Imperial College London
Department of Computing

Inductive Inference From Latent Generative Factors

Damiaan R. Twelker

Submitted in partial fulfilment of the requirements for the MSc degree in Computing Science / Artificial Intelligence of Imperial College London
September 2017

Abstract

Neural networks are increasingly deployed to assist in a variety of real-world tasks. A major obstacle preventing a more widespread adoption is the lack of trust associated with them, due to their inherent obscurity. We propose a learning framework that draws an analogue between human reasoning, and argue that it can efficiently be powered by two components: an obscure neural network to identify low-level concepts, and a human-interpretable ILP component to reason over the extracted concepts. We demonstrate the feasibility of the framework in a proof-of-concept application: with high accuracy, the framework is capable of expressing an unwritten preference describing comfortability in terms of low-level visual factors.

Acknowledgments

I would like to thank Stefanos Zafeiriou, Hugh Salimbeni, and Marc Deisenroth for fruitful discussions. Gratitude is extended to Irina Higgins, Loic Matthey and Arka Pal for helpful comments regarding implementation details of Higgins et al. (2017a). Thanks to Mohammed and Naveen for listening to my endless ideas and telling me if they made sense or not. Lloyd Kamara with DoC CSG has been pivotal allocating the required computational resources. I would like to thank my supervisors, Dr Krysia Broda, Mark Law and Dimitris Kollias for their guidance and commitment over the course of the past six months. I thank my family for supporting my dream to pursue graduate studies abroad.

Contents

Chapter 1

Introduction

