is equivalent to minimizing the negative log-likelihood:

$$J(\boldsymbol{W}, \boldsymbol{B}) = -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\hat{p}_{\mathbb{X}}}[\log p_{\boldsymbol{W},\boldsymbol{B}}(\mathbf{y}|\mathbf{x})] \tag{2.1}$$

Here, $\hat{p}_{\mathbb{X}}$ is the function describing the dataset distribution, and $p_{\boldsymbol{W},\boldsymbol{B}}$ is the function the network optimises.

If $\mathbf{y}|\mathbf{x}$ is assumed to have a Bernoulli distribution, with $\boldsymbol{y} \in \{0, 1\}$, this is equivalent to minimising the cross entropy between the empirical distribution of the training data and $p_{\boldsymbol{W},\boldsymbol{B}}$:

$$J(\boldsymbol{W}, \boldsymbol{B}; \boldsymbol{x}, \boldsymbol{y}) = -\sum_n \boldsymbol{y} \log \hat{\boldsymbol{y}} + (1 - \boldsymbol{y}) \log(1 - \hat{\boldsymbol{y}}) \tag{2.2}$$

If $\mathbf{y}|\mathbf{x}$ is instead assumed to have a Gaussian distribution, maximising the log-likelihood becomes equivalent to minimising the mean squared error between the true value $\boldsymbol{y}$ and the predicted value $\hat{\boldsymbol{y}}$. This is also known as the L^2-error:

$$J(\boldsymbol{W}, \boldsymbol{B}; \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\sum_n \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2 \tag{2.3}$$

The networks are usually trained to optimise the objective function through stochastic gradient descent using backpropagation, an algorithm that allows for efficient calculation of the objective function derivatives with regards to each of the learnable parameters in $\{\boldsymbol{W}, \boldsymbol{B}\}$.

A regularisation term can be added to the objective function to improve the model's ability to generalize to new, unseen data. One way of doing this is by adding restrictions to the values the parameters can take during training. This can be done by adding a second term to the objective function J:

$$J_{reg}(\boldsymbol{W}, \boldsymbol{B}; \boldsymbol{x}, \boldsymbol{y}) = J(\boldsymbol{W}, \boldsymbol{B}; \boldsymbol{x}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{W}, \boldsymbol{B}) \tag{2.4}$$

where α is a hyperparameter weighting the contribution of the new regularization term to the full objective. A simple example is L^2-regularization, where the squared L^2-norm of the $\boldsymbol{W}$ parameters is added to the objective function. The L^2-norm of a vector or matrix is defined as the square root of the sum of all its entries squared, hence for L^2-regularisation, the Ω-term is given as $\Omega(\boldsymbol{W}) = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{l} w_{i,j,k}^2$

A subfield of machine learning that is closely related to deep learning, is representation learning. It is concerned with the layout of the data that is presented to a learning algorithm, and what features are used to describe it. The data representation refers to these features, their content and their location relative to each other. One example is the three-dimensional tensor of pixel values that represents a colour picture. In this case, the tensor is the picture representation, while each tensor entry - representing the blue, green or red content of a single pixel - is a feature. Another example is using vector-valued representations for each word in a vocabulary when dealing with natural language learning tasks. These vectors are called word embeddings.

Machine learning is highly dependent on good data representations in order to get good results on most learning tasks. Choosing the right set of features through feature engineering has been shown to improve performance greatly. However, most techniques for successfully choosing feature representations require some form of human expert involvement and is therefore expensive on large datasets. A lot of research is invested in finding ways that machine learning itself can learn good representations for the data it is presented with, that in turn can be used in downstream tasks like regression, classification etc.

Deep learning models provide subsequent layers $\boldsymbol{h}$, each being a new transformation of the original input $\boldsymbol{x}$. These can be interpreted as new representations of $\boldsymbol{x}$, with a set of new features which are constructed from the original content of $\boldsymbol{x}$, adjusted according to each transformation in order to better solve the learning task. During training the models are implicitly encouraged to recognise the parts of $\boldsymbol{x}$ that are useful in solving the task, while ignoring the parts that are not. Each new layer transformation $\boldsymbol{h}_i$ can be seen as providing a representation that is more abstract relative to the $\boldsymbol{h}_{i-1}$ it is built on top of, hopefully containing features that are better adapted to solving the task at hand.

One example that illustrates this point of view, is the autoencoder. This model is made up of an encoder network, $f_{\boldsymbol{\theta}_f}(\boldsymbol{x})$, and a decoder network, $g_{\boldsymbol{\theta}_g}(\boldsymbol{z})$, shown in Figure 2.4. The encoder takes as input $\boldsymbol{x}$ and calculates an output $\boldsymbol{z} = f_{\boldsymbol{\theta}_f}(\boldsymbol{x})$, and the decoder takes the $\boldsymbol{z}$ as its input and tries to output a reconstruction $\hat{\boldsymbol{x}} = g_{\boldsymbol{\theta}_g}(\boldsymbol{z})$ that is as close to the original $\boldsymbol{x}$ as possible. The model parameters $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ are trained together, often denoted jointly by $\boldsymbol{\theta}$. Training is conducted by using $\boldsymbol{x}$ as both input and target. This can be interpreted as trying to make a good representation $\boldsymbol{z}$ of $\boldsymbol{x}$, since the decoder has to reconstruct $\boldsymbol{x}$ from $\boldsymbol{z}$, with $\boldsymbol{z}$ often called a latent code, or a feature vector. The autoencoders capability to reconstruct $\boldsymbol{x}$ is learned by minimizing an objective function based on a reconstruction error between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}} = g_{\boldsymbol{\theta}_g}(f_{\boldsymbol{\theta}_f}(\boldsymbol{x}))$. Often the autoencoder is regularised in some way to make sure it learns useful representations $\boldsymbol{z}$. $f_{\boldsymbol{\theta}_f}$ and $g_{\boldsymbol{\theta}_g}$ can be modelled as anything from simple single-layer networks to more complex convolutional or

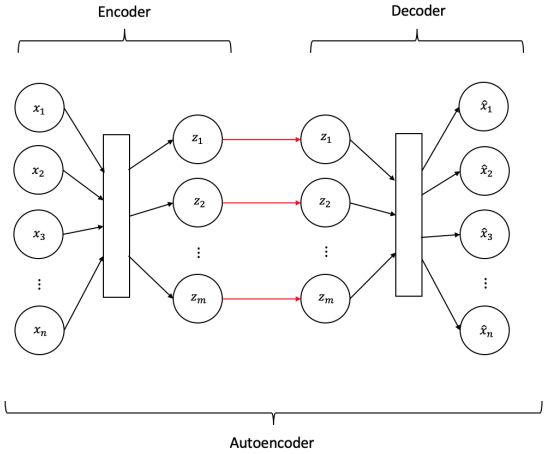


Figure 2.4: The autoencoder, composed of an encoder network and a decoder network. The black arrows represent parametric connections, the rectangles represent an arbitrary number of hidden layers, and the red arrows represent the output of the encoder passed on unchanged as input to the decoder

recurrent networks, depending on the complexity of the input $\boldsymbol{x}$.

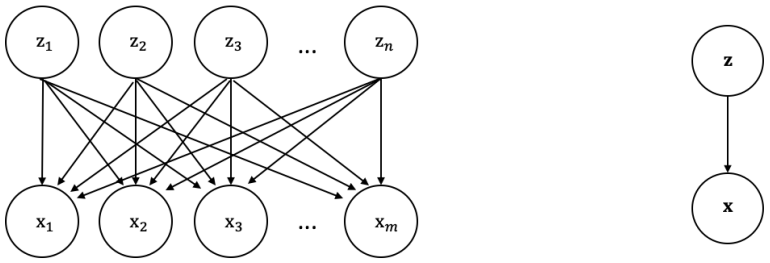
So far the models discussed have been based on deterministic mappings between layers, where moving from one layer to the next is a computational step. Another way to consider this is that the layer values calculated for an input $\boldsymbol{x}$ after training are point estimates of underlying probability distributions. By using the same learning techniques discussed here to train probabilistic models, combined with techniques for approximating inference, layers can be modelled to learn these distributions and to sample values from them. The next section in this chapter as well as Chapter 3 discuss inferring distributions over the values of the hidden layer nodes and the output layer nodes.

The next section covers variational inference, which is a way of approximating inference in complex probabilistic models.

2.3 Variational Inference

Probability theory deals with uncertainty, extending the formal rules of logic to allow for reasoning about uncertain events, in order to determine and quantify the likelihood of the events occurring. Probabilistic learning utilize probability theory to account for uncertainty when learning a model to fit the data. Given a dataset $\mathbb{X}$, probabilistic learning models attempt to find a set of parameters $\boldsymbol{\theta}$ of a function $p_{\boldsymbol{\theta}}$ describing a probability distribution that is likely to have generated the data, i.e. $p_{\boldsymbol{\theta}}$ is a probability density function or a probability mass function.

In this text, $\hat{p}_{\mathbb{X}}$ denotes the probability mass function of the empirical distribution that is determined by the dataset $\mathbb{X}$ at hand. $p^*_{\mathbb{X}}$ denotes the function assumed to describe the true underlying probability distribution seen as having generated the



(a) (b)

Figure 2.5: a) A directed graph showing the conditional dependencies between variables $\mathbf{z}$ and $\mathbf{x}$ that have joint distribution $p(\mathbf{x}, \mathbf{z}) = \prod_n p(\mathrm{z}_n) \prod_m p(\mathrm{x}_m|\mathrm{z}_1, \mathrm{z}_2, ..., \mathrm{z}_n)$. b) A more compact representation of the graph in a).

data in the dataset, often called the data generating distribution. $p_{\boldsymbol{\theta}}$ is the function learned by the model with the goal of approximating $p^*_{\mathbb{X}}$ as closely as possible.

To solve the supervised learning problem of predicting $\hat{\boldsymbol{y}}$ from $\boldsymbol{x}$ using probabilistic learning, would require modelling $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$. Assigning a target value for a given $\boldsymbol{x}$ can then be achieved by sampling a value $\hat{\boldsymbol{y}} \sim p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x} = \boldsymbol{x})$, or by returning the expected value $\hat{\boldsymbol{y}} = \mathbb{E}_{\mathbf{y} \sim p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}=\boldsymbol{x})}[\mathbf{y}]$. Alternatively, an unsupervised learning problem could require modelling the function $p_{\boldsymbol{\theta}}(\mathbf{x})$ by trying to approximate $p^*_{\mathbb{X}}(\mathbf{x})$. In probability learning, the functions $p_{\boldsymbol{\theta}}$ are learned by inference, that is the process of predicting unknown properties or quantities of underlying probability distributions of a set of random variables from a given set of observations for some of these random variables.

The remainder of this section considers the case of unsupervised learning given unlabeled data $\mathbb{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(n)}\}$. The samples in $\mathbb{X}$ is considered observations of a random variable $\mathbf{x}$, referred to as an observed, or visible, random variable. Thus $\boldsymbol{x}^{(i)}$ for $1 \leq i \leq n$ are assumed to be $\boldsymbol{x}^{(i)} \sim p^*_{\mathbb{X}}(\mathbf{x})$. From this dataset, the data distribution $p^*_{\mathbb{X}}(\mathbf{x})$ is wanted modelled by some probabilistic model $p_{\boldsymbol{\theta}}(\mathbf{x})$.

In order to do so, it is often included in the model a set of unobserved random variables, denoted $\mathbf{z}$, of which there are no observations. These are also referred to as hidden or latent random variables. One common way to model the variables $\mathbf{x}$ and $\mathbf{z}$ is to assume that they form a directed graph $\mathbf{z} \to \mathbf{x}$, shown in Figure 2.5. In such a model, each of the variables in $\mathbf{x}$ is dependent on each variable in $\mathbf{z}$, but there are no dependencies within the sets $\mathbf{x}$ and $\mathbf{z}$. The joint probability is therefore:

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \prod_n p(\mathrm{z}_n) \prod_m p(\mathrm{x}_m|\mathrm{z}_1, \mathrm{z}_2, ..., \mathrm{z}_n) \tag{2.5}$$

where m is the number of variables in $\mathbf{x}$ and n is the number of variables in $\mathbf{z}$. The variables $\mathbf{z}$ can be seen as a way of representing the data $\mathbf{x}$, with $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.

The following equation is known as Bayes' rule, or Bayes' theorem:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \tag{2.6}$$

Bayes' rule is the foundation of Bayesian inference. For a set of observations $\boldsymbol{x}$ for the observed variables $\mathbf{x}$, the inference task is often dependent on the computation of a probability distribution over the unobserved variables $\mathbf{z}$, given these observations. When used in the context of inference, the different terms of Bayes' rule are interpreted as follows:

- $p(\mathbf{z})$ - The prior probability. This term reflects any existing belief about the probability of the values of $\mathbf{z}$ before the $\mathbf{x}$ are observed.

- $p(\mathbf{x}|\mathbf{z})$ - The likelihood. This term gives the probability of the observations of $\mathbf{x}$ for values of $\mathbf{z}$.

- $p(\mathbf{x})$ - The marginal likelihood, also called the model evidence. This is the distribution over the observed variables $\mathbf{x}$ marginalised over $\mathbf{z}$: $p(\mathbf{x}) = \int p(\boldsymbol{z})p(\mathbf{x}|\boldsymbol{z})d\boldsymbol{z}$

- $p(\mathbf{z}|\mathbf{x})$ - The posterior probability. This term gives the probability distribution over $\mathbf{z}$ after observing $\boldsymbol{x}$, and is often the term being inferred.

In many probabilistic models, exact inference can be computed using different algorithms. However, the probability models that are usually come across in the context of deep learning, are too complex to allow for exact inference in a reasonable amount of time, and techniques for approximating inference are therefore often used instead.

Variational inference is a general technique to perform approximate inference by viewing inference as an optimization problem, where the goal is to maximize some modelled probability distribution over the observed $\boldsymbol{x}$. Ideally, this would mean solving $\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}} p_{\boldsymbol{\theta}}(\mathbf{x})$, or equivalently $\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$. However, as solving $\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$ is often intractable, the evidence-lower bound (ELBO) on the log-likelihood $\mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$ can be maximized instead. The ELBO is defined to be:

$$\mathcal{L}(\boldsymbol{\theta}, q, \mathbb{X}) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))] \tag{2.7}$$

and is a lower bound on the log-likelihood for any distribution q over $\mathbf{z}$. The D_{KL} term in Equation (2.7) is the Kullback-Leibler divergence (KL divergence), a dissimilarity measure between two probablity distributions q and p, defined as:

$$D_{\mathrm{KL}}(q(\mathbf{x})\|p(\mathbf{x})) = \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[\log \frac{q(\mathbf{x})}{p(\mathbf{x})}] \tag{2.8}$$

$D_{\mathrm{KL}} \geq 0$ always holds, ensuring that Equation (2.7) is indeed a lower bound on the log-likelihood.

While still containing the intractable $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ term in Equation (2.7), the ELBO

objective can be reformulated as follows:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, q, \boldsymbol{x}) &= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbb{E}_{\mathbf{z}\sim q}[\log \frac{q(\mathbf{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{x})}] \\
&= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbb{E}_{\mathbf{z}\sim q}[\log \frac{q(\mathbf{z}|\boldsymbol{x})}{\frac{p_{\boldsymbol{\theta}}(\mathbf{z},\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})}}] \\
&= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbb{E}_{\mathbf{z}\sim q}[\log q(\mathbf{z}|\boldsymbol{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}, \boldsymbol{x}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{x})] \\
&= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbb{E}_{\mathbf{z}\sim q}[\log q(\mathbf{z}|\boldsymbol{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}, \boldsymbol{x})] \\
&= -\mathbb{E}_{\mathbf{z}\sim q}[\log q(\mathbf{z}|\boldsymbol{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})] \\
&= -\mathbb{E}_{\mathbf{z}\sim q}[\log q(\mathbf{z}|\boldsymbol{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})] \\
&= \mathbb{E}_{\mathbf{z}\sim q}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})] - D_{\mathrm{KL}}(q(\mathbf{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}))
\end{aligned} \tag{2.9}$$

Now, the ELBO version in the last line of Equation (2.9) is a lower bound on the maximum log-likelihood of the data that contains neither the term $p_{\boldsymbol{\theta}}(\mathbf{x})$ nor $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, and if q is carefully chosen, $\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, q, \boldsymbol{x})$ will allow for tractable optimization.

One example of this approach is called mean field, where the q distribution is restricted to be a factorial distribution:

$$q(\mathbf{z}|\mathbf{x}) = \prod_i q_i(\mathrm{z}_i|\mathbf{x}) \tag{2.10}$$

The optimization problem would solve for parameters of each q_i.

Generally, restrictions on q can be made according to each specific learning problem in ways that make sure the inference by optimization is tractable.

Chapter 3

Disentangled representation learning

3.1 Introduction

This chapter presents a literature review of disentangled representation learning, with Section 3.2 discussing representation learning in general and Section 3.3 introducing the disentanglement property and the variational autoencoder. Section 3.4 presents a set of unsupervised disentanglement methods, and Section 3.5 presents a set of supervised methods. Parts of this chapter is based on a preliminary review performed in [Bjøru, 2020].

3.2 Representation learning

Section 2.2 introduced the concept of creating good representations for improved machine learning performance. This section continues the discussion on data representations in the context of deep learning.

One important advantage of the deep learning models is their ability to produce distributed representations of the data. A distributed representation is a representation that can encode different descriptive features of the input simultaneously and separately [Hinton, 1986].

For a model that takes an input $\boldsymbol{x}$ and produces a representation vector $r(\boldsymbol{x})$ with 2 binary entries, the different possible distributed representations are $r(\boldsymbol{x}) \in \{(0,0),(1,0),(0,1)(1,1)\}$. The model would be able to separate inputs $\boldsymbol{x}$ into four different groups, or concepts, by assigning them one of the above representations. Inputs assigned to $(0,1)$ and $(1,1)$ could be considered similar in the feature they have in common, encoded by the second entry, whereas $(0,0)$ and $(1,1)$ would differ in both features. For a representation vector of length n, the number of distinct binary-valued representations would be 2^n. Additional representational power is obtained in deep learning models due to the fact that layer nodes, which correspond to representation vector entries, can be continuously valued.

Nondistributed representations can only separate its input based on one feature or feature-combination. To represent four different concepts by binary vector

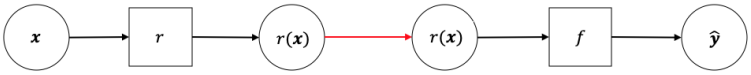


Figure 3.1: The figure illustrates how a representation learning model r and a predictor f can be modelled separately. If $r(\boldsymbol{x})$ is general and not related to any specific f, different models f producing different predictions $\hat{\boldsymbol{y}}$ can be applied to the same representation $r(\boldsymbol{x})$, ideally with better results than when applying f directly to $\boldsymbol{x}$. Priors, e.g. assumptions about the world state underlying the observations $\boldsymbol{x}$, can be utilized by r in order to make $r(\boldsymbol{x})$ better than $\boldsymbol{x}$ at predicting $\hat{\boldsymbol{y}}$.

representations would require four entries, giving representation vectors $r(\boldsymbol{x}) \in \{(1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1)\}$. These representations can only compare input as belonging either to the same concept or not, they do not contain any information about how similar or dissimilar inputs of different concepts are. They are sometimes called one-hot representations, or symbolic representations.

These examples illustrate how distributed representations allow for measuring similarity between inputs in terms of their features. The distributed representations can be seen as located in a multi-dimensional Euclidean space, where each dimension corresponds to a feature, and where distance between two inputs can be measured for instance as the Euclidean distance between their respective representations. Having established the distributed properties of deep learning representations, it is interesting to examine more closely what other feature properties should be attempted obtained. This includes what knowledge each representation feature should encode, as well as how they can be useful in solving the problems at hand.

For any defined supervised problem, the best representation would be the one allowing for the best solution. Trivially, this means that $\boldsymbol{y}$ itself can be considered a representation of its corresponding $\boldsymbol{x}$. However, reducing $\boldsymbol{x}$ to a representation only containing information relevant for assigning the correct $\boldsymbol{y}$ can cause a lot of information in $\boldsymbol{x}$ to be discarded. This limits the chosen representation and would potentially make it useless in other contexts.

Often it is not possible to say right away what information is useful and what is not. In the most general case of unsupervised representation learning, the goal is to create as good a representation $r(\boldsymbol{x})$ as possible of the input $\boldsymbol{x}$ without knowledge about what downstream tasks these representations may later be used for, and this would suggest keeping as much information as possible in the representation. Figure 3.1 illustrates how representations can be learned separately from the tasks they can be helpful in solving.

Bengio et al. [2014] lists several general task-invariant priors about the world that can be useful when exploited in a representation, suggesting what can make a representation $r(\boldsymbol{x})$ better than $\boldsymbol{x}$ at solving downstream tasks. Mentioned in their list are several assumptions about manifolds and generative factors.

The manifold assumption suggests that while the data may be located in a high dimensional space, most data points are located on a region with much smaller dimensionality [Bengio et al., 2014]. This implies good representations can be of significantly lower dimensionality than the original data. The autoencoders introduced in Section 2.2 can be constructed to make use of this assumption, by constricting the representation, i.e. the output $\boldsymbol{z}$ of the encoder, to be of lower dimension than the input. Another hypothesis is that data separates into different manifolds according to different values of certain variables, such as object class. Then, for a set of classes $y \in \{i\}_{i=1}^{n}$, there would be little or no overlap between $p(\mathbf{x}|\mathrm{y} = i)$ for different i. This is referred to as natural clustering [Bengio et al., 2014].

Generative factors refer to the underlying causes that lead to a given world state observed by the data, and can be seen as causing, or generating, the data. This process is discussed further in Section 3.2. Often learning representations can be seen as trying to capture these factors. Many assumptions can be made about these factors. One example is the assumption that they may be organised hierarchically such that lower-level factors may be combined in making more abstract factors [Bengio et al., 2014]. Another useful assumption is that these factors often are considered as having linear dependencies, such that a representation with features corresponding to these factors may be used as input to simple, linear models in solving downstream tasks [Bengio et al., 2014].

A third assumption about these generative factors is that they may be shared by data across the tasks attempted to be solved [Bengio et al., 2014]. This way, knowledge about the data may be increased when considering data from other datasets. Generalising between datasets may be enabled with representations that record these shared factors of the data.

There are different machine learning setups that are illustrative of why good representations matter, and how representations that generalize to help solve different problems can be useful. They are based on the idea of increasing statistical strength through sharing knowledge, where some problem is to be solved based on a dataset of limited size, and the hope is that some related datasets can contribute useful knowledge. Bengio et al. [2014] mentions transfer learning, multi-task learning and domain adaptation as such setups, which have in common that examples from different task-specific datasets may share important features such that knowledge can be shared.

In the transfer learning setup, two different tasks $T1$ and $T2$ are considered, with datasets $\mathbb{X}_{T1}$ and $\mathbb{X}_{T2}$. Often $|\mathbb{X}_{T1}| >> |\mathbb{X}_{T2}|$ and $\boldsymbol{x} \in \mathbb{X}_{T1}$ come from the same distribution as $\boldsymbol{x} \in \mathbb{X}_{T2}$. The idea is that a representation $r(\boldsymbol{x})$ that is useful in mapping $\boldsymbol{x}$ to $\boldsymbol{y}_{T1}$ can also be useful in mapping $\boldsymbol{x}$ to $\boldsymbol{y}_{T2}$.

Multi-task learning refers to the setting where several tasks $T1, T2, ..., Tn$ are considered, where there exists labelled datasets $\mathbb{X}_{T1}, \mathbb{X}_{T2}, ..., \mathbb{X}_{Tn}$ with data from the same distribution, and where some or all of the datasets are limited in size. A shared representation $r(\boldsymbol{x})$ is used to solve all tasks simultaneously.

Domain adaptation describes a setup where a certain task is attempted generalised to data from slightly different distributions, that is they belong to different domains $D1, D2, ..., Dn$. There are labelled datasets $\mathbb{X}_{D1}, \mathbb{X}_{D2}, ..., \mathbb{X}_{Dn}$, and the

learned model should generalise to performing the same task on all these datasets. It is concerned with generalizing between domains, such that domains where there exists little data can take advantage of domain-invariant knowledge from other domains with a lot of data. One interesting extension to domain adaptation is domain generalization, explained in [Muandet et al., 2013], which is the case where the goal is to apply knowledge from known and learned domains to previously unseen domains. That is, the domain examples encountered at test-time is not viewed at training time.

Also mentioned in [Bengio et al., 2014] is the more general setting of semi-supervised learning, which illustrates another way of taking advantage of good representations. The datasets available are typically a large unlabelled dataset $\mathbb{X}_U$ and a much smaller labelled datset $\mathbb{X}_S$. The idea is that using the data in $\mathbb{X}_U$ to help create general representations $r(\boldsymbol{x})$ can help with learning about the mapping from inputs to outputs in $\mathbb{X}_S$.

3.3 Disentangled representations

Returning to the observed random variable $\mathbf{x}$ and the hidden random variable $\mathbf{z}$ from Section 2.3, with the directed graph $\mathbf{z} \to \mathbf{x}$ from Figure 2.5. The joint distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ is often considered as describing a process where $\mathbf{z}$ is interpreted as being a set of generative causes of $\mathbf{x}$. This process is referred to as a generative model. For a $\boldsymbol{z}$ sampled from $p(\mathbf{z})$, $\mathbf{x}$ takes on values $\boldsymbol{x}$ by sampling $\boldsymbol{x} \sim p(\mathbf{x}|\mathbf{z} = \boldsymbol{z})$. Trying to infer distributions $p(\mathbf{z}|\mathbf{x})$ over $\mathbf{z}$ is then interpreted as recovering these causes from the observations.

Recovering $p(\mathbf{z}|\mathbf{x})$ can also be considered as representing $\mathbf{x}$ by $\mathbf{z}$, for instance by using the expected value of $\mathbf{z}$ as the representation $r(\boldsymbol{x})$ for a datapoint $\boldsymbol{x}$. The elements of $\mathbf{z}$ are often called factors, with the terms factors of variation, generative factors and explanatory factors used interchangeably.

Kingma and Welling [2014] introduced a probabilistic autoencoder called a variational autoencoder (VAE) to model $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$. Starting with a dataset $\mathbb{X}$, the VAE assumes as explained above that there exists generative factors $\mathbf{z}$ that produce the observed $\boldsymbol{x}$ following the model $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is considered a probabilistic decoder, modelled using a neural network, and it gives a distribution over $\mathbf{x}$ for input values $\boldsymbol{z}$. $p_{\boldsymbol{\theta}}(\mathbf{z})$ is often chosen to be $\mathcal{N}(\mathbf{z}; \mathbf{0}, I)$, but can also be modelled differently.

Since $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ generally is not tractable to compute in this setting, a model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is introduced as an approximation to $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is modelled using a neural network as a probabilistic encoder. This network then gives a distribution over $\mathbf{z}$ given inputs $\boldsymbol{x}$.

To train the joint model consisting of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{z})$, the lower bound

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}) = \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))] \tag{3.1}$$

from Section 2.3 is required optimised. This is however problematic to differentiate with regards to $\boldsymbol{\phi}$, and [Kingma and Welling, 2014] introduce the stochastic gradient

variational bayes (SGVB) approximation. By using a reparameterization of the $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as $\boldsymbol{z} = f_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}), \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ [Kingma and Welling, 2014; Rezende et al., 2014], the approximation for an input $\boldsymbol{x}^{(i)}$ becomes:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}) = \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)}) - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x}^{(i)})\|p(\mathbf{z})) \tag{3.2}$$

Where $\boldsymbol{z}^{(i,l)} = f_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \boldsymbol{x}^{(i)}), \boldsymbol{\epsilon}^{(i,l)} \sim p(\boldsymbol{\epsilon})$. L is usually set to 1.

As an example, assume $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 I)$. The probabilistic encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can then be implemented as a neural network $f_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$, that performs the mapping $f_1 : \boldsymbol{x} \to (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ followed by $f_2 : (\boldsymbol{\epsilon}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \to \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$. That gives $f_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}) = f_2(\boldsymbol{\epsilon}, f_1(\boldsymbol{x})) = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon} = \boldsymbol{z}$. In the case where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 I)$ and $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, I)$, the D_{KL}-term can be calculated analytically as:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x}^{(i)})\|p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2) \tag{3.3}$$

Rezende et al. [2014] independently introduced the same reparameterization of variables $\mathbf{z}$, and illustrated its use on a model with more than one layer of hidden random variables $\mathbf{z}$.

With this reparameterization, the approximation $\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ is differentiable w.r.t both $\boldsymbol{\theta}, \boldsymbol{\phi}$ and can be optimized using stochastic gradient descent. The optimisation on a dataset $\mathbb{X}$ involves point estimation on the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, and variational inference on the variables $\mathbf{z}$. This algorithm is general and works well both in the case of intractability concerning $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, and for large datasets. A trained model allows for generating artificial data resembling the data in $\mathbb{X}$, by sampling $\boldsymbol{z}$ from $p(\mathbf{z})$ and using the probabilistic decoder to sample $\boldsymbol{x}$ from $p(\mathbf{x}|\mathbf{z})$. It can also provide an approximation of $p(\mathbf{x})$.

The true generative causes of an observation $\boldsymbol{x}$ are usually assumed to consist of semantically meaningful elements, such that a change in a generative cause as interpreted by humans would be reflected by a change along one dimension in the space of the true generative causes [Bengio et al., 2014; Higgins et al., 2018]. For observations $\boldsymbol{x}$ that are pictures of objects, examples of such generative cause can be the colour of the object, or the location of the object. They change one aspect of the observation, while leaving the rest invariant.

Recovering $\mathbf{z}$ using the VAE framework often results in a $\mathbf{z}$ where the generative causes are entangled, such that adjusting a single variable value of $\mathbf{z}$ may cause multiple changes in $\mathbf{x}$. When representing an observation $\boldsymbol{x}$ by $r(\boldsymbol{x}) = \mu_{\mathbf{z}}$, $r(\boldsymbol{x})$ would be entangled if some or all of its features encodes more than one of the underlying generative factors. $r(\boldsymbol{x})$ is considered disentangled if each feature is influenced only by a single semantically interpretable underlying factor, such that a change in one feature only changes one aspect of the observation it represent [Higgins et al., 2018]. The representation is then said to disentangle the underlying causes.

The disentangled representation approach to learning representations assumes a benefit from disentangling the underlying structure of the world state behind an observation into separate parts of the representation. There is not yet a single agreed

upon definition of disentangled representations in the deep learning community, however three properties of disentangled representations have been considered by several contributors as a basis for discussion [Eastwood and Williams, 2018; Ridgeway and Mozer, 2018; Higgins et al., 2018; Zaidi et al., 2021]. They are usually referred to as modularity, compactness and explicitness.

Modularity is satisfied when each dimension in $\mathbf{z}$ encodes information about at most one generative factor. This property ensures that factors that are mutually independent in the generative factor space are also independent in $\mathbf{z}$-space [Zaidi et al., 2021]. This property is agreed upon by most approaches to disentanglement as a necessary criterion for disentanglement [Higgins et al., 2018].

Compactness is satisfied if each generative factor is encoded by at most one dimension in $\mathbf{z}$. This property is less agreed upon, with many arguing that compactness should not be a criterion for disentanglement [Ridgeway and Mozer, 2018; Higgins et al., 2018; Zaidi et al., 2021]. Arguments against compactness suggest that allowing multiple $\mathbf{z}$ dimensions to encode a single factor will increase the flexibility of the model, both in terms of factor complexity and model optimization. Multiple $\mathbf{z}$ dimensions encoding a factor would allow for encoding of complex factors that are not easily represented by one dimensional variables [Zaidi et al., 2021], which are likely to occur when working with complex data. It will also allow for more than one equivalent solution when encoding any factor, such that training is less likely to get stuck in local optima [Ridgeway and Mozer, 2018].

Explicitness refers to whether the values of all of the generative factors can be decoded from a representation $\mathbf{z}$. This property requires that all factors are encoded in $\mathbf{z}$, and that they are encoded in such a way that little or no information is lost. In order to satisfy explicitness, a transformation T from $\mathbf{z}$-space to factor space should be implicitly learned by the model that generalise the relationship between $\mathbf{z}$ and the generative factors. So far, this property is agreed upon by most disentanglement approaches. Some additionally argue that the transformation T should be simple, and ideally linear [Ridgeway and Mozer, 2018]. Explicitness is therefore often discussed distinguishing between two versions, non-linear and linear explicitness [Higgins et al., 2018; Zaidi et al., 2021].

Higgins et al. [2018] make an attempt at formally defining disentangled representations by starting with the transformation properties of the world, saying it is the transformations that only change some aspects of the world state, while leaving the rest unchanged, that will give data structure that can be exploited in representation learning. Higgins et al. connects the concept of symmetry transformations from physics - covering rotations, translations etc. - with vector representations through group theory, and use this to define disentangled representations. A vector representations qualifies as disentangled if it decomposes into a set of subspaces such that each subspace is independently transformed by a corresponding symmetry transformation. The definition presented is:

> A 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., 2018, p. 6].

In addition they also present a definition of a linear disentangled representation, which adds the criterion that the actions of the subgroups on their subspaces are linear.

3.4 Unsupervised disentanglement

Unsupervised disentanglement refers to the case of attempting to recover the semantically meaningful generative causes in a disentangled representation from the observations $\boldsymbol{x} \in \mathbb{X}$ through inferring $p(\mathbf{z}|\mathbf{x})$ over $\mathbf{z}$. In this setting, no knowledge is available about $\mathbf{z}$ apart from the assumptions made about the prior $p(\mathbf{z})$. Locatello et al. [2019] show that unsupervised disentangled representation learning is impossible unless inductive biases are present in both the model architecture and the data.

Several attempts at creating models that perform unsupervised disentanglement have been suggested that are based on introducing some adjustment to the objective of the VAE model, to try to improve the models disentangling performance. One such framework is the β-VAE [Higgins et al., 2017], where the VAE objective is modified to

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}, \beta) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - \beta D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))] \tag{3.4}$$

Here, one new parameter β is introduced that controls the contribution of the KL-divergence to the total objective. Higgins et al. arrive at this objective by first considering the reconstruction loss $\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]]$ as their objective function, and then adding the constraint that $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) < \epsilon$. Adding this constraint is seen as trying to match $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to a prior $p(\mathbf{z})$ set to the factorial distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, I)$ to encourage disentangling. Because the prior is assumed to be a factorial distribution, its separate dimensions are linearly independent, and it is argued that minimizing the divergence between such a prior $p(\mathbf{z})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ should encourage the dimensions of the latter to capture independent pieces of information from $\mathbf{x}$.

The β-VAE can be implemented similarly to the VAE using neural networks as probabilistic encoder and decoder, including β in the objective as an additional hyperparameter chosen ahead of training. By choosing values of β such that $\beta > 1$, the model should be encouraged to increase disentanglement of the representations created [Higgins et al., 2017]. However, this may lead to a trade off between disentanglement and reconstruction error, where high values of β leads to lower quality of reconstructions.

In [Alemi et al., 2019] another way of arriving at the lower bound in Equation (3.4) is presented through a variational approximation to the information bottleneck [Tishby et al., 2000]. For a model taking input observations of $\mathbf{x}$ and learning a stochastic representation $\mathbf{z}$ and a reconstruction of $\mathbf{x}$ from this representation $\mathbf{z}$, the unsupervised version of the information bottleneck objective is

$$\max I(\mathbf{z}, \mathbf{x}) - \beta I(\mathbf{z}, \boldsymbol{x}^{(i)}) \tag{3.5}$$

$I(\mathbf{z}, \mathbf{x})$ is the mutual information between random variables $\mathbf{z}, \mathbf{x}$. $I(\mathbf{z}, \boldsymbol{x}^{(i)})$ is the mutual information between random variable $\mathbf{z}$ and the i-th datapoint $\boldsymbol{x}^{(i)}$. This is understood as maximising the mutual information between $\mathbf{z}$ and $\mathbf{x}$ in order to improve reconstruction, while encouraging that the representation variables $\mathbf{z}$ remembers as little information about each datapoint $\boldsymbol{x}^{(i)}$ as possible. $\mathbf{z}$ is considered a bottleneck limiting the information about $\mathbf{x}$ that can pass from input to reconstruction. Creating a variational lower bound on Equation (3.5) results in the same objective as the one introduced for the β-VAE [Alemi et al., 2019].

Burgess et al. [2018] considers the information bottleneck perspective and presents a further modification to the β-VAE. Thinking of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as the bottleneck for the reconstruction $\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]]$, the term $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ is seen as bounding the information allowed through $\mathbf{z}$. This upper bound is referred to as the capacity of $\mathbf{z}$. When $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$ the capacity is zero, and the model will not be able to store any information about $\mathbf{x}$ in $\mathbf{z}$

The objective introduced by Burgess et al. is

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}, \gamma, C) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - \gamma|D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) - C|] \quad (3.6)$$

Here, γ is a hyperparameter chosen ahead of training, while C is set to 0 at the beginning of training and increased during training. Starting with $C = 0$ is considered enforcing maximum limitation on the capacity for a chosen γ, intuitively explained as forcing $\mathbf{z}$ to keep only the most important information contained in $\mathbf{x}$. This is assumed to result in the representation focusing on remembering the most important generative cause [Burgess et al., 2018]. When C is increased it decreases the pressure on $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to be close to $p(\mathbf{z})$ and thus increases the capacity of $\mathbf{z}$. This allows the model to expand its representation with more information about $\mathbf{x}$, eventually moving on to less and less informative generative causes, thus resulting in a disentangled representation. The model trained with this objective is called Controlled Capacity Increase (CCI)-VAE. It is sometimes also referred to as AnnealedVAE.

Kim and Mnih [2019] introduces a method for unsupervised disentangling called FactorVAE. This framework attempts to drive the marginal distribution of $\mathbf{z}$, $q(\mathbf{z})$ to be factorial, and therefore encourage the dimensions of the distribution to be independent of each other. This distribution is given by

$$q(\mathbf{z}) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[q(\mathbf{z}|\mathbf{x})] = \frac{1}{n}\sum_{i=1}^{n} q(\mathbf{z}|\boldsymbol{x}^{(i)}) \quad (3.7)$$

where $\hat{p}_{\mathbb{X}}$ is the empirical distribution given by the dataset.

The FactorVAE method attempts to improve on the trade-off between disentanglement and reconstruction quality seen in β-VAE, obtaining better disentanglement for a given reconstruction error. The reasoning behind this approach is based on the following decomposition of the KL-term as it appears in the original VAE as well as in the β-VAE objective function:

$$\mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))] = I(\mathbf{x}, \mathbf{z}) + D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z})) \quad (3.8)$$

Minimizing the original KL-term would lead to minimizing both the mutual information $I(\mathbf{x}, \mathbf{z})$ between $\mathbf{x}$ and $\mathbf{z}$, and the KL-distance between $q(\mathbf{z})$ and $p(\mathbf{z})$. Keeping the constraint from β-VAE that $p(\mathbf{z})$ is chosen to be factorial, minimizing $D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}))$ encourages disentanglement of the dimensions of $\mathbf{z}$. However, minimizing the mutual information between $\mathbf{x}$ and $\mathbf{z}$ will limit the amount of information about $\mathbf{x}$ that can be stored in $\mathbf{z}$, which will in turn limit reconstruction quality [Kim and Mnih, 2019].

The FactorVAE objective [Kim and Mnih, 2019] is:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}, \gamma) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))] \\ - \gamma D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z})\|\bar{q}_{\boldsymbol{\phi}}(\mathbf{z}))\end{aligned} \tag{3.9}$$

where $\bar{q}_{\boldsymbol{\phi}}(\mathbf{z}) = \prod_{j=0}^{l} q_{\boldsymbol{\phi}}(\mathrm{z}_j)$. $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z})\|\bar{q}_{\boldsymbol{\phi}}(\mathbf{z}))$ is the Total Correlation (TC), a measure of dependence between the variables $\mathbf{z}$. Since both $q_{\boldsymbol{\phi}}(\mathbf{z})$ and $\bar{q}_{\boldsymbol{\phi}}(\mathbf{z})$ are intractable to compute directly, this term is approximated by using a discriminator d that is trained to output an estimate $d(\boldsymbol{z})$ of the probability that the input $\boldsymbol{z}$ it gets is sampled from $q_{\boldsymbol{\phi}}(\mathbf{z})$, and not $\bar{q}_{\boldsymbol{\phi}}(\mathbf{z})$. Input samples to the discriminator is obtained by sampling from $q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x}^{(i)})$, for datapoints $\boldsymbol{x}^{(i)}$ chosen uniformly at random, to obtain samples from $q_{\boldsymbol{\phi}}(\mathbf{z})$. Samples from $\bar{q}_{\boldsymbol{\phi}}(\mathbf{z})$ are approximated by sampling a batch from $q_{\boldsymbol{\phi}}(\mathbf{z})$ and then randomly permuting each dimension across the batch [Kim and Mnih, 2019].

The approximation used is:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z})\|\bar{q}_{\boldsymbol{\phi}}(\mathbf{z})) = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z})}{\bar{q}_{\boldsymbol{\phi}}(\mathbf{z})}] \approx \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\log \frac{d(\mathbf{z})}{1 - d(\mathbf{z})}] \tag{3.10}$$

and the discriminator and the VAE are trained jointly.

Chen et al. [2019] arrive at the same objective as shown in Equation (3.9) based on a similar argument of decomposing the KL-term in the original VAE objective and increasing the weight on the TC component to encourage disentangling. The method suggested, β-TCVAE, uses a different approach to evaluating the TC component. The following estimator is used as a lower bound on $\log q_{\boldsymbol{\phi}}(\mathbf{z})$, computed on a mini batch of data of size m:

$$\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\log q_{\boldsymbol{\phi}}(\mathbf{z})] \approx \frac{1}{m}\sum_{i=1}^{m}[\log \frac{1}{nm}\sum_{j=1}^{m} q_{\boldsymbol{\phi}}(\mathbf{z}(\boldsymbol{x}^{(i)})|\boldsymbol{x}^{(j)})] \tag{3.11}$$

Where $\mathbf{z}(\boldsymbol{x}^{(i)})$ is a sample from $q(\mathbf{z}|\boldsymbol{x}^{(i)})$, and n is the size of the total dataset. The derivation of this estimator is included in Appendix A. The method using this estimator is called Minibatch Weighted Sampling (MWS) [Chen et al., 2019].

Kumar et al. [2018] presents an alternative approach, also attempting to regularize $q(\mathbf{z})$ to be factorial. The objective function suggested is:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}, \lambda) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))] \\ - \lambda D(q_{\boldsymbol{\phi}}(\mathbf{z})\|p(\mathbf{z}))\end{aligned} \tag{3.12}$$

The term $D(q_{\boldsymbol{\phi}}(\mathbf{z})\|p(\mathbf{z}))$ is intractable if D is taken to be the KL-divergence, and D is therefore instead suggested measured as the matching of the covariance of

$q_{\boldsymbol{\phi}}(\mathbf{z})$ and $p(\mathbf{z})$ [Kumar et al., 2018]. Assuming $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, I)$, matching the covariance will require decorrelating the dimensions of $q_{\boldsymbol{\phi}}(\mathbf{z})$. Considering the case where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x}))$, the covariance of $q_{\boldsymbol{\phi}}(\mathbf{z})$ is given, by the law of total covariance, as:

$$\begin{aligned} \mathrm{cov}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}] &= \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}\mathrm{cov}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\mathbf{z}] + \mathrm{cov}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\mathbf{z}] \\ &= \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x})] + \mathrm{cov}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})] \end{aligned} \tag{3.13}$$

When $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x})$ is assumed to be diagonal, $\mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x})]$ will be diagonal and the correlations between the $\mathbf{z}$ dimensions are captured in $\mathrm{cov}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})]$ alone. One method for regularizing $\mathrm{cov}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}]$ to be close to I is therefore to consider only the $\mathrm{cov}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})]$ term, in order to remove these correlations [Kumar et al., 2018]. Matching $\mathrm{cov}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})]$ to I is done by reducing the element-wise squared L^2-norm, and the resulting objective function is:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}, \lambda_{od}, \lambda_d) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))] \\ - \lambda_{od}\sum_{i\neq j}[\mathrm{cov}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})]]^2_{ij} \\ - \lambda_d\sum_i([\mathrm{cov}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})]]_{ii} - 1)^2 \end{aligned} \tag{3.14}$$

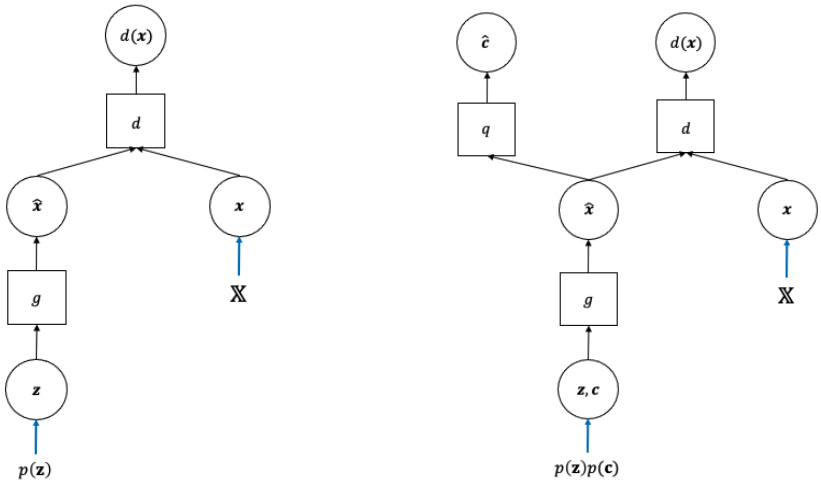
Here, two hyperparameters λ_{od} and λ_d are introduced in order to control the contribution of the matching of the off-diagonal and diagonal entries of $\mathrm{cov}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})]$ to I respectively. The framework based on maximising this objective is called DIP-VAE-I [Kumar et al., 2018].

An analogous framework where the full $\mathrm{cov}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}]$ is matched to I is also presented, called DIP-VAE-II [Kumar et al., 2018]. The accompanying objective function is:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}, \lambda_{od}, \lambda_d) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))] \\ - \lambda_{od}\sum_{i\neq j}[\mathrm{cov}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}]]^2_{ij} \\ - \lambda_d\sum_i([\mathrm{cov}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}]]_{ii} - 1)^2 \end{aligned} \tag{3.15}$$

For both objectives, it is the term controlled by λ_{od} that encourages the covariance between different dimensions of $\mathbf{z}$ to approach 0 and therefore encourage independence and consequently disentanglement. The term controlled by λ_d does not directly contribute to disentangling the factors of $\mathbf{z}$, but setting $\lambda_d = 0$ would allow the model to minimize the off-diagonals by driving the diagonal variance terms towards 0, instead of by decorrelating $\mathbf{z}$. λ_d should therefore be > 0 in order to keep diagonal entries close to one, ensuring that the minimization of the off-diagonals is due to independence [Kumar et al., 2018].

While all models presented in this subsection so far have been based on variations over the VAE objective, other approaches have also been considered. InfoGAN [Chen et al., 2016] is a method for disentangling that is based on the Generative



(a) (b)

Figure 3.2: a) The Generative Adversairal Network architecture. The blue arrows indicates how the inputs are sampled, with $\mathbb{X}$ being the data set. The generator g and the discriminator d are typically parametrised by neural networks, and they are trained using the value function presented in Equation (3.16). The output $d(\boldsymbol{x})$ of the discriminator is the probability that its input comes from the data set and not the generator. b) The extended architecture of the InfoGAN model. The output $\hat{\boldsymbol{c}}$ of the new component q is encouraged to reconstruct the input $\boldsymbol{c}$ to the g component by the term $\mathbb{E}_{\mathbf{c}\sim p(\mathbf{c}),\mathbf{x}\sim g_{\boldsymbol{\theta}_g}(\mathbf{z},\mathbf{c})}[\log q_{\boldsymbol{\phi}}(\mathbf{c}|\mathbf{x})]$ of the extended value function shown in Equation (3.18).

Adversarial Network (GAN) framework [Goodfellow et al., 2014]. The original GAN framework is made up of two models trained simultaneously, a generative model g and a discriminative model d. g takes as input some random noise $\boldsymbol{z} \sim p(\mathbf{z})$ and attempts to learn to output samples from the training data distribution, while d, when given an input sample, is trained to output the probability that the input belongs to the training data rather than being produced by g. g is trained to maximise the probability that d makes a mistake. This competition between g and d, also referred to as a minimax game, eventually leads to g drawing samples that d cannot distinguish from samples from the true data distribution [Goodfellow et al., 2014].

For g and d parametrized by neural networks, the value function of the model is:

$$\min_{\boldsymbol{\theta}_g}\max_{\boldsymbol{\theta}_d} V(\boldsymbol{\theta}_d, \boldsymbol{\theta}_g; \mathbb{X}) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathbb{X}}} \log d_{\boldsymbol{\theta}_d}(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[1 - d_{\boldsymbol{\theta}_d}(g_{\boldsymbol{\theta}_g}(\mathbf{z}))] \tag{3.16}$$

where $\boldsymbol{\theta}_d$ are the parameters of d and $\boldsymbol{\theta}_g$ are the parameters of g. $p(\mathbf{z})$ is the prior distribution on the noise variables $\mathbf{z}$ that are used as input to g. The full architecture is shown in Figure 3.2.

The InfoGAN extends this value function in order to allow for disentanglement.

In addition to the noise $\mathbf{z}$, a latent code $\mathbf{c}$ is introduced, with the assumed distribution $p(\mathbf{c}) = \prod_{i=1}^{l} p(\mathbf{c}_i)$. The generator g is adapted to take as input both $\mathbf{z}$ and $\mathbf{c}$, and in order to force g to consider meaningful $\mathbf{c}$, the objective is regularized by favouring high mutual information $I(\mathbf{c}, g(\mathbf{z}, \mathbf{c}))$ between $\mathbf{c}$ and the output of g for that $\mathbf{c}$.

$I(\mathbf{c}, g(\mathbf{z}, \mathbf{c}))$ is difficult to optimize directly, and a lower bound is therefore used [Chen et al., 2016]. A distribution $q(\mathbf{c}|\mathbf{x})$ is introduced to approximate $p(\mathbf{c}|\mathbf{x})$, and the lower bound on $I(\mathbf{c}, g(\mathbf{z}, \mathbf{c}))$ is:

$$\mathcal{L}_I(\boldsymbol{\theta}_g, \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{c}\sim p(\mathbf{c}), \mathbf{x}\sim g_{\boldsymbol{\theta}_g}(\mathbf{z},\mathbf{c})}[\log q_{\boldsymbol{\phi}}(\mathbf{c}|\mathbf{x})] + H(\mathbf{c}) \tag{3.17}$$

where q is paramterized by a neural network with parameters $\boldsymbol{\phi}$, and $H(\mathbf{c})$ is the entropy over $\mathbf{c}$, which can be considered a constant.

Now, the InfoGAN value function is:

$$\min_{\boldsymbol{\theta}_g, \boldsymbol{\phi}} \max_{\boldsymbol{\theta}_d} V_{\text{InfoGAN}}(\boldsymbol{\theta}_d, \boldsymbol{\theta}_g, \boldsymbol{\phi}; \mathbb{X}, \lambda) = V(\boldsymbol{\theta}_d, \boldsymbol{\theta}_g; \mathbb{X}) - \lambda \mathcal{L}_I(\boldsymbol{\theta}_g, \boldsymbol{\phi}) \tag{3.18}$$

with a hyperparameter λ that controls the contribution of the introduced regularization term. The InfoGAN architecture is shown in Figure 3.2b.

3.5 Supervised disentanglement

Supervised disentanglement is concerned with datasets $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{X}$ and refers to the case of trying to create disentangled representations $\boldsymbol{z}$ of inputs $\boldsymbol{x}$ such that a target $\boldsymbol{y}$ can be assigned to $\boldsymbol{x}$ using $\boldsymbol{z}$ [Bengio et al., 2014]. It is often reasonable to assume that $\boldsymbol{y}$ is closely related to one or more generative causes in $\boldsymbol{z}$. Sometimes $\boldsymbol{y}$ is considered a generative cause itself, and could therefore be recovered as a subspace of $\boldsymbol{z}$. It would then be straightforward to assign $\boldsymbol{y}$ from $\boldsymbol{z}$. Often supervised disentanglement is considered as part of a semi-supervised approach, allowing for improvement on training using additional unlabeled data.

Louizos et al. [2017] present a semi-supervised model based on the VAE that encourage independence between the representation given by recovered generative factors $\boldsymbol{z}$ and some known sensitive factors $\boldsymbol{s}$. Prediction on targets $\boldsymbol{y}$ can then be performed using $\boldsymbol{z}$ as input. The datasets are $(\boldsymbol{x}, \boldsymbol{s}) \in \mathbb{X}_U$ and $(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y}) \in \mathbb{X}_S$, with $\mathbb{X} = \{\mathbb{X}_U, \mathbb{X}_S\}$. The generative model is:

$$p(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \mathbf{s}, \mathbf{y}) = p_{\boldsymbol{\theta}_1}(\mathbf{x}|\mathbf{z}_1, \mathbf{s}) p_{\boldsymbol{\theta}_2}(\mathbf{z}_1|\mathbf{z}_2, \mathbf{y}) p(\mathbf{z}_2) p(\mathbf{y}) \tag{3.19}$$

A graph of this model is shown in Figure 3.3a. The variational approximation to the posterior $q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{y}|\mathbf{x}, \mathbf{s})$ is assumed factorized as:

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{y}|\mathbf{x}, \mathbf{s}) = q_{\boldsymbol{\phi}_1}(\mathbf{z}_1|\mathbf{x}, \mathbf{s}) q_{\boldsymbol{\phi}_2}(\mathbf{y}|\mathbf{z}_1) q_{\boldsymbol{\phi}_3}(\mathbf{z}_2|\mathbf{z}_1, \mathbf{y}) \tag{3.20}$$

The model is trained by maximizing a lower bound on $\mathbb{E}_{(\mathbf{x},\mathbf{s})\sim\hat{p}_{\mathbb{X}}}[\log p(\mathbf{x}|\mathbf{s})]$, which becomes:

$$\begin{aligned}\mathbb{E}_{(\mathbf{x},\mathbf{s})\sim\hat{p}_{\mathbb{X}}}[\mathbb{E}_{(\mathbf{z}_1,\mathbf{z}_2,\mathbf{y})\sim q_{\boldsymbol{\phi}}(\mathbf{z}_1,\mathbf{z}_2,\mathbf{y}|\mathbf{x},\mathbf{s})}[\log p(\mathbf{y}) + \log p(\mathbf{z}_2) + \log p_{\boldsymbol{\theta}_2}(\mathbf{z}_1|\mathbf{z}_2, \mathbf{y}) \\ + \log p_{\boldsymbol{\theta}_1}(\mathbf{x}|\mathbf{z}_1, \mathbf{s}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{y}|\mathbf{x}, \mathbf{s})]]\end{aligned} \tag{3.21}$$

For data points from $\mathbb{X}_U$, the lower bound evaluates to:

$$\begin{aligned}\mathcal{L}_U(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}_U) = \mathbb{E}_{(\mathbf{x},\mathbf{s})\sim\hat{p}_{\mathbb{X}_U}}[\mathbb{E}_{\mathbf{z}_1\sim q_{\boldsymbol{\phi}_1}(\mathbf{z}_1|\mathbf{x},\mathbf{s})}[\log p_{\boldsymbol{\theta}_1}(\mathbf{x}|\mathbf{z}_1,\mathbf{s}) \\ - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}_2}(\mathbf{y}|\mathbf{z}_1)\|p(\mathbf{y}))] \\ + \mathbb{E}_{(\mathbf{z}_1,\mathbf{y})\sim q_{\boldsymbol{\phi}_1,\boldsymbol{\phi}_2}(\mathbf{z}_1,\mathbf{y}|\mathbf{x},\mathbf{s})}[-D_{\mathrm{KL}}(q_{\boldsymbol{\phi}_3}(\mathbf{z}_2|\mathbf{z}_1,\mathbf{y})\|p(\mathbf{z}_2))] \\ + \mathbb{E}_{(\mathbf{z}_1,\mathbf{y},\mathbf{z}_2)\sim q_{\boldsymbol{\phi}}(\mathbf{z}_1,\mathbf{y},\mathbf{z}_2|\mathbf{x},\mathbf{s})}[\log p_{\boldsymbol{\theta}_2}(\mathbf{z}_1|\mathbf{z}_2,\mathbf{y}) \\ - \log q_{\boldsymbol{\phi}_1}(\mathbf{z}_1|\mathbf{x},\mathbf{s})]]\end{aligned} \tag{3.22}$$

For data points from $\mathbb{X}_S$, the lower bound evaluates to:

$$\begin{aligned}\mathcal{L}_S(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}_S) = \mathbb{E}_{(\mathbf{x},\mathbf{s},\mathbf{y})\sim\hat{p}_{\mathbb{X}_S}}[\mathbb{E}_{\mathbf{z}_1\sim q_{\boldsymbol{\phi}_1}(\mathbf{z}_1|\mathbf{x},\mathbf{s})}[\log p_{\boldsymbol{\theta}_1}(\mathbf{x}|\mathbf{z}_1,\mathbf{s}) \\ - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}_3}(\mathbf{z}_2|\mathbf{z}_1,\mathbf{y})\|p(\mathbf{z}_2))] \\ + \mathbb{E}_{\mathbf{z}_1\sim q_{\boldsymbol{\phi}_1}(\mathbf{z}_1|\mathbf{x},\mathbf{s}),\mathbf{z}_2\sim q_{\boldsymbol{\phi}_3}(\mathbf{z}_2|\mathbf{z}_1,\mathbf{y})}[\log p_{\boldsymbol{\theta}_2}(\mathbf{z}_1|\mathbf{z}_2,\mathbf{y}) \\ - \log q_{\boldsymbol{\phi}_1}(\mathbf{z}_1|\mathbf{x},\mathbf{s})]]\end{aligned} \tag{3.23}$$

The parameters $\boldsymbol{\phi}_2$ of $q_{\boldsymbol{\phi}_2}(\mathbf{y}|\mathbf{z}_1)$ are not updated by the $\mathcal{L}_S$ objective, and a classification objective $\mathbb{E}_{(\mathbf{x},\mathbf{s},\mathbf{y})\sim\hat{p}_{\mathbb{X}_S}}[\mathbb{E}_{\mathbf{z}_1\sim q_{\boldsymbol{\phi}_1}(\mathbf{z}_1|\mathbf{x},\mathbf{s})}\log q_{\boldsymbol{\phi}_2}(\mathbf{y}|\mathbf{z}_1)]$ is added to ensure that the parameters learn from all available data. The full objective becomes:

$$\begin{aligned}\mathcal{F}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}, \alpha) = \mathcal{L}_U(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}_U) + \mathcal{L}_S(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbb{X}_S) \\ + \alpha\mathbb{E}_{(\mathbf{x},\mathbf{s},\mathbf{y})\sim\hat{p}_{\mathbb{X}_S}}[\mathbb{E}_{\mathbf{z}_1\sim q_{\boldsymbol{\phi}_1}(\mathbf{z}_1|\mathbf{x},\mathbf{s})}\log q_{\boldsymbol{\phi}_2}(\mathbf{y}|\mathbf{z}_1)]\end{aligned} \tag{3.24}$$

with hyperparameter α to control contribution from the classification objective.

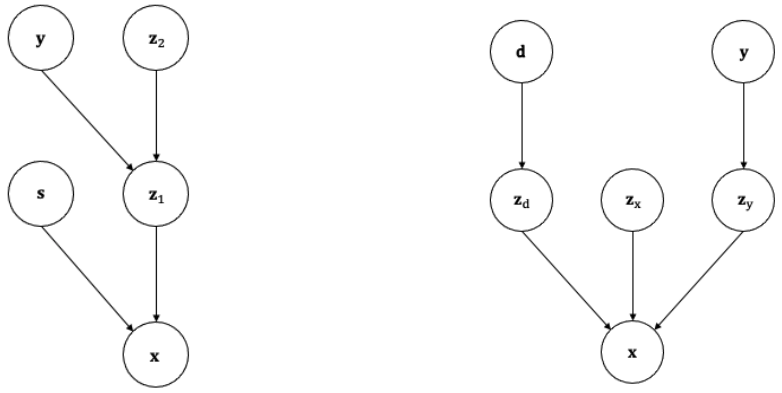
While this model encourages independence between $\mathbf{s}$ and $\mathbf{z}_1$, it is still possible that some information from $\mathbf{s}$ will be encoded in $\mathbf{z}_1$, especially if $\mathbf{s}$ and $\mathbf{y}$ are correlated. In order to further encourage independence $\mathbf{s}$ and $\mathbf{z}_1$, a regularization term is added to the lower bound objective that penalizes dependence in the marginal posterior $q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{s})$. Maximum mean discrepancy (MMD) is considered for this, and a term $D_{\mathrm{MMD}}(q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{x},\mathbf{s}=s)\|q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{x},\mathbf{s}=s'))$ is added to the objective [Louizos et al., 2017].

Ilse et al. [2019] introduces a model for supervised disentanglement based on the VAE framework that separates the $\mathbf{z}$-space into independent subspaces, called Domain Invariant Variational Autoencoder (DIVA). The model considers a dataset $(\boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y}) \in \mathbb{X}$, with data from n different domains $\boldsymbol{d}$. The approach is based on the assumption that a subspace of $\mathbf{z}$ is invariant with respect to the domain.

The $\mathbf{z}$ is divided into subspaces $\mathbf{z}_\mathrm{d}$, $\mathbf{z}_\mathrm{y}$ and $\mathbf{z}_\mathrm{x}$. $\mathbf{z}_\mathrm{d}$ is domain specific, capturing variation in domains $\mathbf{d}$, and $\mathbf{z}_\mathrm{y}$ captures variation in labels $\mathbf{y}$. $\mathbf{z}_\mathrm{x}$ captures any remaining variation in $\mathbf{x}$ that is invariant with respect to both $\mathbf{y}$ an $\mathbf{d}$. Because $\mathbf{z}_\mathrm{d}$ and $\mathbf{z}_\mathrm{y}$ are independent, the model should learn representations $\mathbf{z}_\mathrm{y}$ that are domain invariant. The generative model is

$$p(\mathbf{x}, \mathbf{z}_\mathrm{d}, \mathbf{z}_\mathrm{y}, \mathbf{z}_\mathrm{x}, \mathbf{y}, \mathbf{d}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_\mathrm{d}, \mathbf{z}_\mathrm{y}, \mathbf{z}_\mathrm{x})p_{\boldsymbol{\theta}_\mathrm{d}}(\mathbf{z}_\mathrm{d}|\mathbf{d})p_{\boldsymbol{\theta}_\mathrm{y}}(\mathbf{z}_\mathrm{y}|\mathbf{y})p(\mathbf{z}_\mathrm{x})p(\mathbf{y})p(\mathbf{d}) \tag{3.25}$$

A graph of this model is shown in Figure 3.3b. $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_\mathrm{d}, \mathbf{z}_\mathrm{y}, \mathbf{z}_\mathrm{x})$ is the reconstruction decoder, and encoders $q_{\boldsymbol{\phi}_\mathrm{d}}(\mathbf{z}_\mathrm{d}|\mathbf{x})$, $q_{\boldsymbol{\phi}_\mathrm{y}}(\mathbf{z}_\mathrm{y}|\mathbf{x})$, $q_{\boldsymbol{\phi}_\mathrm{x}}(\mathbf{z}_\mathrm{x}|\mathbf{x})$ are introduced as variational



(a) (b)

Figure 3.3: a) Generative model for the Variational Fair Autoencoder. b) Generative model for the Domain Invariant Variational Autoencoder.

approximations to the posterior distributions. In addition the model learns the parameters of the conditional priors $p_{\boldsymbol{\theta}_\mathrm{d}}(\mathbf{z}_\mathrm{d}|\mathbf{d})$ and $p_{\boldsymbol{\theta}_\mathrm{y}}(\mathbf{z}_\mathrm{y}|\mathbf{z})$, and introduces classifiers $q_{\boldsymbol{\omega}_\mathrm{d}}(\mathbf{d}|\mathbf{z}_\mathrm{d})$ and $q_{\boldsymbol{\omega}_\mathrm{y}}(\mathbf{y}|\mathbf{z}_\mathrm{y})$.

The resulting lower bound objective for learning the parameters $\boldsymbol{\theta}_\mathcal{L} = \{\boldsymbol{\theta}, \boldsymbol{\theta}_\mathrm{d}, \boldsymbol{\theta}_\mathrm{y}, \boldsymbol{\phi}_\mathrm{d}, \boldsymbol{\phi}_\mathrm{y}, \boldsymbol{\phi}_\mathrm{x}\}$ is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_\mathcal{L}; \mathbb{X}, \beta) = \mathbb{E}_{(\mathbf{d},\mathbf{x},\mathbf{y})\sim\hat{p}_\mathbb{X}}[&\mathbb{E}_{\mathbf{z}_\mathrm{d}\sim q_{\boldsymbol{\phi}_\mathrm{d}}(\mathbf{z}_\mathrm{d}|\mathbf{x}),\mathbf{z}_\mathrm{y}\sim q_{\boldsymbol{\phi}_\mathrm{y}}(\mathbf{z}_\mathrm{y}|\mathbf{x}),\mathbf{z}_\mathrm{x}\sim q_{\boldsymbol{\phi}_\mathrm{x}}(\mathbf{z}_\mathrm{x}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_\mathrm{d}, \mathbf{z}_\mathrm{y}, \mathbf{z}_\mathrm{x})] \\ &- \beta D_\mathrm{KL}(q_{\boldsymbol{\phi}_\mathrm{d}}(\mathbf{z}_\mathrm{d}|\mathbf{x})\|p_{\boldsymbol{\theta}_\mathrm{d}}(\mathbf{z}_\mathrm{d}|\mathbf{d})) \\ &- \beta D_\mathrm{KL}(q_{\boldsymbol{\phi}_\mathrm{y}}(\mathbf{z}_\mathrm{y}|\mathbf{x})\|p_{\boldsymbol{\theta}_\mathrm{y}}(\mathbf{z}_\mathrm{y}|\mathbf{y})) \\ &- \beta D_\mathrm{KL}(q_{\boldsymbol{\phi}_\mathrm{x}}(\mathbf{z}_\mathrm{x}|\mathbf{x})\|p(\mathbf{z}_\mathrm{x}))]\end{aligned} \tag{3.26}$$

The hyperparameter β is introduced to control the capacity of the $\mathbf{z}$-spaces, analogously to the β introduced in the β-VAE framework [Higgins et al., 2017].

The full objective with additional objectives on $\boldsymbol{\theta}_\mathcal{F} = \{\boldsymbol{\omega}_\mathrm{d}, \boldsymbol{\omega}_\mathrm{y}\}$ is:

$$\begin{aligned}\mathcal{F}(\boldsymbol{\theta}_\mathcal{L}, \boldsymbol{\theta}_\mathcal{F}; \mathbb{X}, \beta, \alpha_\mathrm{d}, \alpha_\mathrm{y}) =& \mathcal{L}(\boldsymbol{\theta}_\mathcal{L}; \mathbb{X}, \beta) \\ &+ \alpha_\mathrm{d}\mathbb{E}_{(\mathbf{d},\mathbf{x})\sim\hat{p}_\mathbb{X}}[\mathbb{E}_{\mathbf{z}_\mathrm{d}\sim q_{\boldsymbol{\phi}_\mathrm{d}}(\mathbf{z}_\mathrm{d}|\mathbf{x})}[\log q_{\boldsymbol{\omega}_\mathrm{d}}(\mathbf{d}|\mathbf{z}_\mathrm{d})]] \\ &+ \alpha_\mathrm{y}\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\hat{p}_\mathbb{X}}[\mathbb{E}_{\mathbf{z}_\mathrm{y}\sim q_{\boldsymbol{\phi}_\mathrm{y}}(\mathbf{z}_\mathrm{y}|\mathbf{x})}[\log q_{\boldsymbol{\omega}_\mathrm{y}}(\mathbf{y}|\mathbf{z}_\mathrm{y})]]\end{aligned} \tag{3.27}$$

Here, additional hyperparameters $\alpha_\mathrm{d}, \alpha_\mathrm{y}$ are introduced to control the contribution from the classification objectives. The objective $\mathcal{F}$ may also be further extended to allow learning from unsupervised data examples $(\mathbf{d}, \mathbf{x})$.

Kulkarni et al. [2015] presents a different approach to utilizing supervision to encourage disentangling with the Deep Convolution Inverse Graphics Network (DC-IGN). The model consists of a probabilistic encoder and decoder following the VAE framework, implemented as convolutional neural networks.

The factors attempted captured and disentangled in the representation is identified before training and assigned a location in $\mathbf{z}$. This way, a subset $\{z_i\}_{i=1}^{k}$ of $\mathbf{z}$ is reserved for these identified disentangled features, while the remaining $\{z_i\}_{i=k+1}^{n}$, denoted $\mathbf{z}_{k+1:n}$, are left to capture any remaining variation in the data.

The dataset used to train the model is divided in mini-batches such that data in a batch only differ in one of these features $z_1, z_2, ..., z_k$, as well as batches where data are similar in features $z_1, z_2, ..., z_k$ but different in other aspects corresponding to feature-group $\mathbf{z}_{k+1:n}$.

During training, at each step one of $z_1, z_2, ..., z_k, \mathbf{z}_{k+1:n}$ is chosen, along with a mini-batch differing in the relevant feature or feature-group. For a z_i, a corresponding mini-batch is used to train the model to keep the relevant information in this feature, which is done by calculating an average over the examples for the remaining values of $\mathbf{z}$ and using the average values along with the unchanged value of z_i as input to the decoder. The gradients for the averaged variables of $\mathbf{z}$ are set to be the difference from the mean over the mini-batch examples, while the gradient of z_i is left unchanged.

Chapter 4

Disentanglement Evaluation

4.1 Introduction

Chapter 3 introduced the concept of disentangled representations, along with several models that attempt to output disentangled representations $r(\boldsymbol{x})$ given input $\boldsymbol{x}$. When a model is trained on a dataset $\mathbb{X}$, it is necessary to be able to provide evidence to whether it has accomplished the goal of generating representations that are disentangled. Furthermore, being able to establish how good a representation is in terms of disentanglement allow for comparison of and selection between different models. This chapter discusses some techniques used in order to evaluate disentanglement.

Several evaluation techniques have been explored in the disentanglement literature. Both qualitative and quantitative methods are used in order to evaluate the achieved level of disentanglement of a model. Section 4.2 presents some examples of qualitative techniques commonly used, and Section 4.3 focuses on quantitative evaluation, presenting a set of disentanglement metrics developed in order to be able to assign a score on the level of disentanglement accomplished by a given model.

Models that create representations include the probabilistic encoders of the different VAE-based models described in Sections 3.3, 3.4 and 3.5, as well as the q component of the InfoGAN described in Section 3.4. Generally, a trained model considered for evaluation is in this chapter referred to as encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Since the encoder outputs a distribution, the representation is not unambiguously defined given input $\boldsymbol{x}$. When discussing evaluation methods in this chapter, the representation is assumed defined as the mean of the encoder distribution, denoted $\hat{\boldsymbol{z}}$, if nothing else is specified. That is, $r(\boldsymbol{x}) = \hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\mathbf{z}]$. When $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\mathbf{x}))$, the representations therefore become $\hat{\boldsymbol{z}} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})$. An alternative definition is to sample a representation $r(\boldsymbol{x})$ as $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x} = \boldsymbol{x})$.

4.2 Qualitative Evaluation

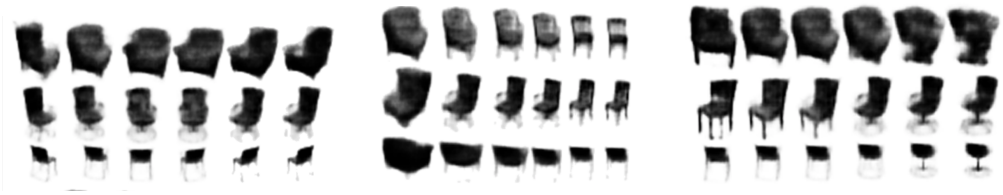
In order to perform qualitative evaluation of a model, some type of descriptive data must be collected by observing the model behaviour. In this context, observations are typically based on examining how input and output of a model is related trough

fixing some values and varying others. The data is analysed by looking for patterns in these observations. The results are typically presented organising input-output pairs according to any detected patterns, hopefully relating meaningful patterns in input to meaningful patterns in output.

A commonly used qualitative technique for evaluating disentanglement is based on generating samples from encoder representations using the corresponding decoder, while traversing dimensions of the representation to visualize the effects in decoder output resulting form changing the value of just one dimension in $\mathbf{z}$-space at a time. Given encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ trained on dataset $\mathbb{X}$, the procedure is as follows:

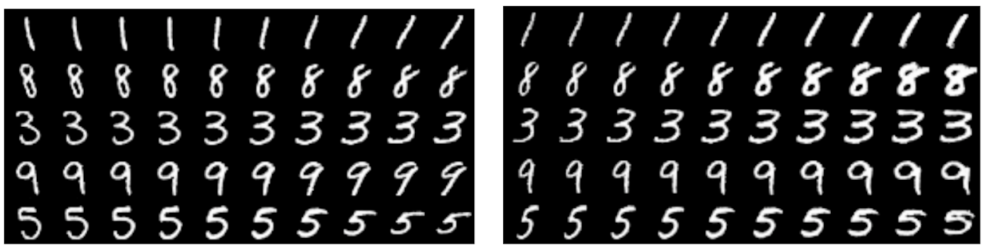
1. Start with a data sample $\boldsymbol{x} \in \mathbb{X}$
2. Use $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to obtain a representation $\hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x})}[\mathbf{z}]$
3. Decide on a dimension l of $\hat{\boldsymbol{z}}$ to traverse. Keep $\hat{\boldsymbol{z}}_{\setminus l}$ fixed.
4. For a range of values of $\hat{z}_l$, use decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ to generate reconstructions $\hat{\boldsymbol{x}} = \mathbb{E}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}=\hat{\boldsymbol{z}})}[\mathbf{x}]$ of original $\boldsymbol{x}$ using different values of $\hat{z}_l$ along with the fixed $\hat{\boldsymbol{z}}_{\setminus l}$.
5. Organise reconstructions $\hat{\boldsymbol{x}}$ in line such that the corresponding value of $\hat{z}_l$ increases or decreases along the line, to observe any changes in $\hat{\boldsymbol{x}}$ resulting from changing only the value of $\hat{z}_l$.

Steps 3.-5. are usually repeated for a number of dimensions of $\hat{\boldsymbol{z}}$ for each $\boldsymbol{x}$. Some examples of the dimension traversal procedure are shown in Figures 4.1 and 4.2.



(a) (b) (c)

Figure 4.1: Figures from [Higgins et al., 2017], where a β-VAE model is trained on the 3D chairs dataset. a), b) and c) show the results of traversing three different dimensions in $\mathbf{z}$-space, using the β-VAE decoder to generate reconstructions. The rows of each figure show the same traversal on different choices of $\boldsymbol{x}$. The dimensions traversed appear to correspond to a) rotation, b) width, and c) leg style.



(a) (b)

Figure 4.2: Figures from [Chen et al., 2016], where an InfoGAN model is trained on the MNIST dataset. a) and b) show the results of traversing two different dimensions in $\mathbf{z}$-space, using the InfoGAN generator to generate reconstructions. The rows of a figure show the same traversal on different choices of $\boldsymbol{x}$. The dimensions traversed appear to correspond to a) rotation, and b) width.

A slightly different example of qualitative evaluation is presented in [Ilse et al., 2019], applied in order to evaluate their DIVA model, presented in Section 3.5. This VAE-based model separates its $\mathbf{z}$-space into three disjoint subspaces, with each subspace being trained to store different information about input $\mathbf{x}$ using a semi-supervised approach. Subspaces are $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$, and the evaluation is done by regulating which subspaces are active, i.e. fed to the decoder, when reconstructing $\mathbf{x}$. Deactivating a subspace is done by setting all its values to 0. For a $\boldsymbol{x} \in \mathbb{X}$, its representation $\boldsymbol{z} = \{\boldsymbol{z}_x, \boldsymbol{z}_y, \boldsymbol{z}_d\}$ is generated using encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Then different reconstructions are created for each $\mathbf{x}$ using decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ on different subsets of $\{\boldsymbol{z}_x, \boldsymbol{z}_y, \boldsymbol{z}_d\}$. The example from [Ilse et al., 2019] is shown in Figure 4.3.

While these example illustrate that qualitative techniques do provide ways of evaluating disentanglement, they are not in general well suited for model comparison and selection. In order to be able to reliably compare the level of disentanglement of representations from different models, access to quantitative metrics giving a numerical score to any given model is considered more useful than using such qualitative evaluations.

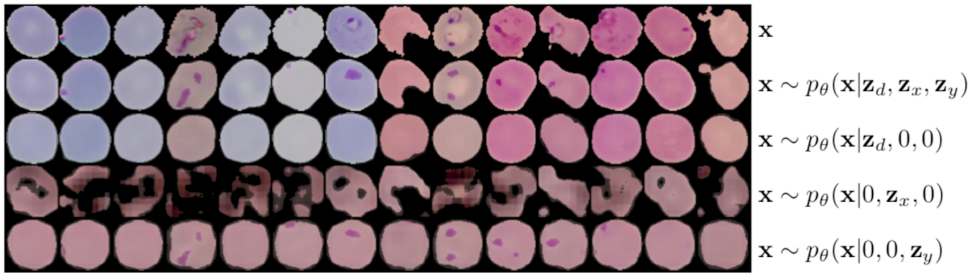


Figure 4.3: Figure from [Ilse et al., 2019], where a DIVA model is trained on a dataset of images of blood cells [Rajaraman et al., 2018], with some cells infected with malaria and others not. The model is trained using targets $y \in \{\text{infected}, \text{uninfected}\}$ and $d \in \{\text{patientID}_m\}_{m=1}^{M}$ for M different patient IDs. Parasite presence is identified in an image by dark pink marks inside the cell, but the colour of the cell itself may vary between patients. This is seen in the first line of the figure, which shows a set of different cell images $\boldsymbol{x}$. The remaining lines show reconstructions of these $\boldsymbol{x}$ using either all subspaces (line 2), or subspace $\boldsymbol{z}_d$ (line 3), $\boldsymbol{z}_x$ (line 4) or $\boldsymbol{z}_y$ (line 5) only. Thus each column shows the difference in reconstruction for a single $\boldsymbol{x}$ for different subsets of active $\boldsymbol{z}$ subspaces. From line 3 it appears that $\mathbf{z}_d$ contributes information about cell colour, and from line 5 that $\mathbf{z}_y$ contributes information about whether a malaria parasite is present or not, indicating that actual model behaviour is consistent with intended behaviour.

4.3 Quantitative evaluation

Quantitative evaluation of disentanglement refers to methods that assign a numerical score to the encoder model. Methods that provide such scores are called disentanglement metrics.

The metrics in this section are discussed in terms of how their evaluation of the model representations correspond to how well the representations meet the disentanglement criteria introduced in Section 3.3, summarised in Table 4.1.

Disentanglement metrics can further be divided into supervised and unsupervised metrics. Supervised here means that the metrics require knowledge about the ground truth generative factors in order to perform calculations. Unsupervised metrics are calculated directly on the representations created by a model, without access to information about the ground truth factors. Supervised metrics are discussed in 4.3.1 and unsupervised metrics in 4.3.2.

4.3.1 Supervised metrics

In order to calculate the supervised metrics, knowledge about the ground truth generative factors must be available, and a representation of this knowledge must be defined.

Criteria	Summary
Modularity	Each dimension $\hat{z}_l$ in $\hat{\mathbf{z}}$ keeps information about at most one ground truth generative factor
Compactness	Information about a ground truth generative factor is kept in $\hat{\mathbf{z}}$ by at most one dimension $\hat{z}_l$
Explicitness (non-linear)	The values of all ground truth generative factors can be fully recovered from $\hat{\mathbf{z}}$
Explicitness (linear)	Non-linear explicitness with the added requirement that the values of the ground truth generative factors are recovered from $\hat{\mathbf{z}}$ by a linear mapping

Table 4.1: Summary of the three disentanglement criteria, with two versions of explicitness, introduced in Section 3.3.

Usually, K generative factors are identified under the assumption of independence, with the requirement that each factor can be represented by a one dimensional random variable. The factors are then jointly represented by a vector $\mathbf{v} \in \mathbb{R}^K$, where the element at index k correspond to factor k. Each variable v_k may be continuous or discrete depending on what is most appropriate, such that $\mathbf{v}$ can be continuous, discrete or mixed. Suitable ranges of possible values for the variables to take should also be defined, such that the full distribution $p^*(\mathbf{v})$ is available.

Synthetic datasets are most often used, where the factors $\mathbf{v}$ and their distribution $p^*(\mathbf{v})$ are designed first. Additionally, the generative process of the data $\mathbf{x}$ given these $\mathbf{v}$ must be defined, denoted $p^*(\mathbf{x}|\mathbf{v})$, from which the dataset is then created by drawing samples $\boldsymbol{v} \sim p^*(\mathbf{v})$ and then $\boldsymbol{x} \sim p^*(\mathbf{x}|\mathbf{v} = \boldsymbol{v})$.

It may also be possible in some cases to provide labels of $\mathbf{v}$ for an existing dataset $\mathbb{X}$. Such labeling can be considered as approximate sampling from $p^*(\mathbf{v}|\mathbf{x} = \boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{X}$ in order to create a dataset $\mathbb{X}^* = \{(\boldsymbol{x}^{(i)}, \boldsymbol{v}^{(i)})\}_{i=1}^n$ that can be used as a basis for the metric calculations. Typically in this case labels will only be partially available, and metrics will be used in order to provide partial disentanglement evaluation.

Supervised metrics evaluate disentanglement with respect to the ground truth factors by considering factor representations $\mathbf{v}$ as weak targets that the model representations $\hat{\mathbf{z}}$ are evaluated against, where $\hat{\mathbf{z}}$ is not expected to recover the exact values of $\mathbf{v}$, but rather recover the same information within a disentangled structure as defined by the criteria in Table 4.1. Some restrictions on the recovery of $\mathbf{v}$ in $\hat{\mathbf{z}}$ are determined by differences between $p^*(\mathbf{v})$ and the model assumption about $p(\mathbf{z})$, typically $p(\mathbf{z}) = N(\mathbf{z}; \mathbf{0}, I)$. For $\hat{\mathbf{z}} \in \mathbb{R}^L$, L is chosen such that $L \geq K$ and often $L > K$ [Zaidi et al., 2021]. Figure 4.4 illustrates the relationship between $\mathbf{v}$ and $\hat{\mathbf{z}}$ given $p^*(\mathbf{x}|\mathbf{v})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$.

In order to return a useful disentanglement score on disentanglement in $\hat{\mathbf{z}}$ relative to $\mathbf{v}$, Zaidi et al. [2021] considers a set of general properties that a metric should ideally posses. First, the scores produced by the metric should be calibrated to lie in the normalized range $[0, 1]$. Given a model that produce perfectly disentangled $\hat{\mathbf{z}}$,

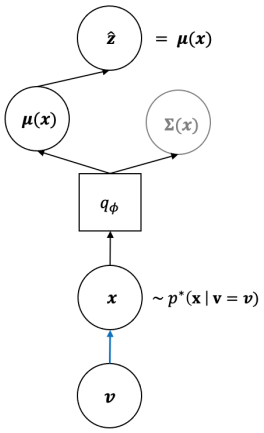


Figure 4.4: From ground truth factors $\mathbf{v}$ to disentangled representation $\hat{\mathbf{z}}$, when $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\Sigma}_\phi(\mathbf{x}))$

the metric should give a maximum score of 1, while a model that produce representations of no disentanglement should get a minimum score of 0. In between these extremes, the assigned score of a model should decrease close to linearly relative to the level of disentanglement obtained by the model, to allow for reliable comparison of models using scores [Zaidi et al., 2021].

Ideally, metrics should also have few hyper parameters, impose few assumptions about the mapping from $\hat{\mathbf{z}}$ to $\mathbf{v}$ and not be sensitive to noise in $\hat{\mathbf{z}}$ [Zaidi et al., 2021]. In the case where $\mathbf{v}$ can only be designed to cover a proper subset of the generative factors assumed to have generated the data, any remaining factors not in $\mathbf{v}$ should still be encoded in $\hat{\mathbf{z}}$, and could appear as noise when calculating a score on a representation relative to $\mathbf{v}$.

In their review of metrics, Zaidi et al. [2021] introduce a taxonomy classifying the metrics into three different groups. These are referred to as Intervention-based, Information-based and Predictor-based metrics, and the metrics discussed here are introduces according to this grouping.

Intervention-based metrics

Intervention-based metrics are based on comparison of model representations $\mathbf{z}$ and ground truth factors $\mathbf{v}$ through creating sets of data points from fixed factor values [Zaidi et al., 2021].

One supervised disentanglement metric classified as an intervention-based metric is introduced in [Higgins et al., 2017], here referred to as the Z-diff metric, in accordance with [Zaidi et al., 2021]. In order to calculate a score using the Z-diff metric, a set of representations $\hat{\mathbf{z}}$ are created from a set of ground truth factors $\mathbf{v}$, with the value of one factor v_k kept fixed. That is, for a fixed value v_k of v_k, sample $\boldsymbol{x} \sim p^*(\mathbf{x}|\mathrm{v}_k = v_k)$ and then generate $\hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}=\boldsymbol{x})}[\mathbf{z}]$. Difference vectors $\hat{\mathbf{z}}_{\text{diff}} = |\hat{\mathbf{z}}' - \hat{\mathbf{z}}''|$ are created from pairs of representations $(\hat{\mathbf{z}}', \hat{\mathbf{z}}'')$ generated from the same value of the fixed factor, such that if the representations are in fact

Algorithm 1 Z-diff Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, true generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$
$\mathbb{B} \leftarrow \{\}$
for $b = 1$ **to** B **do**
  Choose a factor $k \in \{1, ..., K\}$
  for $c = 1$ **to** C **do**
    Fix a value $v_k^{(c)}$ of the factor representation v_k
    Sample values $\boldsymbol{v}'^{(c)}_{\setminus k}$ and $\boldsymbol{v}''^{(c)}_{\setminus k}$ at random.
    Sample datapoints $\boldsymbol{x}'^{(c)} \sim p^*(\mathbf{x}|\mathbf{v}_{\setminus k} = \boldsymbol{v}'^{(c)}_{\setminus k}, \mathrm{v}_k = v_k^{(c)})$ and
    $\boldsymbol{x}''^{(c)} \sim p^*(\mathbf{x}|\mathbf{v}_{\setminus k} = \boldsymbol{v}''^{(c)}_{\setminus k}, \mathrm{v}_k = v_k^{(c)})$
    Infer $\hat{\boldsymbol{z}}'^{(c)} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x}'^{(c)})}[\mathbf{z}]$ and $\hat{\boldsymbol{z}}''^{(c)} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x}''^{(c)})}[\mathbf{z}]$
    Compute the difference $\hat{\boldsymbol{z}}^{(c)}_{\text{diff}} = |\hat{\boldsymbol{z}}'^{(c)} - \hat{\boldsymbol{z}}''^{(c)}|$
  end for
  Compute average $\hat{\boldsymbol{z}}^{(b)}_{\text{avg}} = \frac{1}{C}\sum_{c=1}^{C} \hat{\boldsymbol{z}}^{(c)}_{\text{diff}}$
  $\mathbb{B} \leftarrow \mathbb{B} \cup \{(\hat{\boldsymbol{z}}^{(b)}_{\text{avg}}, k)\}$
end for
Train a linear classifier on $\mathbb{B}$
Return DisentanglementScore $\leftarrow$ Normalized accuracy of linear classifier

disentangled, one would expect values close to 0 in the dimensions of $\hat{\mathbf{z}}_{\text{diff}}$ that capture the fixed factor. Averages over batches of $\hat{\mathbf{z}}_{\text{diff}}$'s generated from the same fixed factor are used as input to a linear classifier trained to output the index of the factor that was kept fixed. The accuracy of the linear classifier acts as the assigned disentanglement score of the model encoder in question. Zaidi et al. [2021] suggests using the fact that a completely random classifier assigning input to K classes will have accuracy of $1/K$ on average to normalize the scores assigned by the Z-diff metric. The details of the metric calculations are shown in Algorithm 1. The process is also shown in Figure 4.5a.

A $\hat{\mathbf{z}}_{\text{diff}}$ would contain 0 in dimensions where the original representations $(\hat{\mathbf{z}}', \hat{\mathbf{z}}'')$ are equal. Equal value of a dimension implies the value is caused by the fixed generative factor value without interference from other factors. This indicates that a good or bad score should correspond to high or low modularity. Kim and Mnih [2019] points out a failure mode of the Z-diff metric, in that it will report 100% accuracy even if only $K - 1$ out of K factors are disentangled according to modularity in the representation, where the linear classifier will be able to map $\hat{\mathbf{z}}_{\text{diff}}$ averages without small value patterns to the remaining factor, despite it not being disentangled.

Kim and Mnih [2019] introduce a metric that attempts to improve on the Z-diff metric. This metric is referred to as Z-min Variance. For a set of representations created keeping one factor fixed, it replaces the average of the difference vectors with the empirical variance of the original representations. Then the index of the representation dimension that has the lowest variance is recorded, and a majority-vote classifier is used to predict generative factor from representation dimension.

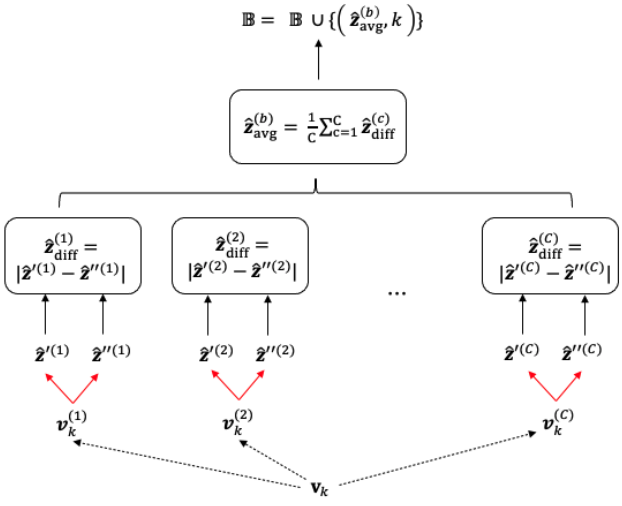
Algorithm 2 Z-min Variance Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, ground truth generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$,
$\boldsymbol{s}_g \leftarrow$ Empirical standard deviation over full data (or large random sample)
$\mathbb{B} \leftarrow \{\}$
for $b = 1$ **to** B **do**
  Choose a factor $k \in \mathbb{G}$
  Fix a value $v_k^{(b)}$ of the factor representation v_k
  for $c = 1$ **to** C **do**
    Sample values $\boldsymbol{v}_{\setminus k}^{(c)}$ at random.
    Sample datapoint $\boldsymbol{x}^{(c)} \sim p^*(\mathbf{x}|\mathbf{v}_{\setminus k} = \boldsymbol{v}_{\setminus k}^{(c)}, \mathrm{v}_k = v_k^{(b)})$
    Infer $\hat{\boldsymbol{z}}^{(c)} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x}^{(c)})}[\mathbf{z}]$
    Obtain normalised $\bar{\boldsymbol{z}}^{(c)}$ by dividing $\hat{\boldsymbol{z}}^{(c)}$ by $\boldsymbol{s}_g$, $\bar{\boldsymbol{z}}^{(c)} = \frac{\hat{\boldsymbol{z}}^{(c)}}{\boldsymbol{s}_g}$
  end for
  $\bar{\bar{\boldsymbol{z}}} \leftarrow \frac{\sum_c \bar{\boldsymbol{z}}^{(c)}}{C}$
  $\boldsymbol{s}^2 \leftarrow$ Empirical variance in each dimension of the $\bar{\boldsymbol{z}}^{(c)}$,
    $\mathrm{diag}(\frac{1}{C-1}\sum_c(\bar{\boldsymbol{z}}^{(c)} - \bar{\bar{\boldsymbol{z}}})(\bar{\boldsymbol{z}}^{(c)} - \bar{\bar{\boldsymbol{z}}})^T)$
  $l^* \leftarrow \mathrm{argmin}_l\, s_l^2$
  $\mathbb{B} \leftarrow \mathbb{B} \cup \{(l^*, k)\}$
end for
Train a majority-vote classifier on $\mathbb{B}$.
Return DisentanglementScore $\leftarrow$ Normalized accuracy of majority-vote classifier

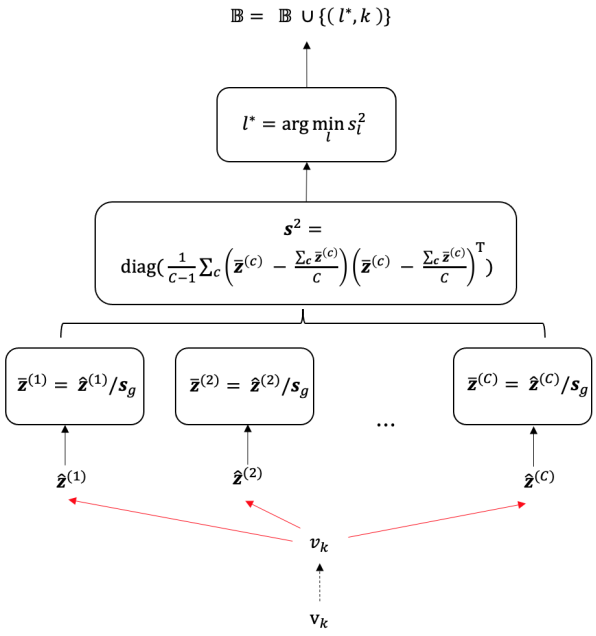
The accuracy of the classifier is used as the disentanglement score. The score can be normalized similarly to Z-diff. Details are presented in Algorithm 2, as well as in Figure 4.5b.

Predicting a factor correctly from variance of a dimension requires that there is a dimension that consistently shows low variance when controlled for that factor, and thus all factors must be present in the representation in order to get an optimal score, correcting for the failure mode mentioned for the Z-diff metric above. Using a majority-vote classifier reduces hyperparameter sensitivity compared to Z-diff metric [Kim and Mnih, 2019].

Similarly to the 0 valued dimensions of the difference vectors before, low variance of a dimension over a batch of representations indicates that that dimension corresponds to the fixed factor, and the more consistent this relationship is, the more modular the representation. Compactness is not measured by either Z-diff or Z-min Variance as the classifier will map vectors with any number of low value dimensions to the right factor index as long as they are consistent for the corresponding fixed factor. Explicitness is also not measured by these metrics, since whether values of $\mathbf{v}$ can be fully or only partially recovered from the detected dimensions of $\hat{\mathbf{z}}_{\text{diff}}$ does not impact the score, as the linear classifier is asked only to map dimensions of $\hat{\mathbf{z}}_{\text{diff}}$ to corresponding dimensions of $\mathbf{v}$, not considering values of either variable.



(a)



(b)

Figure 4.5: a) Illustration of the generation of dataset $\mathbb{B}$ in Algorithm 1, Z-diff metric. The red arrows indicates the process depicted in Figure 4.4 b) Illustration of the generation of dataset $\mathbb{B}$ in Algorithm 2, Z-min Variance metric. The red arrows correspond to Figure 4.4 and $\boldsymbol{s}_g$ is the empirical standard deviation of the dataset.

Z-diff and Z-min Variance do not require all generative factors to be present in $\mathbf{v}$ in order to produce a meaningful score, and can thus be modified to allow calculation of partial disentanglement based on a dataset $\mathbb{X}^*$, assuming the dataset is of sufficient size, with high variance in the labeled factors.

Zaidi et al. [2021] and Sepliarskaia et al. [2021] presents experiment results that show that while Z-diff and Z-min Variance are expected to give a low score when representations are not modular, there are failure modes for both metrics that lead them to assigning a good score to non-modular representations, see Figure 4.6.

(a) (b)

Figure 4.6: a) An example of a non-modular $\mathbf{v}$ - $\mathbf{z}$ relationship that will be assigned a perfect score by Z-diff and Z-min Variance [Sepliarskaia et al., 2021; Zaidi et al., 2021]. Lines connecting dimensions in $\mathbf{v}$ and $\mathbf{z}$ are numbered according to how much information is shared along the line, under the assumption of independence in both spaces. b) Graphs showing the relative variation one would expect in each $\mathbf{z}$ dimension when keeping each factor fixed in Z-diff and Z-min Variation calculatons, assuming a model that has learned the relationship in a). As shown in [Sepliarskaia et al., 2021; Zaidi et al., 2021], the Z-diff classifiers would still be able to map distinct combinations of lowest variance dimensions to the fixed index, and the Z-min Variance classifier will give a maximum score as long as the dimension in $\mathbf{z}$ with the lowest variance consistently maps to the fixed index.

Another intervention-based metric is presented in [Suter et al., 2019], where a causal framework for evaluating disentanglement is considered, starting from a generative process described by the graph $\mathbf{c} \rightarrow \mathbf{v} \rightarrow \mathbf{x}$, assumed to represent a causal model where factors in $\mathbf{v}$ are mutually independent given confounders $\mathbf{c}$. The disentanglement metric introduced is called Interventional robustness score (IRS).

With $\mathbb{L} = \{1, ..., L\}$ as the set of indices of elements in $\hat{\mathbf{z}}$ and $\mathbb{K} = \{1, ..., K\}$ as the set of indices of elements in $\mathbf{v}$, and with subsets $\mathbb{L}' \subseteq \mathbb{L}$ and $\mathbb{K}' \subseteq \mathbb{K}$, $\text{IRS}(\mathbb{L}'|\mathbb{K}', \mathbb{K}\backslash\mathbb{K}')$ is a general metric that measure how robust encodings in $\hat{\mathbf{z}}$ located at indices in $\mathbb{L}'$ is to interventions on factors in $\mathbf{v}$ at indices in $\mathbb{K}\backslash\mathbb{K}'$ when factors at indices in $\mathbb{K}'$ are kept fixed [Suter et al., 2019]. The metric is based on calculating differences after interventions for given values $\boldsymbol{v}_{\mathbb{K}'}$ for $\mathbf{v}_{\mathbb{K}'}$ and $\boldsymbol{v}_{\mathbb{K}\backslash\mathbb{K}'}$ for $\mathbf{v}_{\mathbb{K}\backslash\mathbb{K}'}$:

$$\begin{aligned} &\text{PIDA}(\mathbb{L}'|\boldsymbol{v}_{\mathbb{K}'}, \boldsymbol{v}_{\mathbb{K}\backslash\mathbb{K}'}) = \\ &\quad d(\mathbb{E}[\mathrm{z}_{\mathbb{L}'}|do(\mathbf{v}_{\mathbb{K}'} = \boldsymbol{v}_{\mathbb{K}'})],\ \mathbb{E}[\mathrm{z}_{\mathbb{L}'}|do(\mathbf{v}_{\mathbb{K}'} = \boldsymbol{v}_{\mathbb{K}'}, \mathbf{v}_{\mathbb{K}\backslash\mathbb{K}'} = \boldsymbol{v}_{\mathbb{K}\backslash\mathbb{K}'})]) \end{aligned} \tag{4.1}$$

where d is an appropriate distance function, e.g. the Euclidean distance.

The generative process considered in this chapter considers only generative factors $\mathbf{v}$ of absolute mutual independence. When there is no confounding, $p(\mathbf{x}|do(\mathbf{y} \leftarrow \boldsymbol{y})) = p(\mathbf{x}|\mathbf{y} = \boldsymbol{y})$ [Suter et al., 2019]. Additionally, $\text{IRS}(\{l\}|\{k\}, \mathbb{K}\backslash\{k\})$ is identified in [Suter et al., 2019] as a special case of IRS that measures disentanglement. Presented in the following are details of the disentanglement version assuming no confounding.

$\text{IRS}(\{l\}|\{k\}, \mathbb{K}\backslash\{k\})$ requires calculation of $\text{PIDA}(\{l\}|v_k, \boldsymbol{v}_{\backslash k})$ for a single dimension l of $\hat{\mathbf{z}}$ and for given values v_k of v_k and $\boldsymbol{v}_{\backslash k}$ of $\mathbf{v}_{\backslash k}$. Assuming no confounding, equation (4.1) now becomes:

$$\begin{aligned}\text{PIDA}(\{l\}|v_k, \boldsymbol{v}_{\backslash k}) = \\ d(\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\text{v}_k=v_k)}[\hat{\text{z}}_l]],\ \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\text{v}_k=v_k, \mathbf{v}_{\backslash k}=\boldsymbol{v}_{\backslash k})}[\hat{\text{z}}_l]])\end{aligned} \tag{4.2}$$

In order to assign a score over all values of v_k and $\mathbf{v}_{\backslash k}$, first the maximal PIDA distance caused by any value of $\mathbf{v}_{\backslash k}$ is found, called $\text{MPIDA}(\{l\}|v_k, \mathbb{K}\backslash\{k\})$, and then the expected value of MPIDA is taken over v_k:

$$\text{MPIDA}(\{l\}|v_k, \mathbb{K}\backslash\{k\} = \sup_{\boldsymbol{v}_{\backslash k}} \text{PIDA}(\{l\}|v_k, \boldsymbol{v}_{\backslash k}]) \tag{4.3}$$

$$\text{EMPIDA}(\{l\}|\{k\}, \mathbb{K}\backslash\{k\}) = \mathbb{E}_{\text{v}_k\sim p(\text{v}_k)}[\,\text{MPIDA}(\{l\}|v_k, \mathbb{K}\backslash\{k\})\,] \tag{4.4}$$

IRS is defined as:

$$\text{IRS}(\{l\}|\{k\}, \mathbb{K}\backslash\{k\}) = 1 - \frac{\text{EMPIDA}(\{l\}|\{k\}, \mathbb{K}\backslash\{k\})}{\text{EMPIDA}(\{l\}|\emptyset, \mathbb{K})} \tag{4.5}$$

From Equation (4.1) the general $\text{IRS}(\mathbb{L}'|\mathbb{K}', \mathbb{K}\backslash\mathbb{K}')$ is computed analogously [Suter et al., 2019].

The disentanglement score of $\hat{\text{z}}_l$ is taken as $\max_k \text{IRS}(\{l\}|\{k\}, \mathbb{K}\backslash\{k\})$, and an overall score is given by averaging over all $\hat{\text{z}}_l$. Details of the IRS calculation are summarised in Algorithm 3, which shows calculations in the case where factors are sampled to generate a dataset for which representations are obtained, and expectations $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{v}=\boldsymbol{v})}[\hat{\text{z}}_l]]$ are estimated by frequencies of co-occurring values in $\boldsymbol{v}$ and $\hat{\text{z}}_l$. This requires discretizing continuous factors before estimation.

$\text{IRS}(\{l\}|\{k\}, \mathbb{K}\backslash\{k\})$ gives a score on how robust dimension l of $\hat{\mathbf{z}}$ is to changes in factors $\mathbf{v}_{\backslash k}$ when factor v_k is kept fixed, and will be high if v_k has high influence on $\hat{\text{z}}_l$ and v_k is the only factor influencing the value of $\hat{\text{z}}_l$. Thus $\max_k \text{IRS}(\{l\}|\{k\}, \mathbb{K}\backslash\{k\})$ will be high if there is information about a single factor encoded in $\hat{\text{z}}_l$, and low if multiple factors or noise are encoded in $\hat{\text{z}}_l$. This corresponds to measuring modularity and explicitness under the assumption that all factors expected to be present in $\hat{\mathbf{z}}$ must be present in $\mathbf{v}$. Thus IRS can not be used to measure partial disentanglement.

Algorithm 3 IRS Disentanglement Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, true generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$
Sample a matrix V of N row vectors of generative factors $\boldsymbol{v}$
Sample a tensor $\boldsymbol{X}$ of data points $\boldsymbol{x}$ for V using $p^*(\mathbf{x}|\mathbf{v})$
Obtain representations $\hat{Z}$ of row vectors $\hat{\boldsymbol{z}}$ for $\boldsymbol{X}$ using $\hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x})}[\mathbf{z}]$
Discretize V
From V, get K sets $\mathbb{J}^{(k)}$ of all values present in V for each factor
for $l = 1$ **to** L **do**
  for $k = 1$ **to** K **do**
    for j **in** $\mathbb{J}^{(k)}$ **do**
      $\hat{Z}^{(j)} \leftarrow N_j$ rows of $\hat{Z}$ corresponding to rows of V with element at index k equaling realization j of v_k
      $m \leftarrow$ Estimate $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathrm{v}_k=v_k)}[\hat{\mathrm{z}}_l]]$ as $\dfrac{1}{N_j}\displaystyle\sum_{n=1}^{N_j} \hat{z}^{(j)}_{n,l}$
      $\mathrm{MPIDA}_j \leftarrow 0$
      From V, get $\mathbb{I}^{(\setminus k)}$, the set of all values present of $\mathbf{v}_{\setminus k}$
      for i **in** $\mathbb{I}^{(\setminus k)}$ **do**
        $Z^{(j,i)} \leftarrow N_{ji}$ rows of $\hat{Z}$ corresponding to rows of V with element at index k equaling realization j of v_k and the remaining elements equaling realization i of $\mathbf{v}_{\setminus k}$
        $m' \leftarrow$ Estimate $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathrm{v}_k=v_k,\mathbf{v}_{\setminus k}=\boldsymbol{v}_{\setminus k})}[\hat{\mathrm{z}}_l]]$ as $\dfrac{1}{N_{ji}}\displaystyle\sum_{n=1}^{N_{ji}} \hat{z}^{(j,i)}_{n,l}$
        $\mathrm{PIDA} \leftarrow d(m, m')$
        $\mathrm{MPIDA}_j \leftarrow \max(\mathrm{MPIDA}_j, \mathrm{PIDA})$
      end for
    end for
    $\mathrm{EMPIDA}_{l,k} \leftarrow \displaystyle\sum_{j=1}^{|\mathbb{J}^{(k)}|} \frac{N_j}{N} \mathrm{MPIDA}_j$
    $m_\emptyset \leftarrow$ Estimate $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\hat{\mathrm{z}}_l]]$ as $\dfrac{1}{N}\displaystyle\sum_{n=1}^{N} \hat{z}_{n,l}$
    From V, get $\mathbb{I}^{(K)}$, the set of all values present of $\mathbf{v}$
    $\mathrm{EMPIDA}_{l,\emptyset} \leftarrow 0$
    for i **in** $\mathbb{I}^{(K)}$ **do**
      $Z^{(i)} \leftarrow N_i$ rows of $\hat{Z}$ corresponding to rows of V with elements equaling realization i of $\mathbf{v}$
      $m' \leftarrow$ Estimate $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{v}=\boldsymbol{v})}[\hat{\mathrm{z}}_l]]$ as $\dfrac{1}{N_i}\displaystyle\sum_{n=1}^{N_i} \hat{z}^{(i)}_{n,l}$
      $\mathrm{PIDA}_\emptyset \leftarrow d(m_\emptyset, m')$
      $\mathrm{EMPIDA}_{l,\emptyset} \leftarrow \max(\mathrm{EMPIDA}_{l,\emptyset}, \mathrm{PIDA}_\emptyset)$
    end for
    $\mathrm{IRS}_{l,k} \leftarrow 1 - \dfrac{\mathrm{EMPIDA}_{l,k}}{\mathrm{EMPIDA}_{l,\emptyset}}$
  end for
end for
Return DisentanglementScore $\leftarrow \frac{1}{L}\sum_{l=1}^{L} \max_k \mathrm{IRS}_{l,k}$

Information-based metrics

Information-based metrics are based on using information theory techniques to evaluate the relationship between $\mathbf{v}$ and $\hat{\mathbf{z}}$ [Zaidi et al., 2021]. The Information-based metrics discussed here are based on calculating the mutual information between elements of $\mathbf{v}$ and $\hat{\mathbf{z}}$.

Chen et al. [2019] introduce a metric called Mutual Information Gap (MIG), which is calculated from the empirical mutual information $I(\mathrm{z}_l; \mathrm{v}_k)$ between each dimension l of the representation and each ground truth factor k. If z_l encodes a lot of the information in v_k, the mutual information between these variables will be high. In order to encourage axis-alignment, the metric uses the difference between the top two $I(\mathrm{z}_l; \mathrm{v}_k)$ for each k as the disentanglement score on factor k [Chen et al., 2019]. The differences is then averaged to get a joint score on the representation.

Chen et al. [2019] considers sampling representations $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x} = \boldsymbol{x})$ instead of using the expected value, and suggests estimating the mutual information as

$$I(\mathrm{z}_l; \mathrm{v}_k) = \mathbb{E}_{q(\mathrm{z}_l, \mathrm{v}_k)}[\log \sum_{\boldsymbol{x} \in \mathbb{X}_{\mathrm{v}_k}} q_{\boldsymbol{\phi}}(\mathrm{z}_l|\boldsymbol{x}) \hat{p}_{\mathbb{X}}(\boldsymbol{x}|\mathrm{v}_k)] + H(\mathrm{z}_l) \tag{4.6}$$

where $q(\mathrm{z}_l, \mathrm{v}_k) = \sum_{\boldsymbol{x}} p^*(\mathrm{v}_k) \hat{p}_{\mathbb{X}}(\boldsymbol{x}|\mathrm{v}_k) q_{\boldsymbol{\phi}}(\mathrm{z}_l|\boldsymbol{x})$ and $\mathbb{X}_{\mathrm{v}_k}$ is the support of $\hat{p}_{\mathbb{X}}(\boldsymbol{x}|\mathrm{v}_k)$ [Chen et al., 2019]. MIG normalizes the mutual information by dividing by $H(\mathrm{v}_k)$, using that $0 \leq I(\mathrm{z}_l; \mathrm{v}_k) \leq H(\mathrm{v}_k) = \mathbb{E}_{p(\mathrm{v}_k)}[-\log p(\mathrm{v}_k)]$ holds when v_k is discrete [Chen et al., 2019], which impose an assumption that $\mathbf{v}$ is discrete when using this metric.

Some of the work presented that consider the MIG metric takes a different approach at estimating the mutual information when calculating the metric [Locatello et al., 2019; Zaidi et al., 2021]. Defining model representations to be $\hat{\mathbf{z}} = \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\mathbf{z}]$, they discretize the dimensions $\hat{\mathbf{z}}$, such that for each dimension l, values are binned. When used in the case where $\mathbf{v}$ is continuous or mixed, $\mathbf{v}$ should also be discretized.

Given that $\hat{\mathrm{z}}_l$ and v_k are now two discrete random variables taking values in $\{\hat{z}_{l,1}, \hat{z}_{l,2}, ..., \hat{z}_{l,I}\}$ and $\{v_{k,1}, v_{k,2}, ..., v_{k,J}\}$ respectively, probabilities $q(\hat{z}_{l,i}, v_{k,j})$ $q(\hat{z}_{l,i})$ and $p(v_{k,j})$ can be estimated from a set of N samples as the number of samples where the values occur divided by N. Mutual information between $\hat{\mathrm{z}}_l$ and v_k is then calculated as:

$$I(\hat{\mathrm{z}}_l; \mathrm{v}_k) = \sum_{i=1}^{I} \sum_{j=1}^{J} q(\hat{z}_{l,i}, v_{k,j}) \log \left(\frac{q(\hat{z}_{l,i}, v_{k,j})}{q(\hat{z}_{l,i}) p(v_{k,j})} \right) \tag{4.7}$$

Details of the computation of the MIG metric using discretization of $\hat{\mathbf{z}}$ is presented in Algorithm 4.

With $\hat{\mathrm{z}}_{l^{(k)}}$ being the dimension of $\hat{\mathbf{z}}$ with the highest mutual information with factor k and $\hat{\mathrm{z}}_{l'^{(k)}}$ the dimension with the second highest mutual information with factor k, the difference $I(\hat{z}_{l^{(k)}}; v_k) - I(\hat{z}_{l'^{(k)}}; v_k)$ is high when $\hat{\mathrm{z}}_{l^{(k)}}$ has a lot of information about k, while $\hat{\mathrm{z}}_{l'^{(k)}}$ has very little information about k. Thus in order for all k differences to be high, ensuring a high score, the representation $\hat{\mathbf{z}}$ must be compact, and if the representation is not compact, the score will be lower. Modularity is

Algorithm 4 MIG Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, ground truth generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$
Sample a matrix V of N row vectors of generative factors $\boldsymbol{v}$
Sample a tensor $\boldsymbol{X}$ of data points $\boldsymbol{x}$ for V using $p^*(\mathbf{x}|\mathbf{v})$
Obtain representations $\hat{Z}$ of row vectors $\hat{\boldsymbol{z}}$ for $\boldsymbol{X}$ using $\hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x})}[\mathbf{z}]$
Discretize $\hat{Z}$
for $k = 1$ **to** K **do**
  for $l = 1$ **to** L **do**
    Calculate $I(\hat{\mathrm{z}}_l; \mathrm{v}_k)$ from $\hat{\boldsymbol{z}}_l$ and $\boldsymbol{v}_k$ using Equation (4.7)
  end for
  Calculate $H(\mathrm{v}_k)$ from $\boldsymbol{v}_k$
  $l^{(k)} \leftarrow \mathrm{argmax}_l\, I(\hat{\mathrm{z}}_l; \mathrm{v}_k)$
  $l'^{(k)} \leftarrow \mathrm{argmax}_{l\neq l^{(k)}}\, I(\hat{\mathrm{z}}_l; \mathrm{v}_k)$
end for
Return DisentanglementScore $\leftarrow \frac{1}{K}\sum_k \frac{1}{H(v_k)}(I(\hat{\mathrm{z}}_{l^{(k)}}; \mathrm{v}_k) - I(\hat{\mathrm{z}}_{l'^{(k)}}; \mathrm{v}_k))$

required in order for any $I(\hat{z}_l; v_k)$ to be maximal, however low modularity combined with high compactness will give $I(\hat{z}_{l^{(k)}}; v_k) - I(\hat{z}_{l'^{(k)}}; v_k) = I(\hat{z}_{l^{(k)}}; v_k) - 0 > 0$ as long as some information about v_k is captured compactly in $\hat{\mathbf{z}}$. If all $I(\hat{z}_{l^{(k)}}; v_k) = H(v_k)$, then all information in $\mathbf{v}$ is encoded in $\hat{\mathbf{z}}$, and so the MIG score also measures non-linear explicitness. Discretization of $\hat{\mathbf{z}}$ is shown in in [Zaidi et al., 2021] to cause the MIG metric to score lower on highly non-linear $\hat{\mathbf{z}} - \mathbf{v}$ mappings despite high compactness and explicitness, due to uneven number of samples in value bins.

As the metric only considers differences $I(\hat{z}_{l^{(k)}}; v_k) - I(\hat{z}_{l'^{(k)}}; v_k)$ for each factor in $\mathbf{v}$, it can be used to measure partial disentanglement, and Algorithm 4 can be modified to take as input a labeled dataset $\mathbb{X}^*$.

Another mutual information-based metric is introduced in [Ridgeway and Mozer, 2018]. This metric is referred to as the Modularity metric, and is explicitly designed to measure modularity only. The metric is based on estimating the mutual information between all pairs of $\hat{\mathrm{z}}_l$ and v_k. From all $I(\hat{\mathrm{z}}_l, \mathrm{v}_k)$ a $L \times K$ matrix T is created, such that in each row l, $t_{l,k}$ contains $I(\hat{\mathrm{z}}_l, \mathrm{v}_k)$ if it is the maximum mutual information that any factor has with $\hat{\mathrm{z}}_l$, and 0 otherwise. T is used as a perfectly modular target to compare the actual $I(\hat{\mathrm{z}}_l, \mathrm{v}_k)$'s against, computing a deviation from the target as $\frac{1}{K-1}\sum_{k=1}^{K}\left(\frac{I(\hat{\mathrm{z}}_l,\mathrm{v}_k)-t_{l,k}}{\max_k I(\hat{\mathrm{z}}_l,\mathrm{v}_k)}\right)^2$. This deviation will be 0 if $\hat{\mathbf{z}}$ is in fact perfectly modular, and it will be 1 if dimension l of $\hat{\mathbf{z}}$ has equal mutual information with all k factors [Ridgeway and Mozer, 2018]. The reported overall score is given by the average $\frac{1}{L}\sum_{l=1}^{L}(1 - \frac{1}{K-1}\sum_{k=1}^{K}\left(\frac{I(\hat{\mathrm{z}}_l,\mathrm{v}_k)-t_{l,k}}{\max_k I(\hat{\mathrm{z}}_l,\mathrm{v}_k)}\right)^2)$ over all dimensions of $\hat{\mathbf{z}}$. Calculation details of the Modularity metric are summarised in Algorithm 5, where $\mathbf{v}$ is assumed to be discrete and $\hat{\mathbf{z}}$ is discretized.

Zaidi et al. [2021] show that while the Modualrity metric is intended to assign scores according to level of modularity, normalizing by $\max_k I(\hat{\mathrm{z}}_l, \mathrm{v}_k)$ can lead to assigning good scores to non-modular representations. The modular target T assumes one factor in $\mathbf{v}$ should always be identified as encoded in each dimension of

$\hat{\mathbf{z}}$, and the metric will therefore not be useful on partially labeled data.

Algorithm 5 Modularity Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, ground truth generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$
Sample a matrix V of N row vectors of generative factors $\boldsymbol{v}$
Sample a tensor $\boldsymbol{X}$ of data points $\boldsymbol{x}$ for V using $p^*(\mathbf{x}|\mathbf{v})$
Obtain representation matrix $\hat{Z}$ of row vectors $\hat{\boldsymbol{z}}$ for $\boldsymbol{X}$ using
$\hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x})}[\mathbf{z}]$
Discretize $\hat{Z}$
$T \leftarrow L \times K$ matrix of zeros
for $l = 1$ **to** L **do**
  for $k = 1$ **to** K **do**
    Calculate $I(\hat{\mathrm{z}}_l; \mathrm{v}_k)$ from $\hat{\boldsymbol{z}}_l$ and $\boldsymbol{v}_k$ using Equation (4.7)
  end for
  $k^{(l)} \leftarrow \mathrm{argmax}_k\, I(\hat{\mathrm{z}}_l; \mathrm{v}_k)$
  for $k = 1$ **to** K **do**
    if $k = k^{(l)}$ **then**
      $t_{l,k} \leftarrow I(\hat{\mathrm{z}}_l; \mathrm{v}_{k^{(l)}})$
    end if
  end for
end for
Return DisentanglementScore $\leftarrow \frac{1}{L}\sum_{l=1}^{L}(1 - \frac{1}{(K-1)}\sum_{k=1}^{K}\left(\frac{I(\hat{\mathrm{z}}_l; \mathrm{v}_k) - t_{l,k}}{I(\hat{\mathrm{z}}_l; \mathrm{v}_{k^{(l)}})}\right)^2)$

Predictor-based metrics

Predictor-based metrics produce a score based on training predictors to predict factor values from model representations $\hat{\mathbf{z}}$ [Zaidi et al., 2021].

In addition to the Modularity metric presented above, Ridgeway and Mozer [2018] also introduce a metric to measure explicitness, called the Explicitness metric. The metric is based on training logistic regressors to recover factor values from $\hat{\mathbf{z}}$. It assumes discrete factors, and for each factor v_k with values in $\{v_{k,1}, v_{k,2}, ..., v_{k,J}\}$, J one-verus-rest logistic regressors f_{kj} are trained, such that each f_{kj} takes as input $\hat{\mathbf{z}}$ and predicts whether v_k takes value $v_{k,j}$ or not. The ROC curve of each f_{kj} is created, and the area under the ROC curve is recorded, denoted AUC_{kj} for area under ROC curve of logistic regressor f_{kj}. The final score is the average of all AUC_{kj} over factors and their values. Details of the Explicitness metric are shown in Algorithm 6.

Using one logistic regressor for each value of each factor detects whether the value can be identified from $\mathbf{v}$, but does not make assumption about the mapping, and the metric should measure non-linear explicitness.

Separated Attribute Predictability (SAP) is a predictor-based disentanglement metric introduced in [Kumar et al., 2018]. It is based on predicting factor values from

Algorithm 6 Explicitness Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, ground truth generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$
Sample a matrix V of N row vectors of generative factors $\boldsymbol{v}$
Sample a tensor $\boldsymbol{X}$ of data points $\boldsymbol{x}$ for V using $p^*(\mathbf{x}|\mathbf{v})$
Obtain representation matrix $\hat{Z}$ of row vectors $\hat{\boldsymbol{z}}$ for $\boldsymbol{X}$ using $\hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x})}[\mathbf{z}]$
Discretize V
From V, get K sets $\mathbb{J}^{(k)}$ of all values present in V for each factor
for $k = 1$ **to** K **do**
  for j **in** $\mathbb{J}^{(k)}$ **do**
    Train a logistic regressor f_{kj} to predict whether elements of $\boldsymbol{v}_k$ equal j or not from input $\hat{Z}$
    Create ROC_{kj}, the ROC curve of f_{kj}
    $AUC_{kj} \leftarrow$ Area under ROC_{kj}
  end for
end for
Return DisentanglementScore $\leftarrow \frac{1}{K}\sum_{k=1}^{K}\frac{1}{J^{(k)}}\sum_{j=1}^{J^{(k)}} AUC_{kj}$

$\hat{\mathbf{z}}$ under the assumption that there is a linear mapping from $\hat{\mathbf{z}}$ to $\mathbf{v}$. The metric creates a $L \times K$ matrix of scores where the value in row l, column k corresponds to how well v_k is predicted from $\hat{\mathrm{z}}_l$.

The metric is constructed to deal with both continuous and discrete factor variables. If the generative factor v_k is continuous, the relationship to $\hat{\mathrm{z}}_l$ is evaluated by linear regression, and the R^2-score is used as the matrix entry. R^2 is calculated as

$$\frac{\mathrm{cov}(\hat{z}_l, v_k)^2}{\mathrm{var}(\hat{z}_l)\mathrm{var}(v_k)} \tag{4.8}$$

If v_k is discrete, a linear classifier is trained to predict the value of v_k from $\hat{\mathrm{z}}_l$ and its accuracy is reported as the score entry of the matrix. If a dimension of $\hat{\mathrm{z}}$ has variance close to 0 it is considered inactive and the corresponding score matrix entry is set to 0.

When all values of the score matrix are computed, for each column the difference of the two entries with the greatest score is calculated, and SAP is then taken to be the mean of the K differences. Calculation details are shown in Algorithm 7.

Considering the difference between the two greatest entries of each score matrix column, ensures that in order for the final score to be high, there has to be exactly one entry with a high score in each column, corresponding to high compactness of $\hat{\mathrm{z}}$. This is analogous to the differences considered by the MIG metric. Low compactness conversely results in a low score. A high score requires recovery of values of $\mathbf{v}$ from $\hat{\mathbf{z}}$, which is restricted to linear mappings, and the metric therefore also measures linear explicitness. High or low modularity does not affect the score [Zaidi et al., 2021].

While designed to only report high compactness when there is a linear mapping from $\hat{\mathbf{z}}$ to $\mathbf{v}$, Sepliarskaia et al. [2021] show that SAP can return optimistic compact-

Algorithm 7 SAP Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, ground truth generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$
Sample matrices V and V^{test} of N row vectors of generative factors $\boldsymbol{v}$
Sample a dataset $\boldsymbol{X}$ and a test set $\boldsymbol{X}^{\text{test}}$ for V and V^{test} using $p^*(\mathbf{x}|\mathbf{v})$
Get representation matrices $\hat{Z}$ and $\hat{Z}^{\text{test}}$ of row vectors $\hat{\boldsymbol{z}}$ for $\boldsymbol{X}$ and $\boldsymbol{X}^{\text{test}}$ using $\hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x})}[\mathbf{z}]$
$S \leftarrow L \times K$ matrix
for $k = 1$ **to** K **do**
  for $l = 1$ **to** L **do**
    if $\text{var}(\hat{\text{z}}_l) \approx 0$ **then**
      $s_{l,k} \leftarrow 0$
    else if v_k is continuous **then**
      $s_{l,k} \leftarrow \dfrac{\text{cov}(\hat{\boldsymbol{z}}_l, \boldsymbol{v}_k)^2}{\text{var}(\hat{\boldsymbol{z}}_l)\text{var}(\boldsymbol{v}_k)}$
    else
      Train a balanced linear classifier to predict $\boldsymbol{v}_k$ from $\hat{\boldsymbol{z}}_l$
      $s_{l,k} \leftarrow$ Accuracy of linear classifier on predicting $\boldsymbol{v}_k^{\text{test}}$ from $\hat{\boldsymbol{z}}_l^{\text{test}}$
    end if
  end for
  $l^{(k)} \leftarrow \text{argmax}_l\, s_{l,k}$
  $l'^{(k)} \leftarrow \text{argmax}_{l \neq l^{(k)}}\, s_{l,k}$
end for
Return DisentanglementScore $\leftarrow \frac{1}{K}\sum_k (s_{l^{(k)},k} - s_{l'^{(k)},k})$

ness scores if factors in $\mathbf{v}$ are encoded in $\hat{\mathbf{z}}$ through a compact linear mapping in a subset of dimensions in $\hat{\mathbf{z}}$ as well as through a non-linear mapping in the remaining dimensions of $\hat{\mathbf{z}}$. The non-linear encoding of a factor may not be discovered by the linear predictors of the metric, thus leading to the metric only considering the subspace of $\hat{\mathbf{z}}$ linearly encoding $\mathbf{v}$, falsely recognising the representation as compact.

Another predictor-based approach is introduced in [Eastwood and Williams, 2018], which is a framework consisting of three different metrics in order to measure modularity, compactness and explicitness separately. In their work modularity is referred to as disentanglement, compactness as completeness and explicitness as informativeness.

The metrics are based on training K predictors in order to predict the value of the K generative factors in $\mathbf{v}$ from $\hat{\mathbf{z}}$. Predictor f_j predicts v_j given $\hat{\mathbf{z}}$, and the predictors should be able to provide relative importance's of each $\hat{\text{z}}_i$ in predicting v_j, such that a relative importance matrix R can be constructed, where $r_{i,j}$ is the relative importance of $\hat{\text{z}}_i$ in predicting v_j. Usually two versions are considered, one using linear predictors and one using non-linear.

The Informativeness metric, measuring the explicitness of representation $\hat{\mathbf{z}}$ relative to generative factors $\mathbf{v}$, is given by the average prediction error $\frac{1}{K}\sum_k E(\text{v}_k, f_k(\hat{\mathbf{z}}))$, where E is an error function [Eastwood and Williams, 2018]. The use of linear

predictors $f_k(\hat{\mathbf{z}})$ will restrict the metric to measuring linear explicitness, while non-linear predictors will measure non-linear explicitness.

The Disentanglement metric, measuring the modularity of representation $\hat{\mathbf{z}}$ relative to generative factors $\mathbf{v}$, calculates a score by estimating probabilities for how likely it is that $\hat{\mathrm{z}}_l$ is important in estimating only v_k:

$$p_{l,k} = \frac{\text{Importance of } \mathrm{z}_l \text{ in predicting } \mathrm{v}_k}{\text{Importance of } \mathrm{z}_l \text{ in all predictions}} = \frac{r_{l,k}}{\sum_{k'=1}^{K} r_{l,k'}} \tag{4.9}$$

From this the entropy $H_K(p_{l\cdot}) = -\sum_{k=1}^{K} p_{l,k} \log_K p_{l,k}$ is calculated, and the score for dimension l is given by $1 - H_K(p_{l\cdot})$. In order to give an overall score, $\rho_l = \frac{\sum_{k=1}^{K} r_{l,k}}{\sum_{l'=1}^{L}\sum_{k=1}^{K} r_{l',k}}$ is used to construct a weighted average $\sum_{l=1}^{L} \rho_l(1-H_K(p_{l\cdot}))$ [Eastwood and Williams, 2018]. ρ_l is used such that if a dimension z_l is inactive when predicting all factors $\mathbf{v}$, ρ_l will be small and $\hat{\mathrm{z}}_l$'s contribution to the overall score will be low.

The Completeness metric, measuring the compactness of representation $\hat{\mathbf{z}}$ relative to generative factors $\mathbf{v}$, calculates a score by estimating probabilities for how important each $\hat{\mathrm{z}}_l$ is when predicting v_k [Zaidi et al., 2021]:

$$\tilde{p}_{l,k} = \frac{\text{Importance of } \mathrm{z}_l \text{ in predicting } \mathrm{v}_k}{\text{Importance of all z}'\text{s in predicting } \mathrm{v}_k} = \frac{r_{l,k}}{\sum_{l'=1}^{L} r_{l',k}} \tag{4.10}$$

The entropy $H_L(\tilde{p}_{\cdot k}) = -\sum_{l=1}^{L} \tilde{p}_{l,k} \log_L \tilde{p}_{l,k}$ is calculated, and the score for factor k is given by $1 - H_L(\tilde{p}_{\cdot k})$. The average $\frac{1}{K}\sum_{k=1}^{K}(1 - H_L(\tilde{p}_{\cdot k}))$ is reported as the overall score [Eastwood and Williams, 2018].

Algorithm 8 shows the calculation details of the Disentanglement, Compactness and Informativeness metrics, with the error function E being the accuracy of the predictors when run on test data.

As the predictor-based metrics presented here are based on predicting factor values from representations, none of them requiring that all dimensions of $\hat{\mathbf{z}}$ should contribute information in the predictions, they can be applied to measure partial disentanglement on datasets with factor labels, and Algorithms 6, 7 and 8 can be modified to take $\mathbb{X}^*$ as input.

Algorithm 8 DCI Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, ground truth generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$
Sample matrices V and V^{test} of N row vectors of generative factors $\boldsymbol{v}$
Sample a dataset $\boldsymbol{X}$ and a test set $\boldsymbol{X}^{\text{test}}$ for V and V^{test} using $p^*(\mathbf{x}|\mathbf{v})$
Get representation matrices $\hat{Z}$ and $\hat{Z}^{\text{test}}$ of row vectors $\hat{\boldsymbol{z}}$ for $\boldsymbol{X}$ and $\boldsymbol{X}^{\text{test}}$ using $\hat{\boldsymbol{z}} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x})}[\mathbf{z}]$
$R \leftarrow L \times K$ matrix
$P \leftarrow L \times K$ matrix
$\tilde{P} \leftarrow L \times K$ matrix
for $k = 1$ **to** K **do**
  Train a predictor f_k to predict $\boldsymbol{v}_k$ from $\hat{Z}$
  for $l = 1$ **to** L **do**
    $r_{l,k} \leftarrow$ Relative importance of z_l in predicting v_k using f_k
  end for
  $I_k \leftarrow$ Accuracy of f_k on predicting $\boldsymbol{v}_k^{\text{test}}$ from $\hat{Z}$
end for
$I \leftarrow \frac{1}{K}\sum_{k=1}^{K} I_k$
for $l = 1$ **to** L **do**
  for $k = 1$ **to** K **do**
    $p_{l,k} \leftarrow \frac{r_{l,k}}{\sum_{k'=1}^{K} r_{i,k'}}$
  end for
  $D_l \leftarrow 1 + \sum_{k'=1}^{K} p_{l,k'} \log p_{l,k'}$
  $\rho_l \leftarrow \frac{\sum_{k'=1}^{K} r_{l,k'}}{\sum_{l'=1}^{L}\sum_{k'=1}^{K} r_{l',k'}}$
end for
$D \leftarrow \sum_{l=1}^{L} \rho_l D_l$
for $k = 1$ **to** K **do**
  for $l = 1$ **to** L **do**
    $\tilde{p}_{l,k} \leftarrow \frac{r_{l,k}}{\sum_{l'=1}^{L} r_{l',k}}$
  end for
  $C_k \leftarrow 1 + \sum_{l'=1}^{L} p_{l',k} \log p_{l,k'}$
end for
$C \leftarrow \frac{1}{K}\sum_{k=1}^{K} C_k$
Return DisentanglementScores $\leftarrow (D, C, I)$

Metric	Modularity	Compactness	Explicitness	Partial $\mathbf{v}$
Z-diff	✓[1)]			✓
Z-min Variance	✓[1)]			✓
IRS	✓		✓	
MIG		✓	✓	✓
Modularity	✓[2)]			
Expliciteness			✓	✓
SAP		✓[3)]	✓[4)]	✓
Disentanglement	✓			✓
Completeness		✓		✓
Informativeness			✓	✓

Table 4.2: Summary of supervised disentanglement metrics. [1)] Failure modes of Z-diff and Z-min Variance are identified in [Zaidi et al., 2021; Sepliarskaia et al., 2021], and the resulting modularity score is less reliable. [2)] Modularity scores may be optimistic compared to actual modularity of representation due to normalization by maximal mutual information [Zaidi et al., 2021] [3)] A failure mode of SAP is identified in [Sepliarskaia et al., 2021] caused by the linearity assumption. [4)] Linear explicitness only.

A summary of the metric presented in this section in presented in Table 4.2. General results from [Zaidi et al., 2021] also show that metrics using discretization of $\mathbf{v}$ and/or $\hat{\mathbf{z}}$ perform less accurate if the mapping between $\mathbf{v}$ and $\hat{\mathbf{z}}$ is noisy or non-linear, caused by inaccurate or uneven binning of values.

The supervised metrics are dependent upon an assumption that there is a known representation $\mathbf{v}$ of the ground truth generative factors. While $\hat{\mathbf{z}}$ is not required to be equal to $\mathbf{v}$ in order to give a high score using the metric, the choices in designing $\mathbf{v}$ will limit the ways that $\hat{\mathbf{z}}$ can be constructed in order to be considered disentangled.

To illustrate this, consider a dataset of images of simple objects on a neutral background. The objects position could be considered a generative factor independent of any other factors. In order to describe position using one dimensional random variables one could choose to register Cartesian coordinates x and y as separate continuous random variables. A model representation $\hat{\mathbf{z}}$ that encodes the objects position in two dimensions z_i and z_j according to polar coordinates would succeed in disentangling equally well as the Cartesian $\mathbf{v}$, but compared to $\mathbf{v}$ all modularity and compactness metrics discussed in this section would consider $\hat{\mathbf{z}}$ entangled relative to $\mathbf{v}$.

Another analogous example would be to consider object colour an independent factor, and deciding between encoding the colour using RGB, CYMK or any other color representation. Comparing one such encoding to another would suggest entanglement, when in fact both are disentangled.

One way to attempt to correct for such issues could be to discretize the posi-

tion/colour space in $\mathbf{v}$ and assign each value a single number. An objects position in an image could be represented by counting pixels and assigning the number of its center pixel c as its position. Now both Cartesian x, y and polar r, θ are modular representations of c, and if measured by a reliable metric not inspecting for compactness, a good score should be assigned. Intervention-based metrics also allow for fixing more than one dimension at a time, such that fixing e.g. x and y simultaneously could be considered when calculating the metric.

However, when working with complex data, the task of designing $\mathbf{v}$ becomes considerably more complex, and assumptions on which atomic factors are indeed independent both statistically and conceptually given all others are harder to make. Deciding upon factors to include in $\mathbf{v}$ is not straight forward, and ideally one would want to avoid labeling $\mathbf{v}$'s all together.

4.3.2 Unsupervised metrics

In order to avoid having to design a representation $\mathbf{v}$ of the ground truth generative factors, it is desirable to construct unsupervised disentanglement metrics. Access to such a metric would allow measuring disentanglement of a given model without making any assumptions about factors and factor decomposition.

The informal definitions of disentanglement consider independence and interpretability as required properties [Bengio et al., 2014; Higgins et al., 2017], sometimes also referred to as statistical and conceptual independence [Zaidi et al., 2021]. When evaluating disentangled representations, one must therefore identify statistically mutually independent factors that additionally have a meaningful interpretation if inspected by humans.

In the Disentanglement Library created in association with [Locatello et al., 2019], several unsupervised scores are calculated based on mutual information and total correlation. Low averaged mutual information between the different dimensions of $\hat{\mathbf{z}}$ imply pairwise statistical independence and low total correlation imply mutual statistical independence. Such metrics will only evaluate the statistical independence of dimensions in a model representation, and will not say anything about conceptual independence. While a low score will correspond to representations of low modularity, a high score can not be interpreted as disentanglement.

Duan et al. [2020] introduce a framework for calculating disentanglement called Unsupervised Disentanglement Ranking (UDR). The theory behind the UDR metric is based on results from [Rolinek et al., 2019], which show that disentangling VAEs, trained on non-adversarial datasets, tend to converge to the same disentangled representations, up to permutations and inverse signs. In contrast, entangling VAE's tend to converge to different entangled representations despite having been trained with the same architecture and hyperparameter settings [Duan et al., 2020].

The UDR metric is based on training many models for a given model architecture, with different hyperparameters and initial model weights. Models with the same hyperparameter setting are then pairwise compared by calculating their UDR score, which is done by constructing a similarity matrix $R \in \mathbb{R}^{L \times L}$, given representations of length L. Recording similarity between all pairs of dimensions, one from each model representation, allow for assigning a high score to permutations. The

absolute value of each entry is used in order to cancel out sign inverse.

Given a dataset $\mathbb{X}$ and two models m and u, each entry r_{ij} of R is calculated as the similarity between $\boldsymbol{z}_i^{(m)}$, the values of the i'th dimension of representations produced from $\mathbb{X}$ using model m, and $\boldsymbol{z}_j^{(u)}$, the values of the j'th dimension of representations produced from $\mathbb{X}$ using model u. Similarity is suggested calculated using Lasso regression [Duan et al., 2020] to predict one value form the other, recording accuracy of the predictor as the similarity score.

For a representation $\mathbf{z}^{(m)}$ from model m with encoder $q_{\boldsymbol{\phi}}^{(m)}$, dimension l is defined as informative if it has learned a posterior which diverges from them prior $p^{(m)}$ of the model:

$$I_{KL}(\mathrm{z}_l^{(m)}) = \begin{cases} 1 & D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}^{(m)}(\mathrm{z}_l|\mathbf{x})\|p^{(m)}(\mathrm{z}_l)) > 0.01 \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

The UDR-score between models m and u is calculated from entries $r_{i,j}$ of similarity matrix R as:

$$\overline{\mathrm{UDR}}_{mu} = \frac{1}{2} \cdot \left[\sum_{j=1}^{L} \frac{\max_i |r_{i,j}| \cdot I_{KL}(\mathrm{z}_j^{(u)})}{\sum_{i=1}^{L} |r_{i,j}|} + \sum_{i=1}^{L} \frac{\max_j |r_{i,j}| \cdot I_{KL}(\mathrm{z}_i^{(m)})}{\sum_{j=1}^{L} |r_{i,j}|} \right] \tag{4.12}$$

$$\mathrm{UDR}_{mu} = \frac{\overline{\mathrm{UDR}}_{mu}}{\frac{1}{2} \cdot [\sum_{i=1}^{L} I_{KL}(\mathrm{z}_i^{(m)}) + \sum_{j=1}^{L} I_{KL}(\mathrm{z}_j^{(u)})]} \tag{4.13}$$

As seen in equation (4.13), calculating the UDR-score involves a division by the average number of informative latents in $\mathbf{z}^{(m)}$ and $\mathbf{z}^{(u)}$, which is done in order to account for the case where the two models have learned different subsets of factors [Duan et al., 2020].

Given a VAE-based disentangling model architecture, for H different hyperparameter settings of the model and S different seeds, $M = H \times S$ models are trained, covering all combinations of hyperparameters and seeds. Then, for each model m, $1 \leq m \leq M$, $U \leq S$ models are sampled without replacement from the S trained models that have the same hyperparameter setting as m. For each of these models $u, 1 \leq u \leq P$, compare representations created by m and u and calculate their UDR score, denoted UDR_{mu}. For each model m, the UDR_m score is calculated as the median of the U UDR_{mu} scores. Details are summarized in Algorithm 9.

Disentanglement as measured by UDR corresponds to modularity [Duan et al., 2020].

Theoretical argumentation and empirical results in [Rolinek et al., 2019; Duan et al., 2020] suggest the UDR metric works on VAE-based models β-VAE, CCI-VAE, FactorVAE, β-TCVAE, DIP-VAE-I and DIP-VAE-II.

Algorithm 9 UDR Metric

Input A model architecture A, hyperparameter settings $\mathbb{H} = \{h_1, ..., h_H\}$, seeds $\mathbb{S} = \{s_1, ..., s_S\}$, design matrix $\boldsymbol{X}$
$\mathbb{M} \leftarrow \{\}$
for (h, s) **in** $\mathbb{H} \times \mathbb{S}$ **do**
  Train a model m from A initialized from seed s using hyperparameter h
  $\mathbb{M} \leftarrow \mathbb{M} \cup \{m\}$
  $Z^{(m)} \leftarrow$ representations created from $\boldsymbol{X}$ using model m
  $\boldsymbol{a}^{(m)} \leftarrow$ Vector of zeros of length L
  for $l = 1$ **to** L **do**
    if $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}^{(m)}(\mathrm{z}_l|\mathbf{x}) \| p^{(m)}(\mathrm{z}_l)) > 0.01$ **then**
      $a_l^{(m)} \leftarrow 1$
    end if
  end for
end for

for m **in** $\mathbb{M}$ **do**
  $\mathbb{U} \leftarrow$ A subset of $\{\tilde{m} \mid \tilde{m} \in \mathbb{M}, \tilde{m} \neq m, h \text{ of } \tilde{m} = h \text{ of } m\}$ of size $U \leq S$
  $R \leftarrow L \times L$ matrix
  for u **in** $\mathbb{U}$ **do**
    for (i, j) **in** $\{1, ..., L\} \times \{1, ..., L\}$ **do**
      Train a Lasso regressor to predict $\boldsymbol{z}_i^{(m)}$ from $\boldsymbol{z}_j^{(u)}$
      $r_{i,j} \leftarrow$ Prediction accuracy of trained regressor
    end for
    $$\mathrm{UDR}_{mu} \leftarrow \frac{1}{\sum_{i=1}^{L} a_i^{(m)} + \sum_{j=1}^{L} a_j^{(u)}} \left[\sum_{j=1}^{L} \frac{\max_i r_{i,j} \cdot a_j^{(u)}}{\sum_{i=1}^{L} r_{i,j}} + \sum_{i=1}^{L} \frac{\max_j r_{i,j} \cdot a_i^{(m)}}{\sum_{j=1}^{L} r_{i,j}} \right]$$
  end for
  $\mathrm{UDR}_m \leftarrow$ Median of the U scores UDR_{mu}
end for
Return UDRscores $\leftarrow \{\mathrm{UDR}_m\}_{m=1}^{|\mathbb{M}|}$

Chapter 5

Experiments and Results

5.1 Experimental Plan

The hypothesis to be tested is that disentangling data representations will result in increased performance in learning downstream tasks.

The experiment is based on a group of models that learn disentangled representations. 1800 disentangling models from the Disentanglement library (DisentanglementLib) [Locatello et al., 2019] that learn unsupervised disentangled representations are considered, trained on the dSprites dataset [Matthey et al., 2017]. The models are evaluated on both level of disentanglement and suitability as input to prediction models, in order to test the hypothesis.

The 1800 models used for these experiment are VAE-based and share the same architecture, with all models learning representations of 10 dimensions. They differ in the objective function used during training and in weight initialization.

The level of disentanglement achieved by each model is evaluated using disentanglement metrics as discussed in Section 4.3. In order to evaluate the reliability of the disentanglement scores assigned by the metrics, their agreement on the set of models is quantified by calculating pairwise correlation coefficients. In particular, metrics that measure the same disentanglement criteria (modularity, compactness and explicitness in Table 4.1) are expected to be related by a linear relationship, and thus share a strong correlation. As all models learn representations of equal size, disentanglement scores are assumed to be model independent given true level of disentanglement, and thus to reflect only the level of modularity, compactness or explicitness.

The downstream suitability of representations is evaluated by considering the accuracy achieved by a set of predictors receiving the representations as input. The expected outcome is that the performance of predictors should improve as disentanglement increases, and the impact of disentanglement on predictor performance is quantified by correlation coefficients between disentanglement metrics and predictor accuracy scores.

The experiment setup is shown in Figure 5.1. The setup is further detailed in Section 5.2.

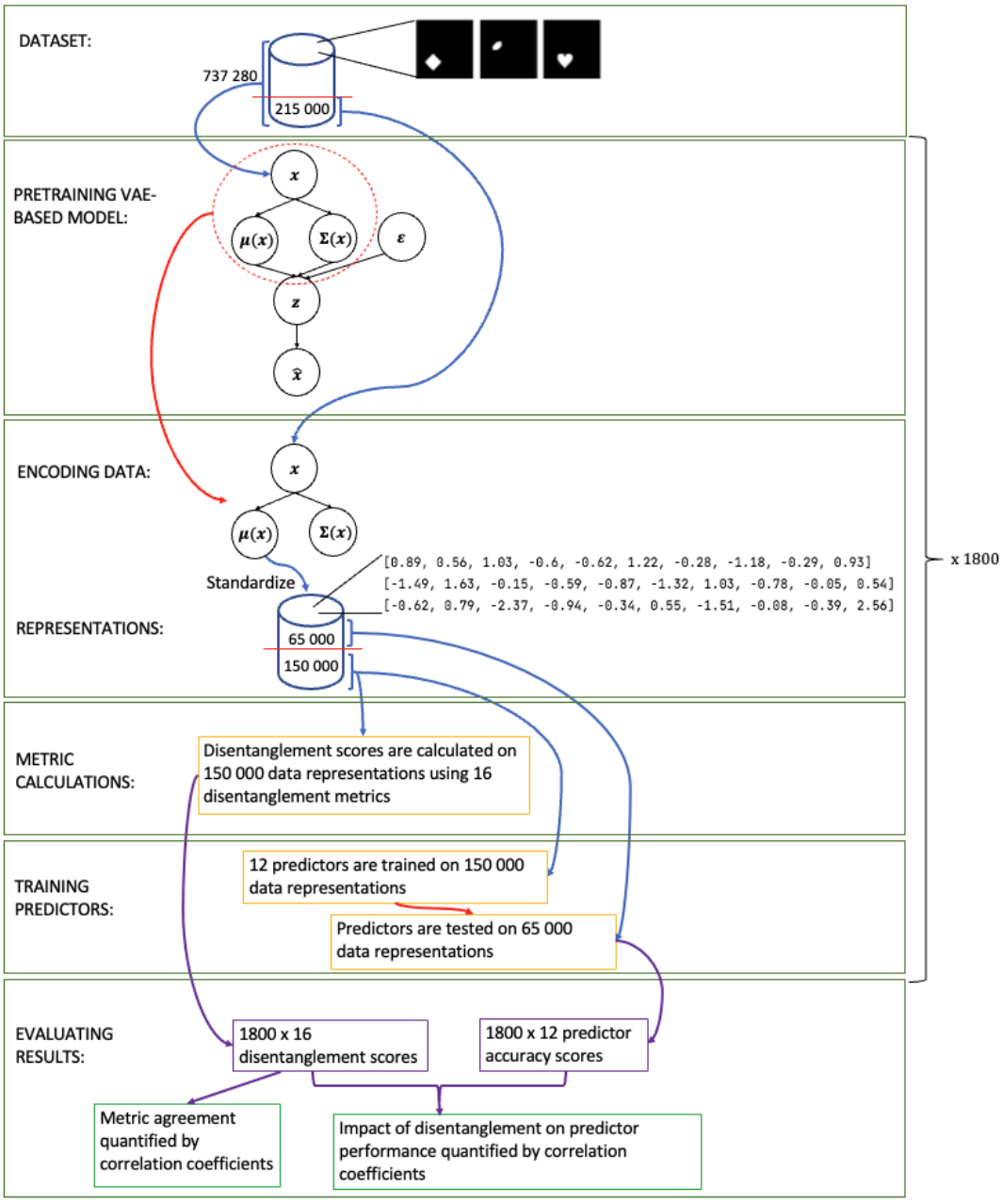


Figure 5.1: The figure shows the experiment setup. Blue arrows correspond to flow of data, red arrows to flow of models, and purple arrows to flow of calculated scores. The examples of data representations shown in this figure show values rounded to two decimal precision for readability.

5.2 Experimental Setup

5.2.1 Metrics

The metrics used to calculate scores in this experiment are the unsupervised metric UDR measuring modularity, the supervised metrics Z-diff, Z-min Variance (Z-min-var), IRS and Modularity measuring modularity, Explicitness measuring explicitness, MIG and SAP measuring compactness, and linear DCI and nonlinear DCI measuring all three disentanglement criteria. The modularity score calculated by the DCI metrics is referred to as D, the compactness score as C and the explicitness score as I. All metrics are presented in Section 4.3.

The implementation of the UDR metric used in the experiment is from the DisentanglementLib [Locatello et al., 2019]. The UDR metric calculates representation similarity using Lasso regression, as suggested in [Duan et al., 2020].

Implementations of the supervised metrics are from the code accompanying the metrics review by Zaidi et al. [2021], with a few changes detailed in the following. The supervised metrics are all calculated using discrete factor class labels, with the predictor-based metrics implemented using classifiers. The SAP metric is implemented using logistic regression, in order to retain linearity. The linear DCI implementation uses logistic regression with the importance weights taken as the absolute value of the weights of the model. The nonlinear DCI implementation uses random forest classification, with the importance weights calculated as permutation importances.

The predictor-based metrics divide the dataset into training (7/10 of the data) and test set (3/10 of the data), and returns a score based on evaluating the trained classifiers on the test set. The permutation importances of the nonlinear DCI metric are evaluated on a subset of the training set, in order to correspond to the linear version where the importance weights are determined by the training set.

Out of the supervised modularity scores listed above, both IRS and Modularity expects a complete set of factor labels covering all generative factors of the data, thus limiting their flexibility and use outside of toy data experiments. The remaining metrics Z-diff, Z-min-var, linear D and nonlinear D all involve prediction of either factors or factor values from codes, potentially resulting in optimistic scores if disentanglement is calculated based on labels of few factors, and rendering them unable to calculate a modularity score for one factor only. A new predictor-free metric based on the Z-min-var metric is therefore suggested here to allow for more stable measurement of partial modularity.

As for Z-min-var (presented in Section 3.5), representation variance vectors are created by keeping a factor value fixed and sampling others at random, considering the corresponding representations and calculating the variance of each representation dimension. Dividing each dimension of the variance vector by the global variance of that dimension calculated over the entire dataset ensures a variance approaching 1 in each dimension before factor interventions.

The dimensions that encode information about the fixed factor are expected to have lower variance than the ones that do not encode this factor. Specifically, if the

Algorithm 10 Z (2) Metric

Input Trained probabilistic encoder $q_{\boldsymbol{\phi}}$, ground truth generative factor representation $\mathbf{v}$ with distribution $p^*(\mathbf{v})$, ground truth generative process $p^*_{\mathbb{X}}(\mathbf{x}|\mathbf{v})$,
$\boldsymbol{s}_g \leftarrow$ Empirical standard deviation over full data (or large random sample)
for $n = 1$ **to** N **do**
  Choose a factor $k \in \mathbb{G}$
  Fix a value $v_k^{(n)}$ of the factor representation v_k
  for $j = 1$ **to** J **do**
    Sample values $\boldsymbol{v}_{\setminus k}^{(j)}$ at random.
    Sample datapoint $\boldsymbol{x}^{(j)} \sim p^*(\mathbf{x}|\mathbf{v}_{\setminus k} = \boldsymbol{v}_{\setminus k}^{(j)}, \mathrm{v}_k = v_k^{(n)})$
    Infer $\hat{\boldsymbol{z}}^{(j)} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}=\boldsymbol{x}^{(j)})}[\mathbf{z}]$
    Obtain normalised $\bar{\boldsymbol{z}}^{(j)}$ by dividing $\hat{\boldsymbol{z}}^{(j)}$ by $\boldsymbol{s}_g$, $\bar{\boldsymbol{z}}^{(j)} = \frac{\hat{\boldsymbol{z}}^{(j)}}{\boldsymbol{s}_g}$
  end for
  $\bar{\bar{\boldsymbol{z}}} \leftarrow \frac{\sum_j \bar{\boldsymbol{z}}^{(j)}}{J}$
  $\boldsymbol{c} \leftarrow$ Empirical variance in each dimension of the $\bar{\boldsymbol{z}}^{(j)}$,
    $\mathrm{diag}(\frac{1}{J-1}\sum_j(\bar{\boldsymbol{z}}^{(j)} - \bar{\bar{\boldsymbol{z}}})(\bar{\boldsymbol{z}}^{(j)} - \bar{\bar{\boldsymbol{z}}})^T)$
  $c_{l^*}^{(n)} = \min_l c_l$

  $\tilde{\boldsymbol{c}}^{(n)} = \boldsymbol{c}_{\setminus l^*}$
  $c^{*(n)} = \frac{1}{L-1}\sum_{l=1}^{L-1} \tilde{c}_l^{(n)}(1 - \tilde{c}_l^{(n)})$
end for
$\bar{c}_{l^*} = \frac{1}{N}\sum_n c_{l^*}^{(n)}$
$\bar{c}^* = \frac{1}{N}\sum_n c^{*(n)}$
Return DisentanglementScore $\leftarrow 1 - (\bar{c}_{l^*} + 4\bar{c}^*(1 - \bar{c}_{l^*}))$

representation satisfies the modularity criterium with respect to the fixed factor, the representation dimensions encoding the factor should have variance approaching 0, while the dimensions not encoding the fixed factor are expected to have variance of approximately 1 after division by the global representation variance. Disregarding noise and assuming a large sample size, one would expect perfect modularity to correspond to variance vector dimensions either being close to 0 or close to 1.

The suggested metric calculates a modularity score by considering the lowest variance of each representation variance vector $\boldsymbol{c} \in \mathbb{C}$, denoted $c_{l^*} = \min_l c_l$, and assigns a score $1-\bar{c}_{l^*}$, where $\bar{c}_{l^*}$ is the average of all c_{l^*} in $\mathbb{C}$. A high score corresponds to there being a subset of the representation where some information about each of the labeled factors are disentangled according to modularity. However, additional and/or redundant information about the factors may still appear in the remaining representaion space in a non-modular fashion.

In order to include all dimensions in the score calculations, the average across the dimensions of $\boldsymbol{c}(1-\boldsymbol{c})$ could be considered in a score given by $1-4\frac{1}{L}\sum_{l=1}^{L} c_l(1-c_l)$. A high score now includes the case where no information about the factor is encoded in the representation (no explicitness), which is not consistent with the other modularity metrics.

Combining the two scores is suggested by extending $1 - \bar{c}_{l^*}$, considering the

remaining variances $\tilde{\boldsymbol{c}} = \boldsymbol{c}_{\setminus l^*}$. A score c^* of how modular they appear are given by $c^* = \frac{1}{L-1}\sum_{l=1}^{L-1} \tilde{c}_l(1-\tilde{c}_l)$, with $0 \leq 4c^* \leq 1$. The final score is given by $1 - (\bar{c}_{min} + 4\bar{c}^*(1-\bar{c}_{min}))$, where $\bar{c}^*$ is the average over $\mathbb{C}$. The metric requires clipping the dimensions of all $\boldsymbol{c}$ to $[0,1]$ in order to ensure the assigned score is in $[0,1]$.

The suggested metric is an intervention-based metric measuring modularity that allows for calculating partial disentanglement, including when labels of only one factor is present. Analogously to Z-min-var, a good score may be assigned despite the representation not displaying perfect explicitness. The metric is however not robust to noise. The lack of predictors presents an additional advantage in that the computational cost is lowered.

The metric is included in these results in two versions, Z (1): $1 - \bar{c}_{l^*}$ and Z (2): $1 - (\bar{c}_{l^*} + 4\bar{c}^*(1-\bar{c}_{l^*}))$. Z (2) is detailed in Algorithm 10.

The linear and nonlinear DCI metrics both calculate three different scores, while the remaining metrics calculate one score. Thus the total number of disentanglement scores considered in this experiment is 16. Of these, 8 are supervised modularity scores: Z-diff, Z-min-var, Z (1), Z (2), D (linear), D (nonlinear), IRS and Modularity. 4 are supervised compactness scores: MIG, SAP, C (linear) and C (nonlinear). 3 are supervised explicitness scores: I (linear), I (nonlinear) and Explicitness. The last score is the unsupervised UDR score.

As discussed in Section 4.3, the supervised metrics are classified as either intervention-based, information-based or prediction-based, grouping the metrics on the technique they use to compare factors and codes. Despite differences in construction, the metrics that measure the same properties should be expected to increase or decrease close to linearly relative to the true level of that property, and thus to other metrics measuring the same property, even if values do not coincide. In order to establish reliability of the metrics given the other metrics, pairwise correlation of metric scores is quantified using the Pearson correlation coefficient, measuring the strength of a linear relationship between the metrics.

IRS, MIG and SAP measure explicitness in addition to modularity/compactness. The influence of multiple properties may impact their correlation with other metrics measuring only one property.

5.2.2 Dataset

The experiment is conducted on the dSprites dataset [Matthey et al., 2017]. This is a synthetic dataset generated from 5 factors of variation: shape (3 values), size (6 values), orientation (40 values), x-coordinate of position (32 values) and y-coordinate of position (32 values). Some samples from the dataset are shown in Figure 5.2. The factors and their values are presented in detail in Appendix B.

The dataset consists of 737 280 samples, one for each factor configuration. In order to limit computational cost, a subset of the data is used in calculating metrics and predictions. Score convergence is considered to determine the size of this subset. The I score calculated by the nonlinear DCI metric, using the accuracy of random forest classifiers to measure representation explicitness, is identified as the score that requires the largest dataset in order to converge to a stable score. Figure 5.3

Figure 5.2: Samples from the dSprites dataset

shows the nonlinear DCI metric scores plotted against dataset size. The plot to the left shows the I score. Calculating the score on the full dataset is seen to be required in order to reflect actual explicitness of the representations, however as the dataset size grows, the resulting increase in the score is less significant.

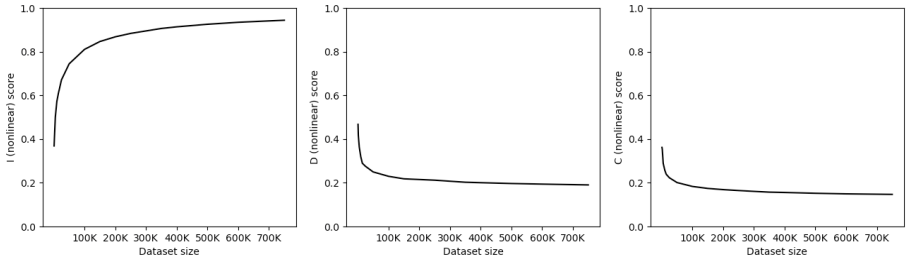


Figure 5.3: Nonlinear DCI metric convergence. The plot to the left shows the I score as dataset size increases, along with the D (middle) and C (right) scores.

Due to computational cost, a data subset of size 150 000 is randomly chosen for metric calculation, as increasing the sample size beyond this show limited impact on the D (Figure 5.3, middle) and C (Figure 5.3, right) scores. The resulting nonlinear DCI scores may however be slightly inaccurate, as explicitness may be slightly underestimated.

For comparison, similar plots for linear DCI are included in Figure 5.4, and plots for the remaining metrics are included in Appendix C.

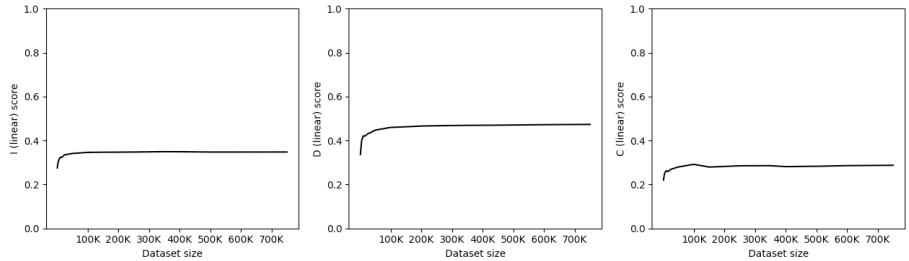


Figure 5.4: Linear DCI metric convergence. The plot to the left shows the I score as dataset size increases, along with the D (middle) and C (right) scores.

Predictors are trained on the same dataset of size 150 000 that disentanglement is calculated on. Another subset of 65 000 samples is chosen as a test set for evaluating predictor accuracy. The disentangled representation learning models

used in the experiment are trained using the entire dSprites dataset, thus data representations are in this case learned with access to information from the test set, potentially overestimating test data accuracy. As this is the case for all models that are compared in this experiment, this should not impact results significantly.

5.2.3 Disentanglement learning models

The models used in the experiment are pretrained models from the DisentanglementLib [Locatello et al., 2019]. 1800 unsupervised VAE-based models are considered, identified by numbers 0-1799, trained to reconstruct the dSprited dataset with a representation space of 10 dimensions. There are 300 models of each of the unsupervised VAE variants β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE, β-TCVAE and CCI-VAE, introduced in Section 3.4. The model encoder and decoder architecture, shared by all 1800 models, is shown in Appendix D.1, along with the Factor-VAE discriminator architecture. Hyperparameters shared by all models are listed in Appendix D.2.

Table 5.1 shows the hyperparameters that differ between the models along with their values.

The dSprites data samples are preprocessed using each of the 1800 pretrained variational encoders, taking the mean of the encoder output distribution as the data representation. After encoding using a variational encoder, the resulting set of 150 000 data representations is scaled to have a mean of 0 and a variance of 1 over the representation dataset, before metric calculation and predictor training. Predictor test data is scaled according to training data mean and variance.

Model numbers	Model	Parameter values
0-299	β-VAE	$\beta \in [1, 2, 4, 6, 8, 16]$
300-599	FactorVAE	$\gamma \in [10, 20, 30, 40, 50, 100]$
600-899	DIP-VAE-I	$\lambda_{od} \in [1, 2, 5, 10, 20, 50] \quad (\lambda_d = 10\lambda_{od})$
900-1199	DIP-VAE-II	$\lambda_{od} \in [1, 2, 5, 10, 20, 50] \quad (\lambda_d = \lambda_{od})$
1200-1499	β-TCVAE	$\beta \in [1, 2, 4, 6, 8, 10]$
1500-1799	CCI-VAE	$C_{max} \in [5, 10, 25, 50, 75, 100] \quad (\gamma = 1000)$

Table 5.1: The parameter settings of the pretrained models from the Disentanglement Library. 50 models initialized from different seeds are trained for each parameter value.

5.2.4 Predictors

Four learning tasks are defined on the dataset. These are classification of object shape into three classes, and regression of size, position in Cartesian coordinates and position in polar coordinates. Logistic regression, gradient boosting classification

and random forest classification are considered for the classification task, and linear regression, gradient boosting regression and random forest regression are considered for the regression tasks. Accuracy on Cartesian coordinate position is reported as the average of the accuracy on predicting the $x \in [0, 1]$ and $y \in [0, 1]$ coordinate, and accuracy on polar coordinate position is reported as the average of the accuracy of predicting the angle, $\theta \in [0, \frac{\pi}{2}]$, and radial distance, $\rho \in [0, \sqrt{2}]$, after converting new label values from original x and y values.

All predictors are implemented using the Scikit-Learn library [Pedregosa et al., 2011].

In total, 12 different tasks are solved on each set of data representations, and 12 accuracy scores are reported. Prediction accuracy scores are compared to level of disentanglement as measured by the 16 disentanglement metrics in the experiment, and the relationship between disentanglement and predictor performance is quantified by Pearson correlation coefficients.

5.2.5 Computations

The computation of metric scores and prediction accuracies constituting the results presented in this chapter were performed on resources provided by the NTNU IDUN/EPIC computing cluster [Själander et al., 2019].

5.3 Experimental Results

For representations from each of the 1800 models, the following disentanglement scores are computed: scores given by the supervised modularity metrics Z-diff, Z-min Variance (Z-min-var), IRS, Modularity, linear DCI Disentanglement (D) and nonlinear D, the supervised compactness metrics MIG, SAP, linear DCI Completeness (C) and nonlinear C, the supervised explicitness metrics Explicitness, linear DCI Informativeness (I) and nonlinear I, and the unsupervised modularity metric UDR. Supervised scores are calculated using class labels of factors shape, size, orientation, x coordinate and y coordinate. As UDR provides a score that is independent of factor labels, this score is considered particularly important when presenting the results of this experiment.

Metrics measuring the same disentanglement property are expected to correlate strongly across model groups if the metric scores does in fact reflect the true level of the measured property of the representations. While UDR measures modularity, this is not in reference to any ground truth factors, and the relationship with the other modularity metrics is unclear. Examining Pearson correlation coefficients should provide some indication of how reliable the different metrics are given scores from the other metrics.

Figure 5.5 shows the correlation coefficients of the scores assigned by all 16 metrics for representations from the 1800 models in the experiment.

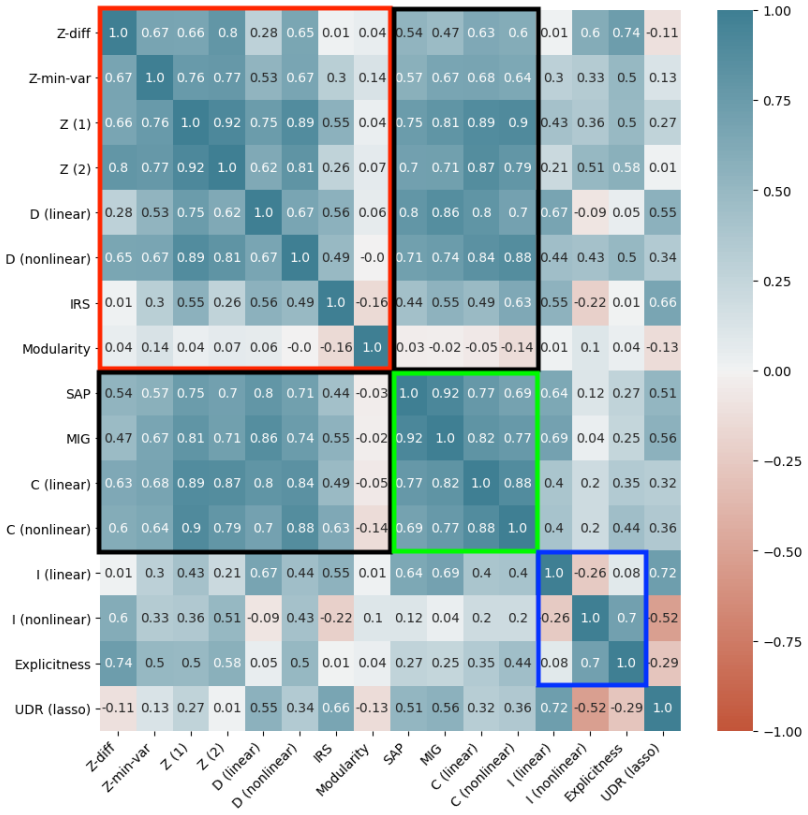


Figure 5.5: The correlation matrix of all metrics calculated for the 1800 experiment model representations. The red box shows correlation among the modularity metrics, the green box shows correlation among compactness metrics and the blue box shows correlation among explicitness metrics. The black boxes shows correlation of modularity metrics with compactness metrics.

Summarizing Figure 5.5:
Modularity metrics (red box): All metrics show positive correlation, except for Modularity, which does not appear well correlated with any metrics.
Compactness metrics (green box): All metrics correlate well with each other.
Explicitness metrics (blue box): The blue box contains the metrics that measure only explicitness. The nonlinear metrics are correlated, while the linear metric does not correlate well with the others. IRS, SAP and MIG are also mentioned in Section 4.3 as measuring explicitness, however as these metrics also measure other disentanglement properties correlation is not as easily interpreted.
Modularity and compactness metrics (black box): These metrics appear to be well correlated on these models, again with Modularity as the exception.
UDR (bottom row): The unsupervised disentanglement metric shows varying correlation with the supervised metrics, possibly indicating some ambiguity in what the metric scores are reflecting.

Modularity metrics and compactness metrics are intended to capture different properties of representations, and the level of correlation between these groups would not in general be expected to be as strong as correlation of metrics within these groups. However, they are related in that for a representation space and a generative factor space of equal size, perfect modularity of all factors in the representation space is equivalent to perfect compactness. In this experiment the scores are calculated from representations having 10 dimensions while assuming a total of 5 factors, and thus perfect modularity will follow from perfect compactness, while the reverse is not necessarily true.

From Figure 5.5, Modularity appear to assign scores of particularly low correlation with all other metrics. Pairwise comparison of Modularity with all other metrics is detailed in Appendix E.1, where hypothesis testing indicate that a negative correlation with supervised modularity metric IRS as well as UDR is significant ($\alpha = 0.01$), even when considering Spearman correlation.

UDR is seen to share notable negative correlation with the nonlinear explicitness metrics. With UDR only considering similarity, the amount of information relative to the original data that is present in the representation does not influence the score assigned. As argued by [Higgins et al., 2017; Burgess et al., 2018], disentanglement in some of the models is enforced through reducing the capacity of the representation space, which could potentially impact the resulting explicitness. Perhaps more suprising is the low correlation shared by UDR with several of the modularity metrics, which is even negative in the case of Z-diff.

Also worth noting is that the explicitness metrics included here are all implemented using predictors trained to predict the ground truth factors, resulting in explicitness metrics that potentially favour to some degree representations where these factors are disentangled according to modularity and compactness, as suggested by the hypothesis at hand. This could account for some of the positive correlation shared by explicitness metrics with modularity and compactness metrics.

For representations from each of the 1800 models, the following 12 prediction accuracy scores are computed: Classification of object shape into classes square, heart and circle using logistic regression, gradient boosted classification and random forest classification, regression of object size using linear regression, gradient boosted regression and random forest regression and regression of object position using two different target representations, one given by the Cartesian coordinates of the position of the object and one given by the polar coordinates of the position, both predicted using linear regression, gradient boosting regression and random forest regression. Prediction of polar coordinates of position is included in order to observe behaviour of comparison of supervised metric scores and prediction scores that do not see labels of identical factors, where a nonlinear combination of a subset of factors is instead attempted predicted.

In order to compare level of disentanglement with predictor accuracy, 16×12 correlation coefficients are computed for each of the disentanglement metrics with respect to each of the prediction tasks. Assuming that increased disentanglement does indeed lead to increased predictor performance, one would expect to see pos-

itive correlation coefficients of some strength for all prediction tasks. Figure 5.6 shows the correlation coefficients for these models.

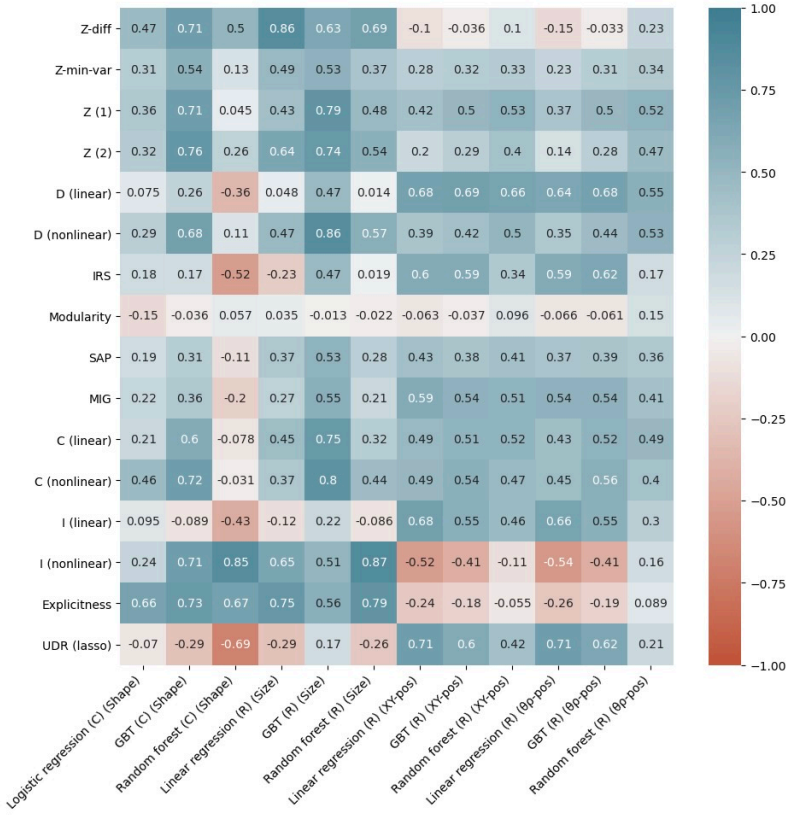


Figure 5.6: The correlation matrix shows correlation of accuracies of 12 prediction tasks (x-axis) with scores from 16 metrics (y-axis), calculated on representations from all models.

As seen in Figure 5.6, the supervised modularity and compactness metrics show positive correlation of varying strength with most of the prediction tasks, disregarding the Modularity metric, supporting the hypothesis that disentanglement does improve predictor performance. As the Modularity metric displays weak correlation with all prediction tasks as well as with all other metrics in Figure 5.5, this metric is not considered in discussion of the remaining experiments in this section. It is included in subsequent figures for completeness.

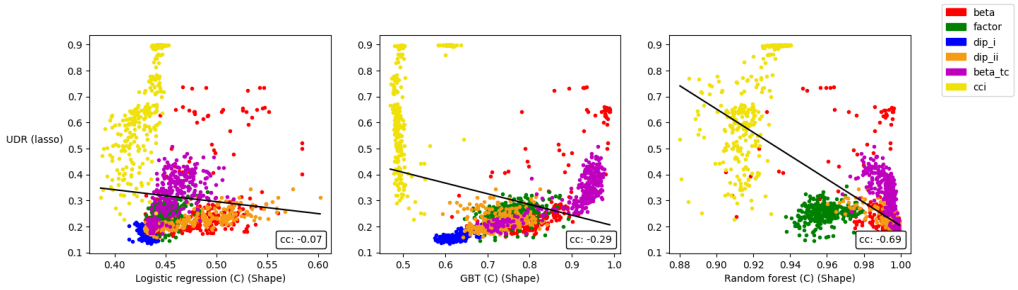
The supervised metrics are calculated on class labels of the factors shape and size as well as the Cartesian coordinates of the position. While the downstream regressors use factor values, all datapoints belonging to a factor class share the same value, giving a one-to-one mapping from label to value. Shared access to labels may influence correlation, particularly between predictor-based metrics and downstream tasks. However, position prediction using polar coordinates appears only slightly if at all less correlated with the supervised metrics than the prediction using the Cartesian coordinates, even in the case of linear regression, indicating

that disentanglement in general may be useful also for these prediction tasks.

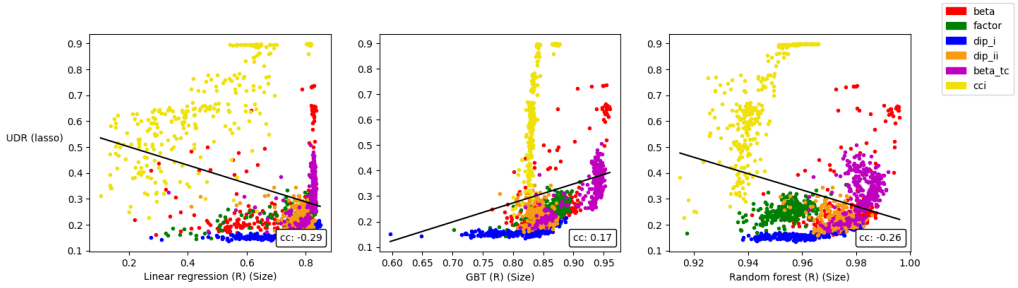
From Figure 5.6, nonlinear explicitness appears to be negatively correlated with position prediction as measured by both the nonlinear I and the Explicitness metrics. While prediction of position contributes to the explicitness scores, if in metric calculations the prediction accuracy of the remaining factors display greater variance, the resulting score will reflect that. As the representations have a fixed size, more information about some factors may potentially correspond to less information about other factors.

The unsupervised disentanglement metric UDR (bottom row in Figure 5.6) is particularly interesting, having no knowledge of factor labels. In this experiment, UDR scores appear well correlated with position prediction, both linear and nonlinear. However, prediction of shape and size correlates much weaker with these UDR scores, with most prediction scores even displaying notable negative correlation. Moreover, this is contradictory to most supervised metric scores.

Looking closer at the relationships between the downstream predictor scores and the UDR score, Figures 5.7 and 5.8 shows scatter plots of the UDR score with respect to prediction score for all 12 of the prediction tasks. β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE, β-TCVAE and CCI-VAE models are plotted in different colours.

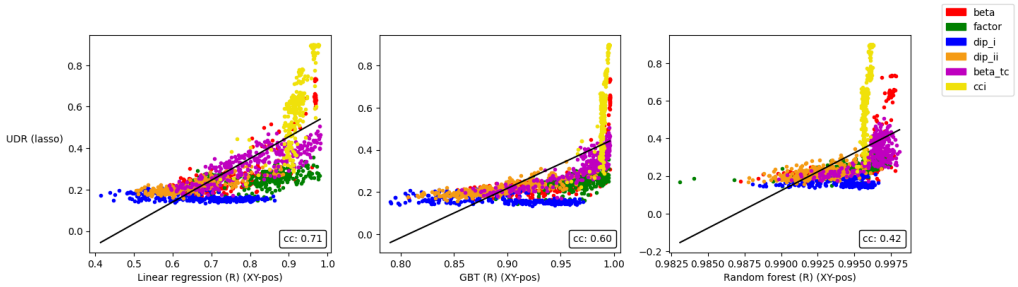


(a)

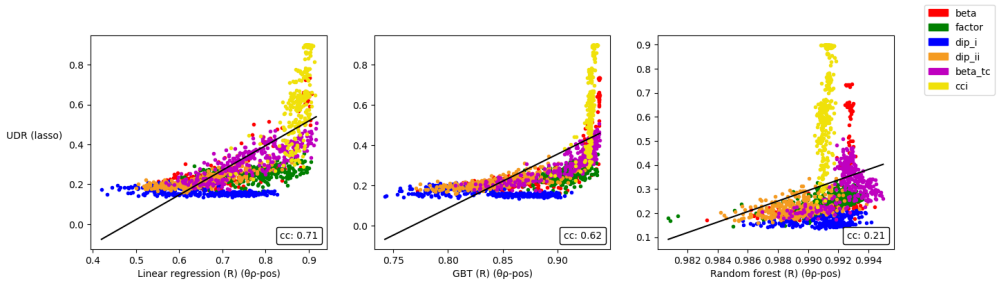


(b)

Figure 5.7: Scatter plots showing the relationship between UDR scores and prediction scores on model 0-1799, when predicting a) shape and b) size



(a)



(b)

Figure 5.8: Scatter plots showing the relationship between UDR scores and prediction scores on all models, when predicting a) Cartesian coordinate position and b) polar coordinate position

The plots in Figures 5.7 and 5.8 show that the CCI-VAE model scores in particular appear to be differently distributed, with generally higher UDR scores, and in the case of shape (Figure 5.7a) and size (Figure 5.7b) prediction, lower prediction scores. Additionally, while the UDR scores appear to cover a wide range compared to other models, most CCI-VAE prediction accuracies lie in a narrow range, particularly for the nonlinear predictors.

As discussed in Section 3.4, the UDR metric assigns scores based on the similarity of the model representations up to permutations and sign inverse, based on results in [Rolinek et al., 2019] that argue that VAE models that disentangle well tend to converge to similar representation.

The CCI-VAE, discussed in Section 3.4, uses a parameter C, which is increased during training, to gradually increase the capacity of the representations. At the beginning of training, very little information is allowed encoded in the representation, and the information most important in reconstruction of the input is prioritized. In [Burgess et al., 2018] it is shown that in the case of the dSprites dataset the factors describing object position are encoded first using a CCI-VAE model. Included from [Burgess et al., 2018], the plots in Figure 5.9 show position being encoded early in training when capacity is low, followed by size, shape and orientation respectively as capacity increases.

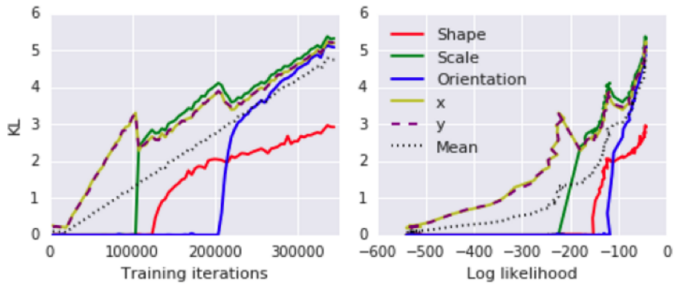


Figure 5.9: Figures from [Burgess et al., 2018]. As capacity is increased during training, the KL-divergence of each factor, that is $D_{\mathrm{KL}}(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathrm{v}_k)\|p(\mathbf{z}))$, is plotted with respect to training iterations (left) and reconstruction accuracy (right) for v_k corresponding to factors shape, size (scale), orientation, x and y.

From Figure 5.8, it is seen that all CCI models score well on predicting position, with low range despite these being scores on representation from 300 different models trained with 6 different maximum capacity C_{max} values (ranging from 5 to 100). This is the case even for linear regression of both Cartesian (Figure 5.8a) and polar (Figure 5.8b) coordinates, suggesting that position information is indeed easily accessible in these representations.

The CCI-VAE objective gradually allows new pieces of information to be encoded in decreasing order of importance as determined by reconstruction of the input, until max capacity is reached. The similarity of the CCI-VAE models as measured by the UDR metric may be partly due to enforcing such an order of importance when encoding information about the data, rather than similarity just relating to the level of disentanglement.

In order to further inspect the behaviour of the UDR scores, the metrics relationship with the other disentanglement metrics is similarly considered in detail. As seen in Figure 5.5, UDR does not show as strong a correlation with all metrics as would be expected, particularly some of the other modularity metrics. Figure 5.10 shows scatter plots of the UDR score against scores from the two versions of the predictor free modularity metric Z, as well as the predictor free compactness metric MIG.

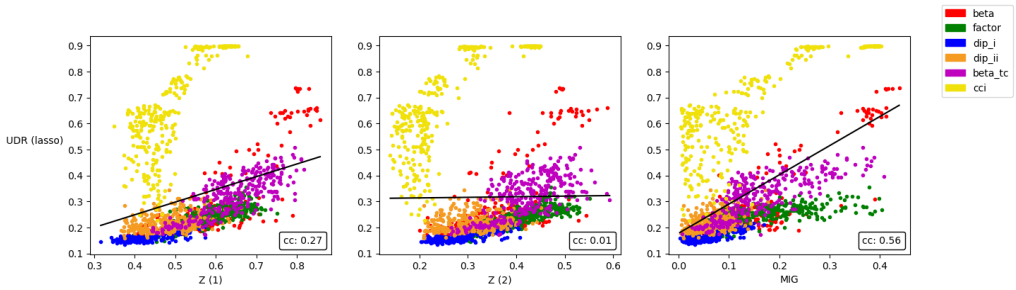


Figure 5.10: The figure shows scatter plots of UDR score (y-axis) against the modularity metrics Z (1) (left) and Z(2) (middle), and the compactness metric MIG (right).

Figure 5.10 indicates that UDR does indeed assign higher scores to CCI-VAE models than the other models for a given level of disentanglement as measured by the Z and MIG metrics. Pairwise scatter plots of UDR and all supervised metrics are shown in Appendix E, displaying the same pattern. This may indicate that the results from [Rolinek et al., 2019] utilized by the UDR metric is not entirely reversible, potentially resulting in a failure mode of this metric. While converging to similar representations is a consequence of enforcing disentanglement, converging to similar representations will not guarantee disentanglement.

Figure 5.11: UDR correlation with prediction tasks, calculated on CCI-VAE model representation

Figure 5.11 shows UDR correlation with predictors on representations from CCI-VAE models 1500-1799. When considering models trained within this framework, several predictors show strong correlation with unsupervised disentanglement measured by UDR, suggesting that UDR might still be useful if comparing models within this group. Noting again, as seen in Figure 5.8, that the accuracies obtained by most predictors trained on CCI-VAE model representations of the dSprites dataset occur within a small range compared to the other models, especially nonlinear predictors, contrasting the wide range of the UDR scores. Linear regression of size (Figure 5.7b left) is an exception, where increased UDR score corresponds to significant increase in prediction accuracy training on these model representations.

Identifying the CCI-VAE model representations to be distributed differently according to the UDR scores, the results presented in Figure 5.6 is reproduced discarding the CCI-VAE models. Figure 5.12 shows correlation between metrics and predictors on representations from models 0 - 1499.

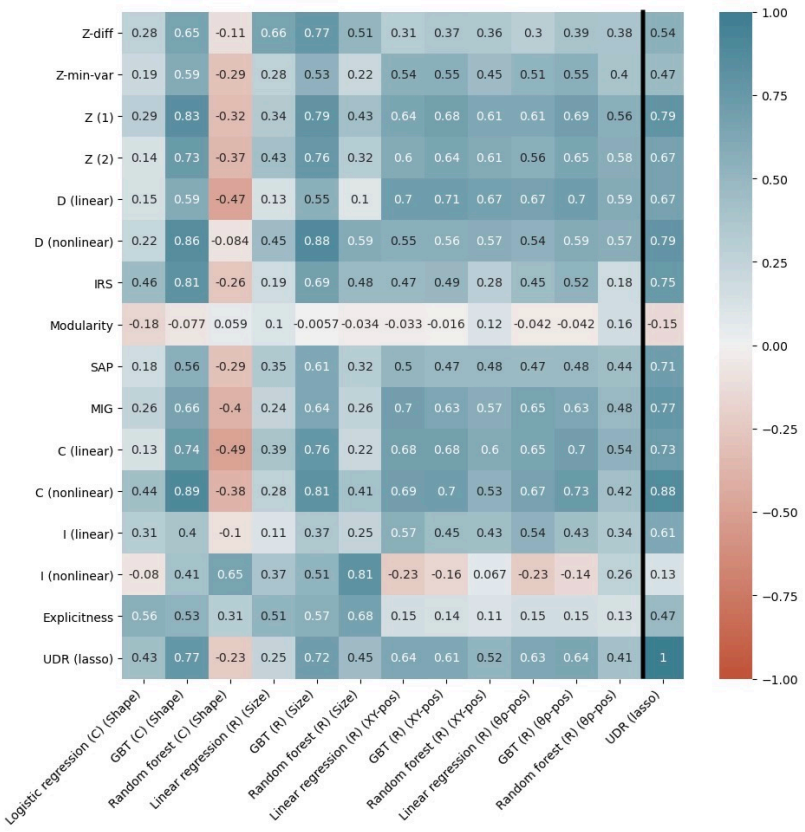


Figure 5.12: The correlation matrix shows correlation of accuracies of 12 downstream prediction tasks (x-axis) with scores from all metrics (y-axis), calculated on representations from models 0-1499

Figure 5.12 shows that UDR now appears in general better correlated with all prediction tasks, with only random forest classification of factor shape still displaying negative correlation with UDR. However, in these results this negative correlation appear to be more consistent with correlations shared with the predictor by the supervised metrics measuring modularity and compactness. Included in Figure 5.12 in the rightmost column are also the correlations of UDR with the supervised metrics. Compared to the results shown in Figure 5.5 considering all models, discarding the CCI-VAE also results in greater correlation between the UDR scores and the supervised scores, as indicated by the scatter plots in Figure 5.10.

It is also seen from Figure 5.12 that in general, the random forest predictors show lower correlation with the metrics than the gradient boosted predictors, both reporting nonlinear prediction accuracies on the same datasets. This can possibly be explained by the low range and high score of the random forest predictors as seen in Figures 5.7 and 5.8, not shared by the gradient boosted predictors. This discrepancy might follow from a higher capacity of the random forest predictors, leaving disentanglement to a greater extent irrelevant as enough data is provided.

To investigate disentanglement impact on these prediction tasks when limited data is available, the prediction accuracy scores are recalculated training the predictors on 1000 data samples. Figure 5.13 shows metric correlation with these predictor accuracies, on representations from models 0-1499.

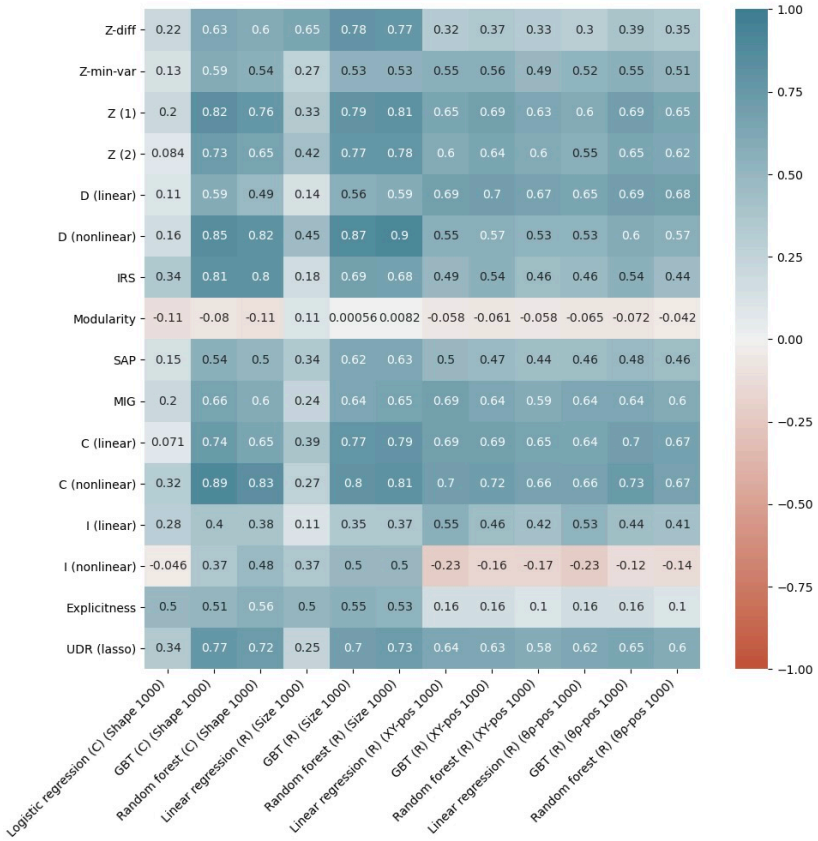


Figure 5.13: Metric score correlation with predictor accuracies when the predictors are trained on 1000 data samples, calculated on representations from models 0-1499.

In the results shown in Figure 5.13, performance of all prediciton tasks appear to benefit from disentanglement as measured by UDR as well as the supervised metrics. All nonlinear predictors calculated on limited sample sizes appear well correlated with level of unsupervised disentanglement as measured by UDR and the supervised modularity and compactness metrics. All linear predictors also show positive correlation, while for the shape and size prediction tasks these correlations are not as strong.

Looking closer at these two prediction tasks in particular, Figure 5.14 show UDR scores plotted against logistic regression of shape (left) and linear regression of size (right), when CCI-VAE models are not included and the predictors are trained using 1000 samples.

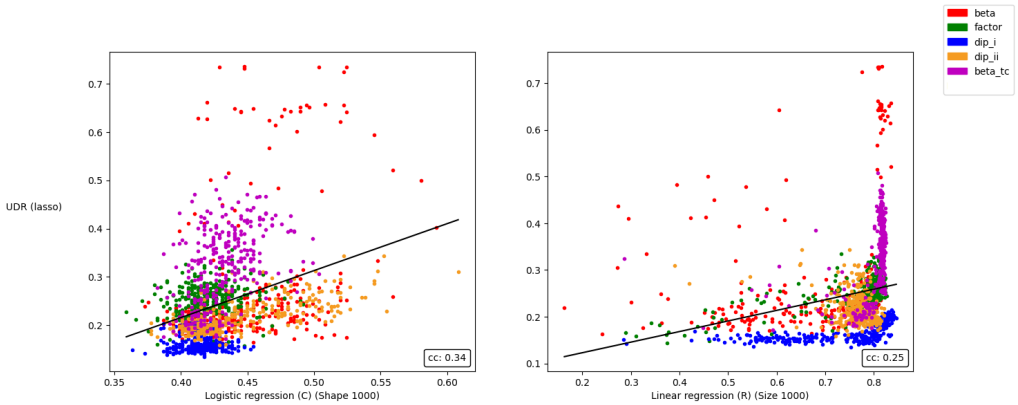


Figure 5.14: UDR scores plotted against accuracy of logistic regression of shape (left) and linear regression of size (right), for models 0-1499.

In the case of the linear shape classification, scores are generally low considering that accuracy of random classification is 1/3 when classifying objects into three classes. An increase in disentanglement does however appear related to a slight increase in prediction accuracy. In the case of linear size regression, accuracy is generally high, indicating that a high level of general disentanglement might not be required in order to solve this task on the data in question. In this case as well, some positive correlation is present.

When the targets of the prediction tasks considered are the generative factors also attempted disentangled, one would expect that disentanglement of the factor that is attempted predicted is particularly useful, especially in linear prediction tasks. Looking at the impact of disentanglement of each of the ground truth factors on the prediction tasks separately, Figure 5.15 shows correlation of prediction tasks on shape, size and position with disentanglement scores from metrics UDR, Z(1), Z(2) and MIG calculated on all factors, as well as scores from metrics Z(1), Z(2) and MIG calculated on shape, size, position and orientation separately.

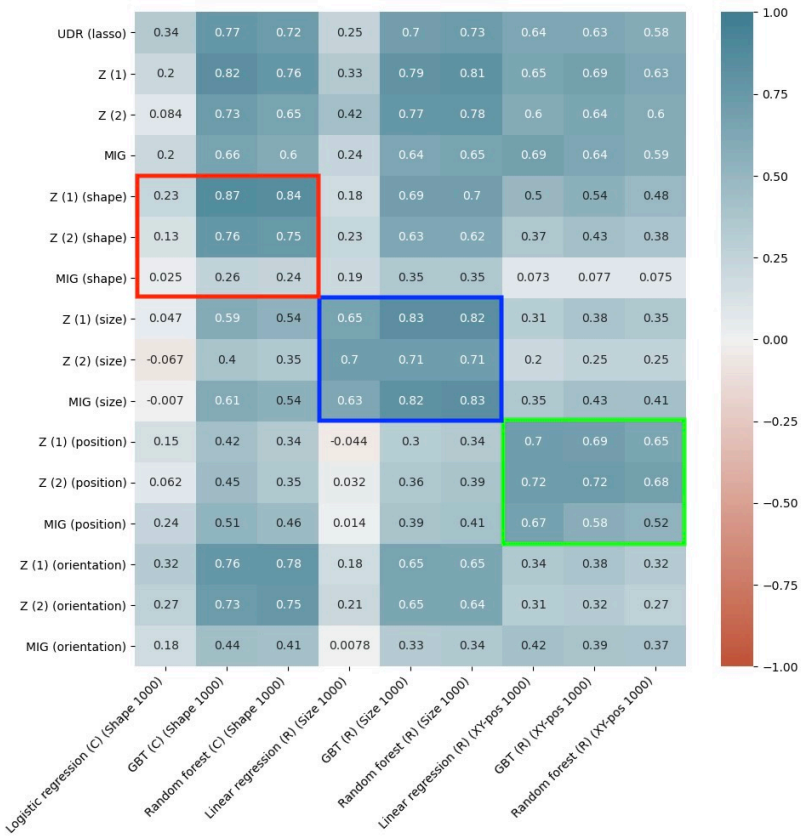


Figure 5.15: The correlation matrix show correlation of disentanglement of each of the factors with the prediction tasks where generative factors acts as targets. The metrics of the four top rows are calculated on all factors. For each predictor of generative factors, correlation of accuracy with disentanglement of the target factor is marked, shape by the red box, size by the blue box and position given by x and y coordinates by the green box.

In Figure 5.15, the red box shows correlation of shape classifiers with disentanglement of shape, the blue box shows correlation of size regressors with disentanglement of size and the green box shows correlation of Cartesian position regressors with disentanglement of position. The figure shows that while linear prediction of size is not as strongly correlated with overall disentanglement, the correlation is indeed as expected much stronger when considering disentanglement of size only (blue box). Note that neither the Z-metrics nor MIG use a predictor in assigning their scores, so the results are not entirely trivial. Similar results are found for position (green box).

However, the same tendency can not be found in the case of linear shape prediction (red box), which for the models in this experiment appear as related with disentanglement of the other factors, particularly the orientation factor.

A close to zero correlation is detected between compactness of shape (MIG) and logistic regression of shape. As shape is a categorical variable without order,

a compact encoding in one dimension of a continuously valued representation may not be the most efficient way to represent such a variable, even less so when a linear mapping to factor space is assumed. In general, a linear mapping may be to restrictive when recovering information about this factor, also when modularity is increased. Size and position are continuously valued factors, and would potentially benefit more from relevant information being both modularily and compactly encoded, also when assuming linearity.

While shape and orientation are statistically independent factors in the data distribution, shape prediction appears to benefit in particular from disentangling orientation, providing some support to the claim that disentanglement in general is a desired property of representations.

Chapter 6

Evaluation and Conclusion

6.1 Evaluation and discussion

The results presented in Section 5.3 provide support to the hypothesis that disentanglement is a desirable property in representations when solving downstream prediction tasks.

From Figure 5.5 it is seen that the scores of the supervised disentanglement metrics employed in these experiments for the most part appear to correlate well on the set of representations for the dSprites dataset from the VAE-based unsupervised disentangling models that form the basis of the experiment. These results indicate that the metrics considered to some extent succeed in capturing the properties they intend to, independently of model. The exception is the Modularity metric, which is disregarded in the following discussion.

As shown in Figure 5.10, the UDR metric is found to be distributed differently for the CCI-VAE models given disentanglement as measured by supervised metrics. However, when considering representations learned by the remaining models, UDR also correlates well with the supervised metrics, quantified by correlation coefficients in Figure 5.12. This provides further support to the measurements, as UDR is calculated without knowledge of ground truth generative factors. In general, the metrics are considered reliable when evaluated jointly in these experiments. However, the scores from metrics measuring the same representation do not in general agree on value of disentanglement, suggesting metrics are to some extent influenced by implementation technique. Appendix E.3 shows plots of average metric scores over models with the same hyperparameter settings that illustrate this, suggesting that scores should be interpreted with care.

Figures 5.6 and 5.12 show that the accuracy scores of several predictors are found to correlate well with disentanglement as measured by the disentanglement metrics. This includes the unsupervised UDR disentanglement score, when disregarding the CCI-VAE models. Additionally, the results presented in Figure 5.13 show that in the case of predictors of high capacity, disentanglement may be particularly useful when limited data is available.

While disentanglement is suggested useful in a holistic sense, when evaluated it is subdivided into distinct properties that a representations may possess, namely modularity, compactness and explicitness (linear or nonlinear). In the case of ex-

plicitness, a specific downstream task will benefit from receiving all information relevant in solving the task, motivating that general representations should retain as much information as possible when subsequent tasks are not known at the time of learning. Modularity and compactness are the properties that address the partitioning of information in reference to the generative factors, and these properties take on values in $[0, 1]$ given any level of explicitness > 0. The results presented in Figures 5.6, 5.12 and 5.13 show that predictor accuracy in general tend to improve with both increased modularity and compactness.

It is potentially more useful in some cases to evaluate explicitness and modularity/compactness separately and independently, focusing on metrics that measure one property only, such that impact of e.g. modularity may be evaluated for representations of a given level of explicitness and so forth.

The dSprites dataset is by design a very simple dataset, where samples are constructed from predefined factors without noise. Thus samples corresponding to all factor combinations appear exactly once in the dataset, ensuring both conceptual and statistical independence. While the results clearly suggest that disentanglement increase predictor accuracy, the low complexity of the data, as well as low complexity of the downstream tasks considered, may limit the generalisability of the results beyond synthetic data.

Even in the context of this simple dataset, Figure 5.15 provides some results indicating that the nature and complexity of factors should be kept in mind, suggesting compactness as a representation property may not be entirely beneficial with respect to generative factors of some complexity. The usefulness of compactness as a disentanglement criteria has been discussed [Higgins et al., 2018; Ridgeway and Mozer, 2018], some arguing that it might be to restrictive when dealing with real world datasets where generative factors may appear increasingly complex. Results from Figure 5.15 indicate compactness may indeed be less favourable when representations are intended to solve subsequent tasks. However, compactness may still be considered useful in other situations, e.g. if representations are required to be interpretable.

Figure 5.15 suggest that disentanglement of some factors corresponds to improved performance in predictions of other factors. While noting that unsupervised models encourage disentanglement in general and thus disentanglement of different factors would not be expected to be entirely independent of each other, some factors display particularly strong correlation with unrelated prediction tasks. Such relationships further indicate that disentanglement in general is useful and that disentangling as many factors as possible is desirable regardless of prediction task.

In the experiments, the Pearson correlation coefficient is considered both when quantifying relationships between metrics, and when comparing metrics with predictor performance. In the case of metric evaluation, this is justified by the assumption that the metrics capture true level of disentanglement independently of model type and other representation properties. For consistency, Pearson correlation coefficients are considered when quantifying all relationships in Chapter 5. However, when comparing disentanglement with predictor performance, assuming a linear

relationship may be too restrictive. What is suggested by the hypothesis, is that increasing disentanglement should correspond to improving predictor performance, however, it may not be reasonable to expect this relationship to be linear. Thus, one could argue that strength of monotonically increasing relationships as measured by rank correlation, quantified by Spearman correlation coefficients, would be a more suitable measure when evaluating disentanglement impact on predictor performance. This would be supported by the data distributions seen in Figures 5.7 and 5.8.

Appendix E.4 contains Spearman correlation coefficient equivalents of the correlation matrices shown in Figures 5.6, 5.12, 5.13 and 5.15, that compare disentanglement scores with prediction accuracy scores, quantified using the Spearman correlation coefficient. On these data it does in general appear like overall results indicate similar findings when compared to Pearson coefficients, with Spearman sometimes indicating a stronger positive relationship between disentanglement and predictor performance.

Moreover, in the experiments presented in Chapter 5, the assumption that metrics are independent across models is extended to apply to relationship between disentanglement scores and predictor performance, as the group of models that is considered includes unsupervised disentangling models trained from several different objective functions. In particular, the property measured by a metric, whether it is modularity, compactness, explicitness or a combination, is in each comparison assumed linearly related to prediction accuracy on representations form the chosen set of models. This assumption might not be justified, as model objectives may otherwise influence representations affecting the results. More reliable results may be obtained considering models of given objective functions separately. It is seen in the case of the UDR metric in Figure 5.10 that the assumption does not always hold among metrics either, which could be argued is more problematic, as it causes ambiguity as to what exactly is reflected by the metric.

That correlations of some strength is in fact detected even when including models trained from several different objective functions, may be seen as further support of disentanglement as a desireable property. In the context as defined by the experiment presented in Chapter 5, disentanglement appears comparably determinative of predictor success across all models.

6.2 Contributions

The main contributions of this thesis are presented here.

The first research question posed in Section 1.2 asks whether disentanglement evaluation techniques can be trusted to provide reliable quantification of disentanglement. In the experiments presented in Section 5.3, several supervised disentanglement metrics are observed evaluating disentanglement scores that conform to some extent, and while demonstrating ambiguity in exact values, a subset of the metrics are considered reliable in reflecting increased/decreased level of true disentanglement as scores increase/decrease. As it remains unclear which metrics, if, any, are more accurate, overall trends across all metrics are considered jointly when

evaluating disentanglement.

The unsupervised disentanglement metric UDR is observed demonstrating scores that appear to generally comply with the supervised scores within model groups, but not unanimously across model groups. While this is considered as further supporting the reliability of metrics, some lack of clarity remains as to how scores should be interpreted.

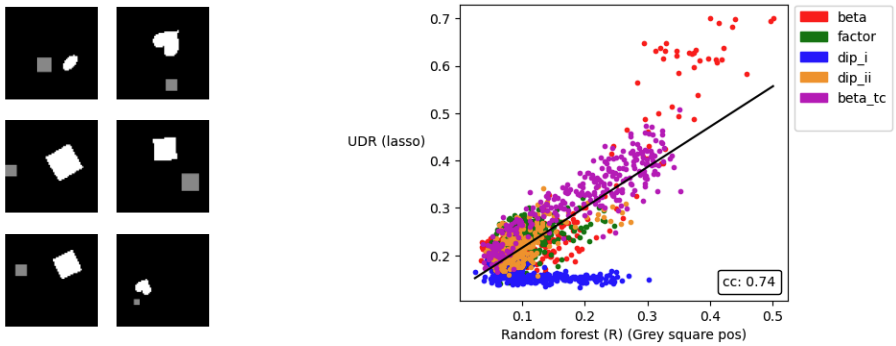
The second research question asks whether unsupervised disentanglement can be seen to improve downstream performance on simple prediction tasks. In the presented experiment, increased disentanglement as measured by the metrics considered reliable, is found related to improved prediction performance of the defined downstream tasks, both according to supervised and unsupervised disentanglement scores. Prediction accuracy when trained on limited data is also shown to benefit from disentanglement.

Additionally, results are presented that indicate further potential in disentanglement when datasets and ground truth generative factors grow increasingly complex.

6.3 Future work

The experiments performed here are simple both in terms of dataset complexity and in terms of downstream tasks complexity. While the results do indicate that disentanglement of a factor is useful beyond recovering the factor itself, more work should be done exploring this.

Figure 6.1a shows examples from the dSprites dataset where each sample is modified by inserting a grey square at random position and of random size. From representations encoded for samples as shown in Figure 6.1a using the models pretrained on the original dSprites dataset without additional training, the plot in Figure 6.1b shows the UDR score with respect to accuracy of prediction of the position of the grey square, both computed on the new representations.



(a) (b)

Figure 6.1: Modified dSprites data, where grey squares are added at random position, shown in a), is encoded using models 0-1499 and UDR score (y-axis) is plotted against regression accuracy of position of the grey square (x-axis), shown in b).

Figure 6.1 indicate that the accuracy of predicting the position of the grey square from the representations seem to benefit from disentanglement according to the UDR score. As the VAE models has not seen the modified samples during training, these findings may be interpreted as indicative of robustness of the models when introduced to out-of-distribution samples.

While these are only initial findings, where UDR calculation and training of the random forest regressor is performed on a representation subset of size 30 000, and where regression accuracy is generally low and range from 0.02 to 0.50, Figure 6.1 further motivates more research on the advantages of disentanglement reflected in model robustness.

Moreover, future work should look at disentanglement of generative factors from real world data. Models trained to do unsupervised disentanglement provides disentanglement in a general context, enabling use of any datasets. Performing similar experiments on datasets of real images, and even text-based and sequential data, would also allow for more interesting downstream tasks, possibly prediction of complex data characteristics where several generative factors would interact in deciding the outcome, as opposed to predicting simple factors/factor combinations.

While most metrics measuring disentanglement require access to at least some labeled ground truth generative factors in order to give a (partial) disentanglement score, the unsupervised UDR metric additionally allows for measuring complete modularity without ground truth knowledge of the generative factors. As UDR is shown here not to agree with the supervised metrics across model groups, more research might be needed in order to establish when UDR and the supervised metrics individually can be trusted to return reliable disentanglement scores.

As discussed in Section 6.1, it might be argued more reasonable that correlation between disentanglement and prediction accuracy should be considered for models of specific types independently. With correlation between UDR modularity and linear classification of object shape on a dataset of size 1000 as an example, the Pearson correlation coefficient is calculated to be 0.34 across all 1500 β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE and β-TCVAE models, as seen in Figures 5.13 and 5.14. Figure 6.2 shows scatter plots of the linear classification accuracy of factor shape (trained on 1000 data samples) with respect to the UDR score for each model type β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE, β-TCVAE and CCI-VAE separately. Within each plot, the models trained from different hyperparameter values are plotted in different colours.

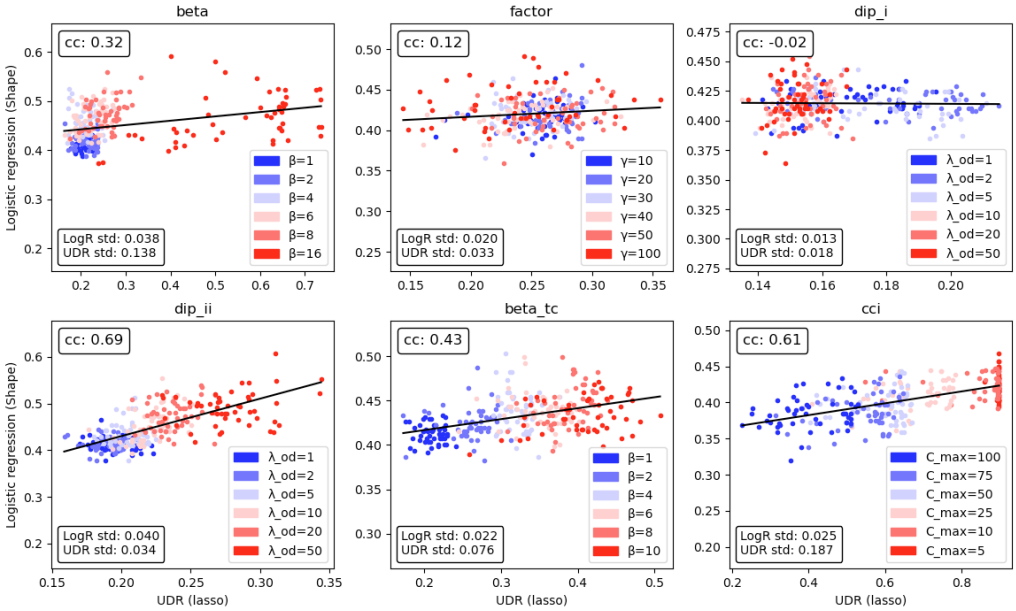


Figure 6.2: The plots show how the accuracy of linear classification of shape using logistic regression is related to UDR modularity score within the 300 models of each model type β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE, β-TCVAE and CCI-VAE. In each plot, models trained with the same hyperparameters, differing in weight initialization only, are plotted with the same colour. Correlation coefficient is included in the upper left corner in each plot. The standard deviation of the predictor accuracy (LogR) and UDR scores are included in the lower left cornes.

Figure 6.2 shows that within several model groups, strength of correlation is indeed stronger, with DIP-II-VAE, β-TC-VAE and CCI-VAE models displaying correlations of 0.69, 0.43 and 0.61 respectively. Contrarily, the Factor-VAE and DIP-I-VAE groups display weak to no correlation, noting however that these are the groups of lowest standard deviation among both prediction accuracy and UDR score.

Generally low predictor accuracies displaying low spread within model groups in this particular example, which is also the case of the UDR score for some groups, suggests these plots might not be reliable in determining a disentanglement advantage, or lack thereof. Still, Figure 6.2 indicate when compared to Figure 5.14 that taking model objective into account when evaluating benefit of disentanglement might be advisable, possibly enabling further understanding of when disentanglement is most useful, increasing reliability. More work is thus needed as to determine impact of disentanglement (modularity, compactness, explicitness) under the influence of specific model objectives.

Bibliography

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2019). Deep variational information bottleneck.

Bengio, Y., Courville, A., and Vincent, P. (2014). Representation learning: A review and new perspectives. *arXiv:1206.5538 [cs.LG]*.

Bjøru, A. R. (2020). Disentangled representation learning. Unpublished.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. (2018). Understanding disentangling in β-vae.

Chen, R. T. Q., Li, X., Grosse, R., and Duvenaud, D. (2019). Isolating sources of disentanglement in variational autoencoders.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets.

Duan, S., Matthey, L., Saraiva, A., Watters, N., Burgess, C. P., Lerchner, A., and Higgins, I. (2020). Unsupervised model selection for variational disentangled representation learning.

Eastwood, C. and Williams, C. K. I. (2018). A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. (2018). Towards a definition of disentangled representations. *arXiv:1812.02230 [cs.LG]*.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

Hinton, G. E. (1986). Learning distributed representations of concepts. *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, pages 1–12.

Ilse, M., Tomczak, J. M., Louizos, C., and Welling, M. (2019). Diva: Domain invariant variational autoencoders.

Kim, H. and Mnih, A. (2019). Disentangling by factorising.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *arXiv:1312.6114 [stat.ML]*.

Kulkarni, T. D., Whitney, W., Kohli, P., and Tenenbaum, J. B. (2015). Deep convolutional inverse graphics network.

Kumar, A., Sattigeri, P., and Balakrishnan, A. (2018). Variational inference of disentangled latent concepts from unlabeled observations.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444. `https://doi.org/10.1038/nature14539`.

Locatello, F., Bauer, S., Mario Lucic, G. R., Gelly, S., Schölkopf, B., and Bachem, O. (2019). Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv:1811.12359 [cs.LG]*.

Louizos, C., Swersky, K., Li, Y., Welling, M., and Zemel, R. (2017). The variational fair autoencoder.

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. (2017). dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/.

Muandet, K., Balduzzi, D., and Schölkopf, B. (2013). Domain generalization via invariant feature representation. *arXiv:1301.2115 [stat.ML]*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rajaraman, S., Antani, S., Poostchi, M., Silamut, K., Hossain, M., Maude, R., Jaeger, S., and Thoma, G. (2018). Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv:1401.4082 [stat.ML]*.

Ridgeway, K. and Mozer, M. C. (2018). Learning deep disentangled embeddings with the f-statistic loss.

Rolinek, M., Zietlow, D., and Martius, G. (2019). Variational autoencoders pursue pca directions (by accident).

Sepliarskaia, A., Kiseleva, J., and de Rijke, M. (2021). How to not measure disentanglement.

Själander, M., Jahre, M., Tufte, G., and Reissmann, N. (2019). EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure.

Suter, R., Miladinović, D., Schölkopf, B., and Bauer, S. (2019). Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method.

Zaidi, J., Boilard, J., Gagnon, G., and Carbonneau, M.-A. (2021). Measuring disentanglement: A review of metrics.

Appendix A

Minibatch Weighted Sampling

Evaluating the Total Correlation $D_{\mathrm{KL}}(q(\mathbf{z})\|\bar{q}(\mathbf{z}))$, requires evaluation of $q(\mathbf{z})$, where $q(\mathbf{z}) = \sum_{i=1}^{n} q(\mathbf{z}|\boldsymbol{x}^{(i)})\hat{p}_{\mathbb{X}}(\boldsymbol{x}^{(i)}) = \mathbb{E}_{\hat{p}_{\mathbb{X}}}[q(\mathbf{z}|\mathbf{x})]$. While this depends on the enitre dataset $\mathbb{X}$, for training it is preferable to be able to approximate it on a minibatch of data samples. Chen et al. [2019] introduce Minibatch Weighted Sampling (MWS), a method to stochastically estimate $\mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})]$ using the minibatch estimator:

$$\mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] = \frac{1}{m}\sum_{i=1}^{m}\left[\log \frac{1}{mn}\sum_{j=1}^{m} q(\mathbf{z}(\boldsymbol{x}^{(i)})|\boldsymbol{x}^{(j)})\right] \tag{A.1}$$

A naive Monte Carlo approximation using a minibatch of samples from $\hat{p}_{\mathbb{X}}$ would be likely to underestimate $q(\mathbf{z})$, explained in [Chen et al., 2019] by arguing that for a sample $\boldsymbol{z}$, if it is a sample from $q(\mathbf{z}|\mathbf{x} = \boldsymbol{x})$, then $q(\mathbf{z} = \boldsymbol{z}|\mathbf{x} = \boldsymbol{x})$ will be large, however for a randomly sampled $\boldsymbol{x}$, $q(\mathbf{z} = \boldsymbol{z}|\mathbf{x} = \boldsymbol{x})$ will be close to 0.

A better estimate can be obatined by weighting the probabilities appropriately, as seen in the estimator in Equation (A.1), inspired by Importance sampling [Chen et al., 2019]. By the same argument, a similar estimator can also be constructed for $\bar{q}(\mathbf{z}) = \prod_j q(\mathrm{z}_j)$. The estimator is biased, and its expectation is a lower bound [Chen et al., 2019].

This section contains a derivation of the estimator in Equation (A.1). This derivation is an extended version of the derivation presented in Appendix C.1 in [Chen et al., 2019]. Table A.1 lists the concepts used in the derivation.

Notation	Description
$\beta_m = \{\mathbf{x}^{(i)}\}^m$	A minibatch of m datapoints $\mathbf{x}^{(i)}$, sampled i.i.d. from $\hat{p}_{\mathbb{X}}$
$\beta_{m,\mathbf{x}} = \{\mathbf{x}^{(i)}\}^{(m-1)} \cup \{\mathbf{x}\}$	A minibatch of $m-1$ datapoints $\mathbf{x}^{(i)}$, sampled i.i.d. from $\hat{p}_{\mathbb{X}}$, and a fixed sample $\mathbf{x}$
Ω_{β_m}	The set of all size m minibatches, $\lvert\Omega_{\beta_m}\rvert = n^m$
$\Omega_{\beta_{m,\mathbf{x}}}$	The set of all size m minibatches with one sample fixed to be $\mathbf{x}$, $\lvert\Omega_{\beta_{m,\mathbf{x}}}\rvert = n^{(m-1)}$, $\Omega_{\beta_{m,\mathbf{x}}} \subset \Omega_{\beta_m}$
$p(\beta_m)$	Probability distribution over β_m, $p(\beta_m) = (\frac{1}{n})^m$
$r(\beta_{m,\mathbf{x}})$	Probability distribution over $\beta_{m,\mathbf{x}}$, $r(\beta_{m,\mathbf{x}}) = (\frac{1}{n})^{(m-1)}$
$\mathbf{z}(\boldsymbol{x}^{(i)})$	A sample from $q(\mathbf{z}\lvert\boldsymbol{x}^{(i)})$

Table A.1: Notation describing concepts used to derive the MWS estimator.

Derivation of the MWS estimator:

$$
\begin{aligned}
&\mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \\
&= \int_{\mathbf{z}} [q(\boldsymbol{z}) \log q(\boldsymbol{z})] d\boldsymbol{z} \\
&= \int_{\mathbf{z}} [q(\boldsymbol{z}) \log q(\boldsymbol{z})] (\sum_{\mathbf{x}} [q(\boldsymbol{x}|\boldsymbol{z})]) d\boldsymbol{z} && \Big| \quad \sum_{\mathbf{x}} [q(\boldsymbol{x}|\mathbf{z}=\boldsymbol{z})] = 1 \\
&= \int_{\mathbf{z}} (\sum_{\mathbf{x}} [q(\boldsymbol{x}|\boldsymbol{z}) q(\boldsymbol{z}) \log q(\boldsymbol{z})]) d\boldsymbol{z} \\
&= \int_{\mathbf{z}} \sum_{\mathbf{x}} [q(\boldsymbol{z}, \boldsymbol{x}) \log q(\boldsymbol{z})] d\boldsymbol{z} \\
&= \mathbb{E}_{q(\mathbf{z},\mathbf{x})}[\log q(\mathbf{z})] \\
&= \mathbb{E}_{q(\mathbf{z},\mathbf{x})}[\log \mathbb{E}_{\mathbf{x}' \sim \hat{p}_{\mathbb{X}}}[q(\mathbf{z}|\mathbf{x}')]] && \Big| \quad q(\mathbf{z}) = \mathbb{E}_{\hat{p}_{\mathbb{X}}}[q(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{q(\mathbf{z},\mathbf{x})}[\log \mathbb{E}_{p(\beta_m)}[\frac{1}{m} \sum_{i=1}^{m} q(\mathbf{z}|\mathbf{x}^{(i)})]] && \Big| \quad \text{Monte Carlo} \\
&= \mathbb{E}_{q(\mathbf{z},\mathbf{x})}[\log \sum_{\Omega_{\beta_m}} p(\beta_m) [\frac{1}{m} \sum_{i=1}^{m} q(\mathbf{z}|\mathbf{x}^{(i)})]] \\
&= \mathbb{E}_{q(\mathbf{z},\mathbf{x})}[\log \sum_{\Omega_{\beta_m}} \frac{p(\beta_m) r(\beta_{m,\mathbf{x}})}{r(\beta_{m,\mathbf{x}})} [\frac{1}{m} \sum_{i=1}^{m} q(\mathbf{z}|\mathbf{x}^{(i)})]] && \Big| \quad \text{Importance sampling} \\
&\geq \mathbb{E}_{q(\mathbf{z},\mathbf{x})}[\log \sum_{\Omega_{\beta_{m,\mathbf{x}}}} \frac{p(\beta_m) r(\beta_{m,\mathbf{x}})}{r(\beta_{m,\mathbf{x}})} [\frac{1}{m} \sum_{i=1}^{m} q(\mathbf{z}|\mathbf{x}^{(i)})]] && \Big| \quad \Omega_{\beta_{m,\mathbf{x}}} \subset \Omega_{\beta_m} \\
&= \mathbb{E}_{q(\mathbf{z},\mathbf{x})}[\log \mathbb{E}_{r(\beta_{m,\mathbf{x}})} \frac{p(\beta_m)}{r(\beta_{m,\mathbf{x}})} [\frac{1}{m} \sum_{i=1}^{m} q(\mathbf{z}|\mathbf{x}^{(i)})]] \\
&= \mathbb{E}_{q(\mathbf{z},\mathbf{x})}[\log \mathbb{E}_{r(\beta_{m,\mathbf{x}})} [\frac{1}{nm} \sum_{i=1}^{m} q(\mathbf{z}|\mathbf{x}^{(i)})]] && \Big| \quad \frac{p(\beta_m)}{r(\beta_{m,\mathbf{x}})} = \frac{1}{n} \quad \text{(Importance weights)} \\
&= \mathbb{E}_{\hat{p}_{\mathbb{X}} q(\mathbf{z}|\mathbf{x})}[\log \mathbb{E}_{r(\beta_{m,\mathbf{x}})} [\frac{1}{nm} \sum_{i=1}^{m} q(\mathbf{z}|\mathbf{x}^{(i)})]]
\end{aligned}
$$

For training on a batch $\bar{\beta}_m, r(\bar{\beta}_{m,\mathbf{x}}) = 1$

$$
\begin{aligned}
&\mathbb{E}_{\hat{p}_{\bar{\beta}_m}}[\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log[\frac{1}{nm} \sum_{i=1}^{m} q(\mathbf{z}|\mathbf{x}^{(i)})]]] \\
&\approx \frac{1}{m} \sum_{j=1}^{m} [\frac{1}{L} \sum_{k=1}^{L} \log \frac{1}{nm} \sum_{i=1}^{m} q([\mathbf{z}(\boldsymbol{x}^{(j)})]^{(k)}|\boldsymbol{x}^{(i)})] \\
&\approx \frac{1}{m} \sum_{j=1}^{m} [\log \frac{1}{nm} \sum_{i=1}^{m} q(\mathbf{z}(\boldsymbol{x}^{(j)})|\boldsymbol{x}^{(i)})] && \Big| \quad L = 1
\end{aligned}
$$

Appendix B

The dSprites dataset

The dataset used in the experiment presented in Chapter 5 is the dSprites dataset [Matthey et al., 2017]. The dataset is generated from 5 factors of variation, listed in Table B.1, with examples shown in Figures B.1 - B.5.

	Factor	Classes	Values
0	Shape	3	Square, ellipse, heart
1	Size	6	$[0.5, 1.0]$
2	Orientation	40	$[0.0, 2\pi]$
3	Position X	32	$[0.0, 1.0]$
4	Position Y	32	$[0.0, 1.0]$

Table B.1: The factors of variation in the dSprites dataset.

Figure B.1: Factor 0 - shape

Figure B.2: Factor 1 - size

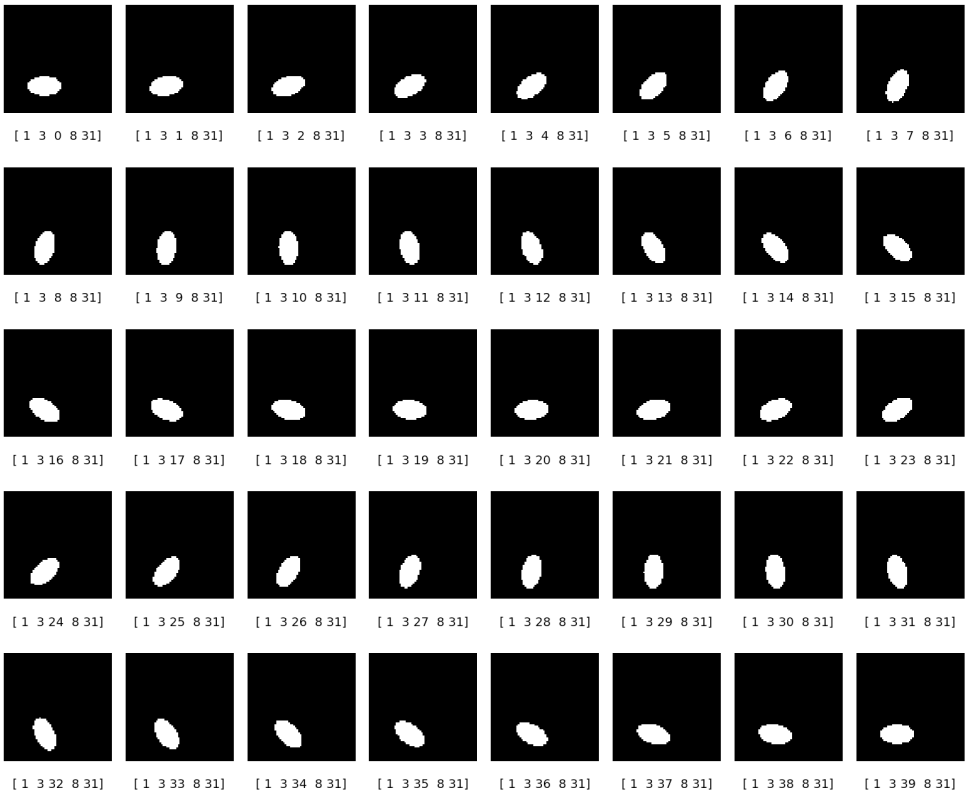


Figure B.3: Factor 2 - orientation

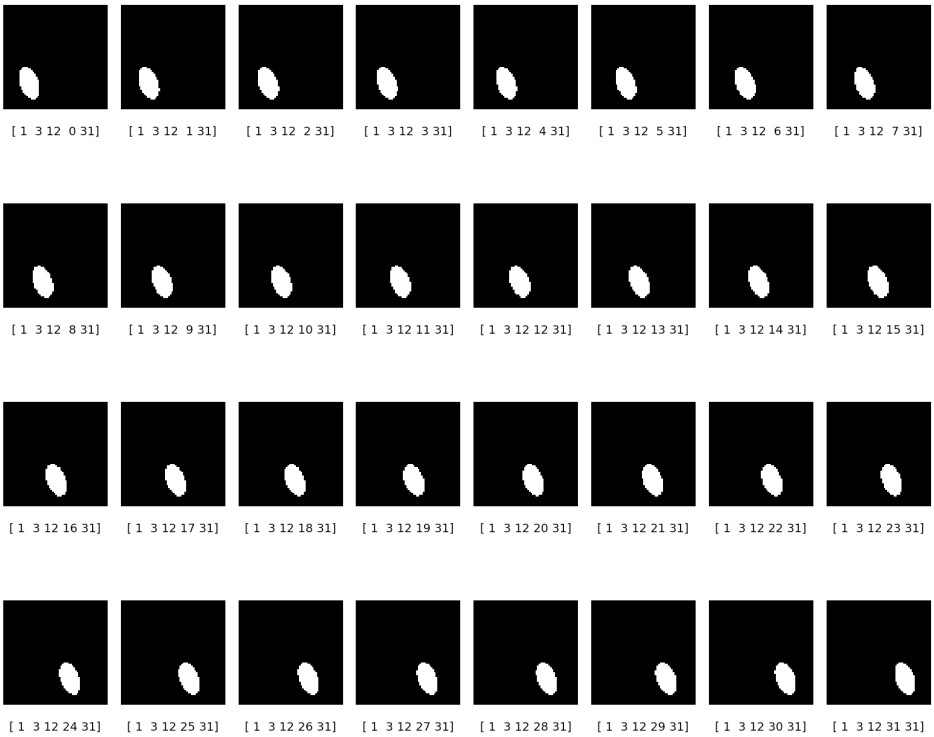


Figure B.4: Factor 3 - position X

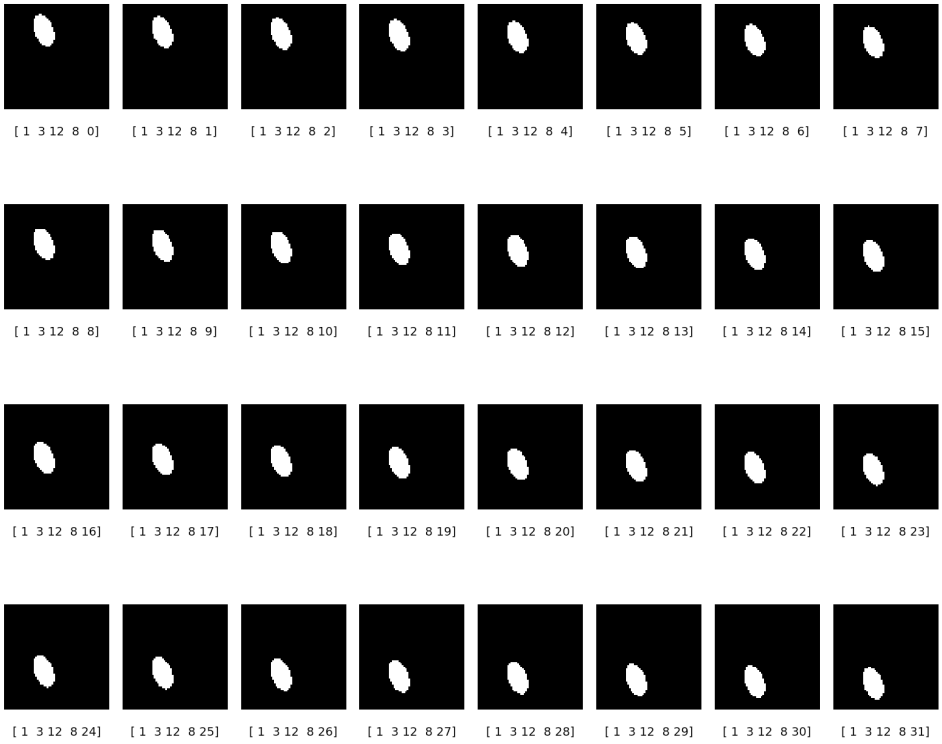


Figure B.5: Factor 4 - position Y

Appendix C

Metric convergence

Figures C.1 and C.2 shows metric scores plotted against dataset size, calculated on representations learned by model 0 from the DisentanglementLib [Locatello et al., 2019].

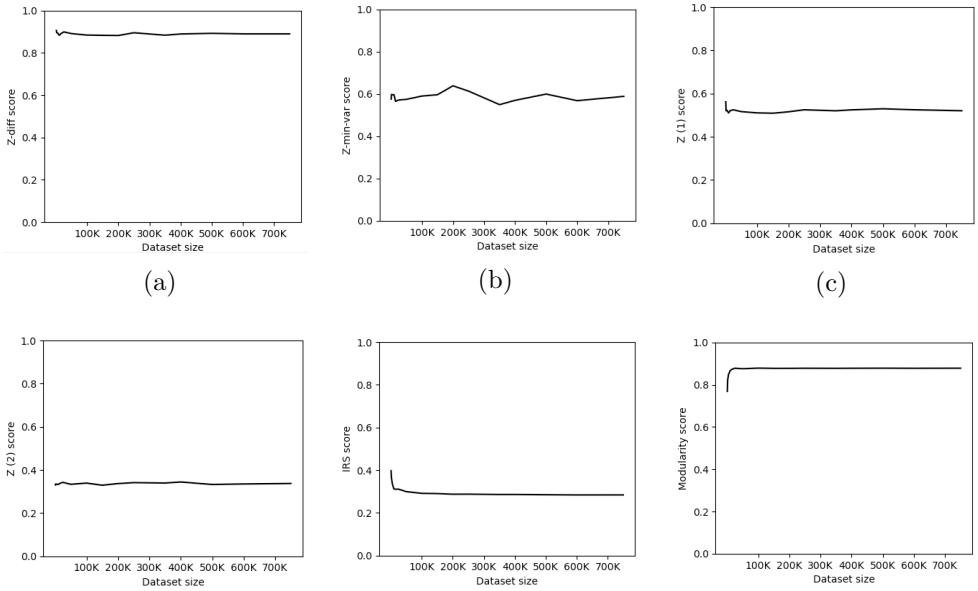


Figure C.1: Convergence of metrics a) Z-diff, b) Z-min-var, c) Z (1), d) Z (2), e) IRS, f) Modularity.

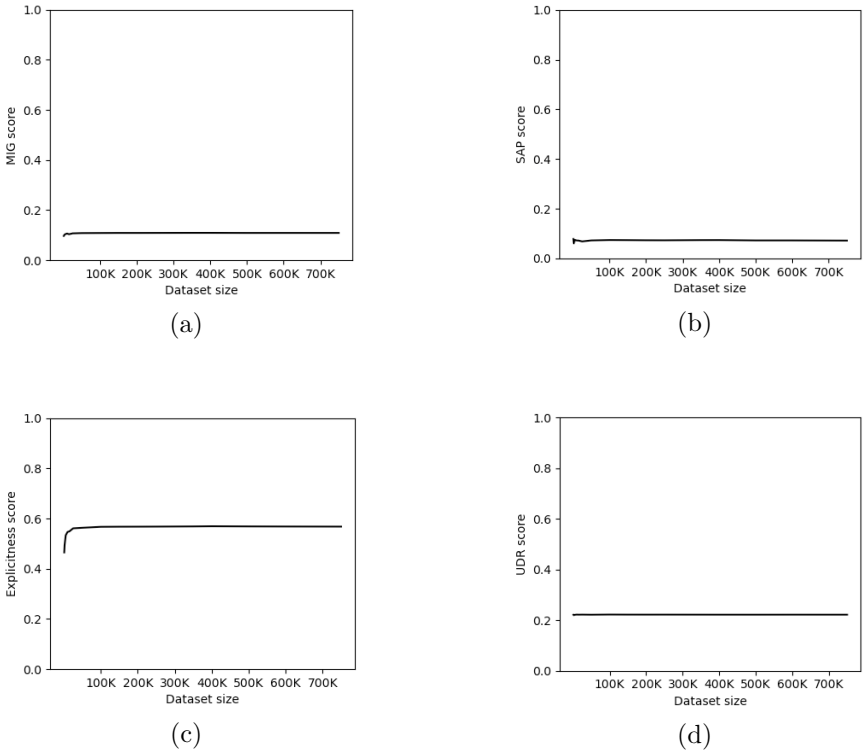


Figure C.2: Convergence of metrics a) MIG, b) SAP, c) Explicitness, d) UDR

Appendix D

DisentanglementLib models

The models used in the experiments are pretrained models from the DisentanglementLib [Locatello et al., 2019]. Model architecture is presented in Section D.1, and model hyperparameters are presented in Section D.2.

D.1 Model Architecture

All models have the same encoder and decoder architecture, shown in Tables D.1 and D.2. Factor-VAE discriminator architecture is shown in Table D.3.

Variational encoder
Input: $64 \times 64 \times 1$
Conv2 - 32 channels, 4×4 kernels, 2×2 strides, ReLU
Conv2 - 32 channels, 4×4 kernels, 2×2 strides, ReLU
Conv2 - 64 channels, 4×4 kernels, 2×2 strides, ReLU
Conv2 - 64 channels, 4×4 kernels, 2×2 strides, ReLU
FC - 256, ReLU
$2 \times$ FC - 10

Table D.1: The layers of the variational encoder.

Decoder
Input: 10
FC - 256, ReLU
FC - 1024 to $4 \times 4 \times 64$, ReLU
Upconv2 - 64 channels, 4×4 kernels, 2×2 strides, ReLU
Upconv2 - 32 channels, 4×4 kernels, 2×2 strides, ReLU
Upconv2 - 32 channels, 4×4 kernels, 2×2 strides, ReLU
Upconv2 - 1 channel, 4×4 kernels, 2×2 strides

Table D.2: The layers of the decoder.

Discriminator (Factor-VAE)
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 D.3: The layers of the Factor-VAE discriminator

D.2 Model hyperparameters

All variational encoders and decoders used in the experiments are trained from the hyperparameters shown in Table D.4. Factor-VAE discriminator hyperparameters are shown in Table D.5

Parameter	Value
Batch size	64
Optimizer	Adam
Adam, beta1	0.9
Adam, beta2	0.999
Adam, epsilon	1e-8
Adam, learning rate	0.0001
Training steps	300 000

Table D.4: The hyperparameters of the variational encoders and decoders

Parameter	Value
Batch size	64
Optimizer	Adam
Adam, beta1	0.5
Adam, beta2	0.9
Adam, epsilon	1e-8
Adam, learning rate	0.0001

Table D.5: The hyperparameters of the Factor-VAE discriminator

Appendix E

Results

E.1 Modularity

In Section 5.3, Modularity is seen to assign scores of particularly low correlation with all other metrics. Here, pairwise comparison of Modularity with all other metrics is detailed, considering both Pearson and Spearman correlations. Each row shows correlation coefficients (r) along with p-values for a hypothesis test where the null hypothesis is no correlation. Under the assumption of normally distributed scores and measuring linear relationships (Pearson), several metrics fail to show any correlation with 99% confidence, including modularity metrics Z-diff, Z (1) and linear and nonlinear D. Others, inlcuding the IRS and UDR metric, show significant ($\alpha = 0.01$) negative correlation, although correlation is still weak. Spearman correlation coefficients and p-values in Figure E.1 shows similar results when relaxing the assumptions of linearity of relationship and normally distributes scores.

	pearson r	pearson p	spearman r	spearman p
Z-diff	0.04	0.06	0.07	<0.01
Z-min-var	0.14	<0.01	0.13	<0.01
Z (1)	0.04	0.11	0.07	<0.01
Z (2)	0.07	<0.01	0.09	<0.01
D (linear)	0.06	0.02	0.08	<0.01
D (nonlinear)	-0	0.96	0.03	0.14
IRS	-0.16	<0.01	-0.07	<0.01
SAP	-0.03	0.17	0.09	<0.01
MIG	-0.02	0.38	0.06	0.02
C (linear)	-0.05	0.03	-0.05	0.05
C (nonlinear)	-0.14	<0.01	-0.12	<0.01
I (linear)	0.01	0.63	0.09	<0.01
I (nonlinear)	0.1	<0.01	0.12	<0.01
Explicitness	0.04	0.10	0.05	0.04
UDR (lasso)	-0.13	<0.01	-0.13	<0.01

Figure E.1: The figure shows Pearson and Spearman correlation of the Modularity metric with the other disentanglement metrics included in the experiment.

E.2 UDR

Figure E.2 shows the UDR scores plotted with respect to scores from each of the supervised metrics.

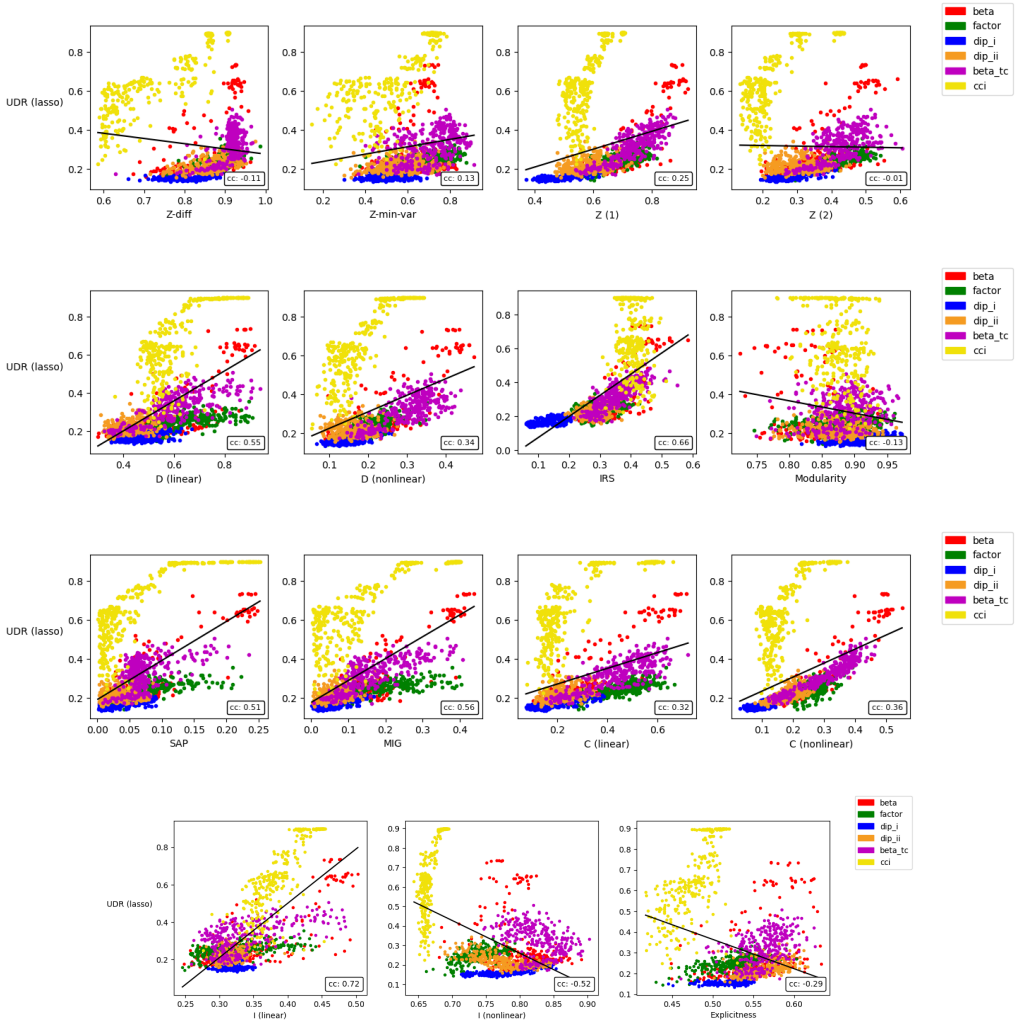


Figure E.2: The UDR scores plotted against scores from supervised metrics. 1st row: Z-diff, Z-min-var, Z (1) and Z (2). 2nd row: D (linear), D (nonlinear), IRS and Modularity. 3rd row: SAP, MIG, C (linear) and C (nonlinear). 4th row: I (linear), I (nonlinear) and Explicitness.

E.3 Metric scores

Figures E.3, E.4 and E.5 show the average score of the 50 models trained for each hyperparameter setting for each of the 6 model types β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE and CCI-VAE, plotted agains hyperparameter value as shown in Table 5.1.

It is seen in all plots that few metrics agree on the value of the reported disentanglement score, however as hyperparameters are increased/decreased, most metrics measuring similar properties increase/decrease comparably.

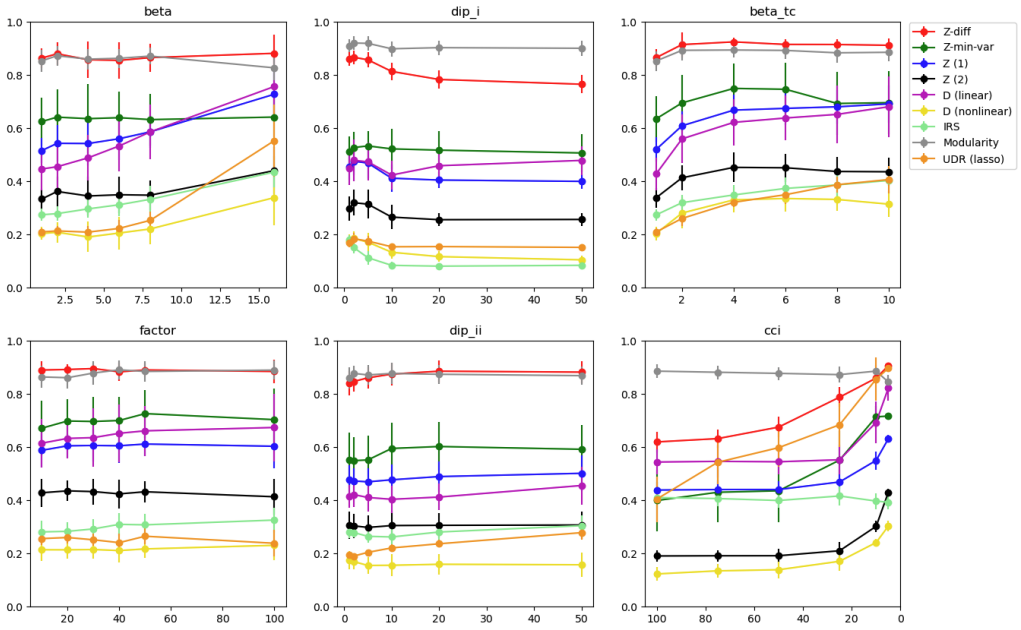


Figure E.3: Separate plots for model types β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE and CCI-VAE show average values of modularity scores assigned to the 50 models trained with the same hyperparameter setting (y-axis) with respect to hyperparameter value (x-axis).

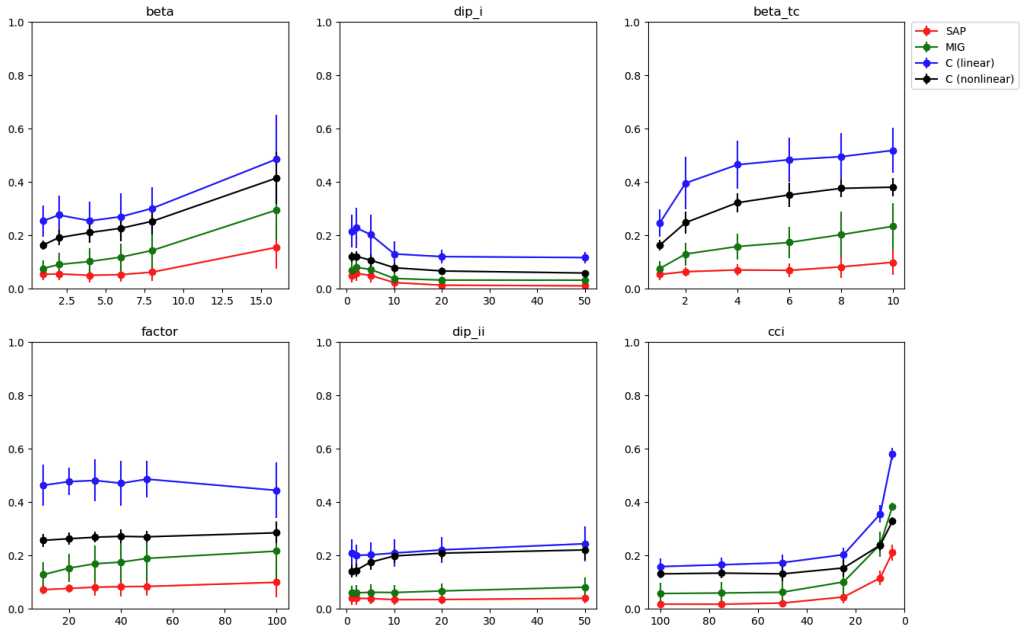


Figure E.4: Separate plots for model types β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE and CCI-VAE show average values of compactness scores assigned to the 50 models trained with the same hyperparameter setting (y-axis) with respect to hyperparameter value (x-axis).

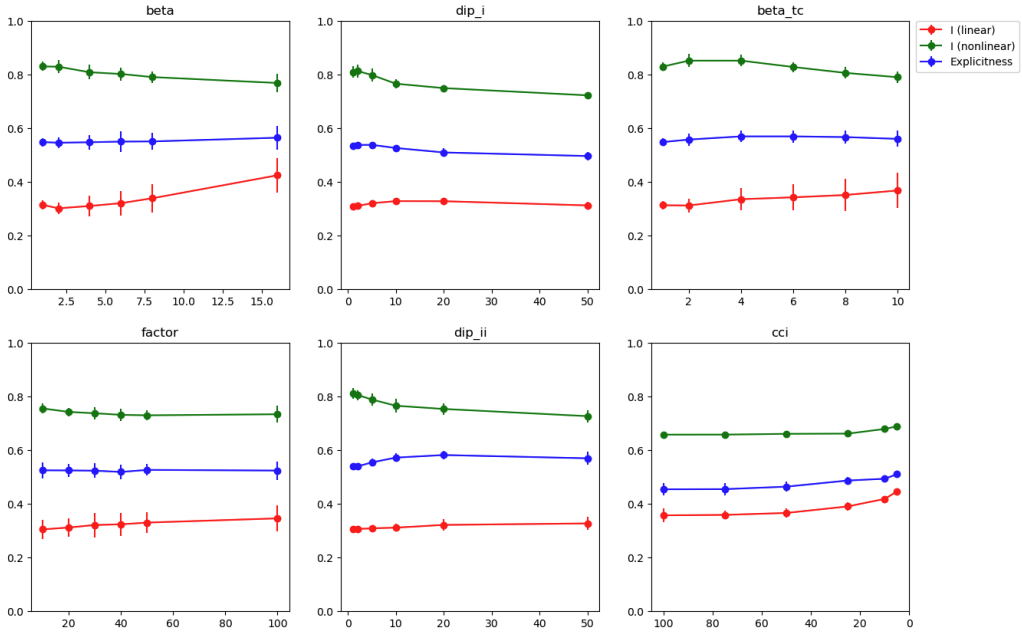


Figure E.5: Separate plots for model types β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE and CCI-VAE show average values of explicitness scores assigned to the 50 models trained with the same hyperparameter setting (y-axis) with respect to hyperparameter value (x-axis).

E.4 Spearman correlation coefficients

This section contains Spearman correlation coefficient equivalents of the correlation matrices comparing disentanglement metrics with prediction accuracy scores in Section 5.3. Figure E.6 correspond to Figure 5.6, Figure E.7 to Figure 5.12, Figure E.8 to Figure 5.13 and Figure E.9 to Figure 5.15.

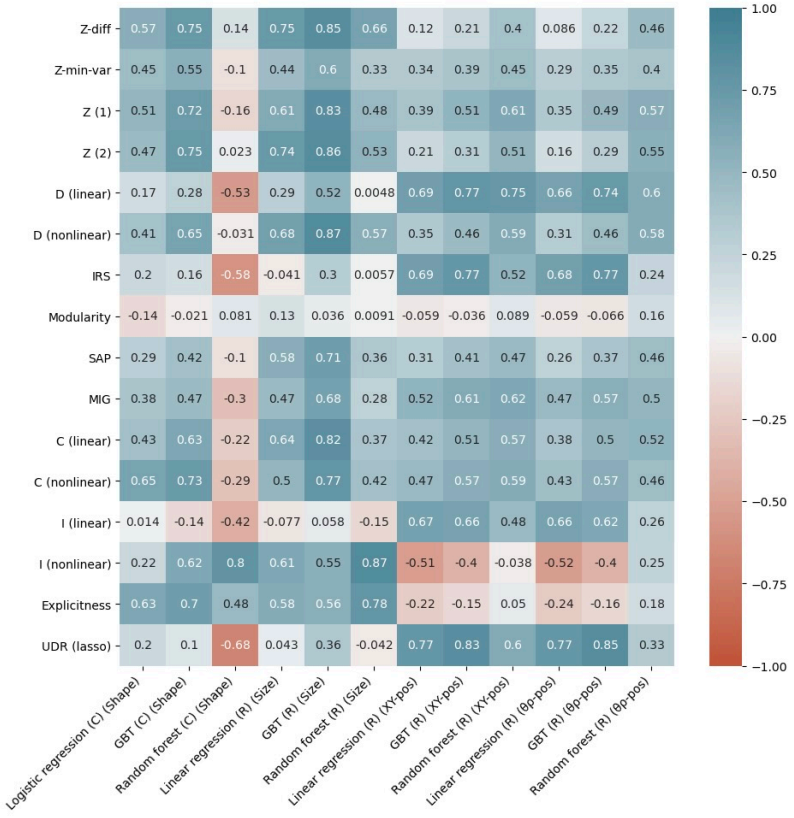


Figure E.6: Spearman correlation coefficient equivalent of Figure 5.6, where representations from all models are included.

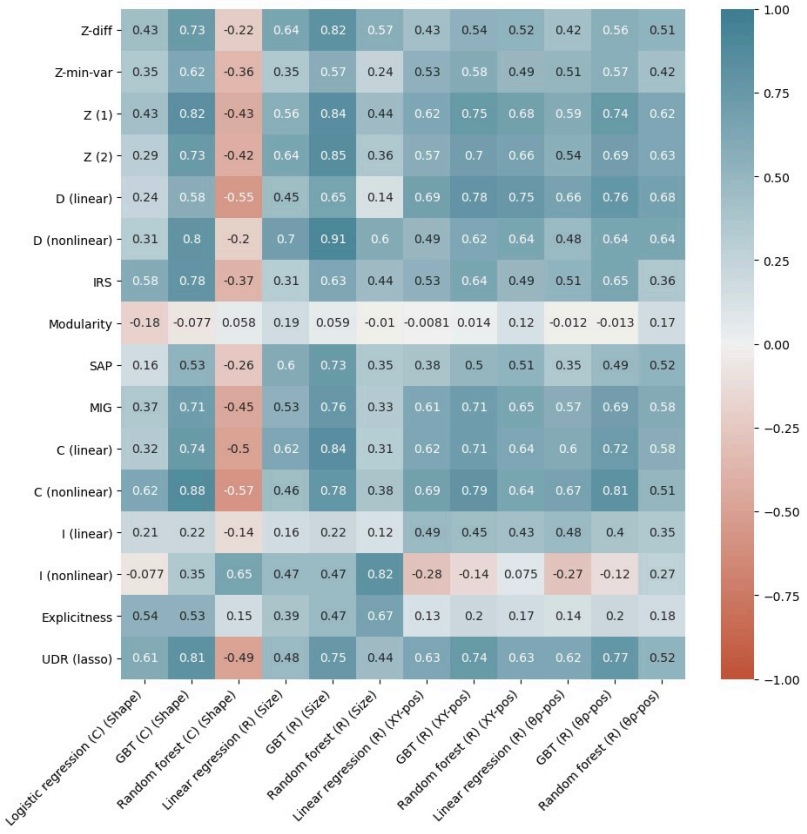


Figure E.7: Spearman correlation coefficient equivalent of Figure 5.12, where representations from models 0-1499 are included.

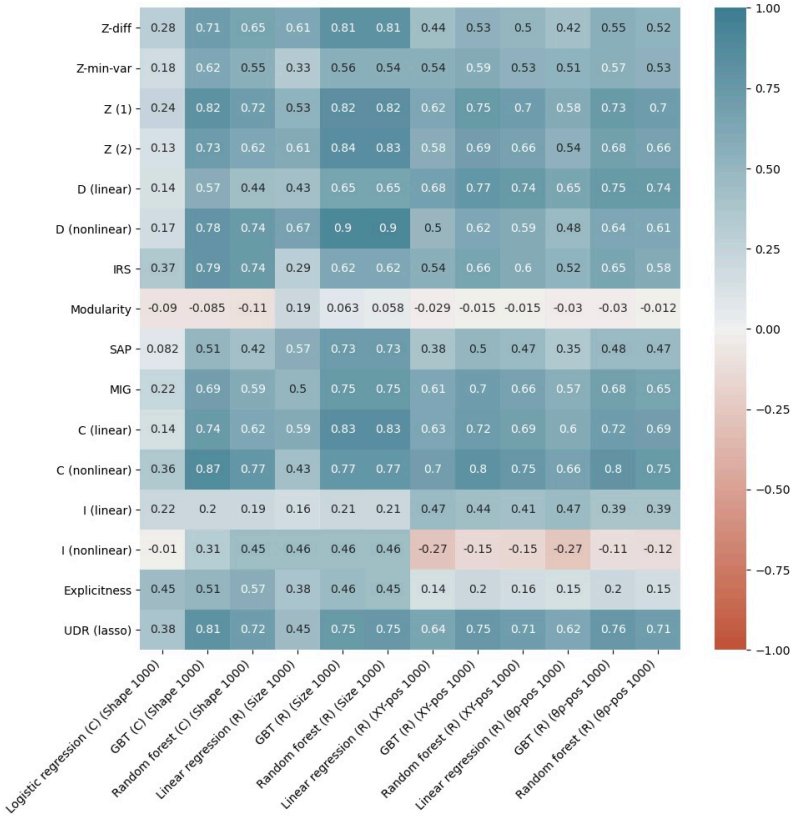


Figure E.8: Spearman correlation coefficient equivalent of Figure 5.13, where predictors are trained on only 1000 data samples and representations from models 0-1499 are included.

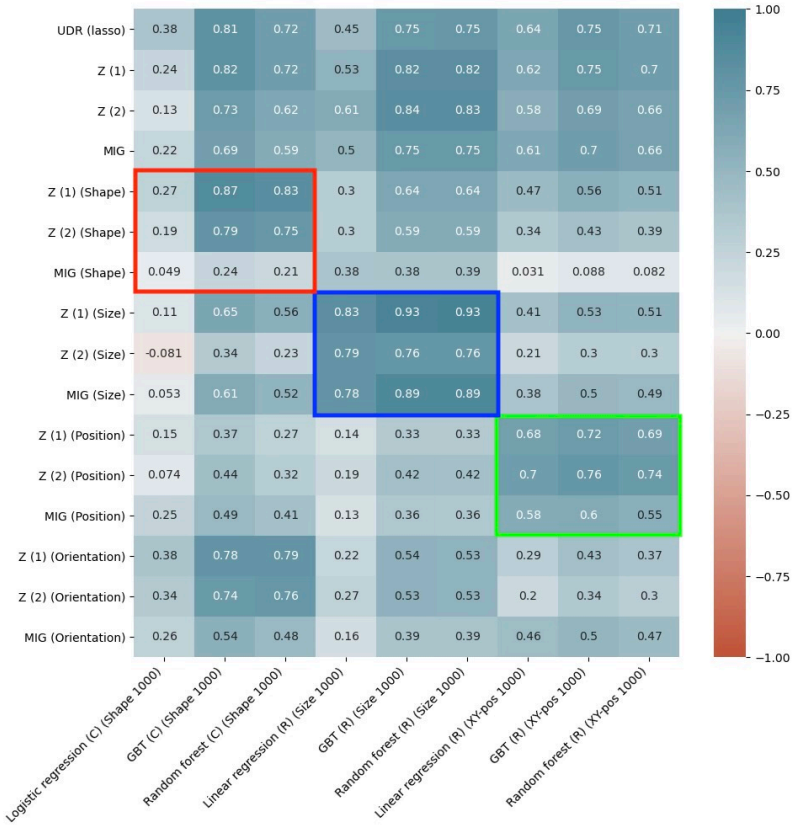


Figure E.9: Spearman correlation coefficient equivalent of Figure 5.15, where predictors are trained on only 1000 data samples and representations from models 0-1499 are included.

Anna Rodum Bjøru

The importance of disentanglement when learning representations

NTNU
Norwegian University of
Science and Technology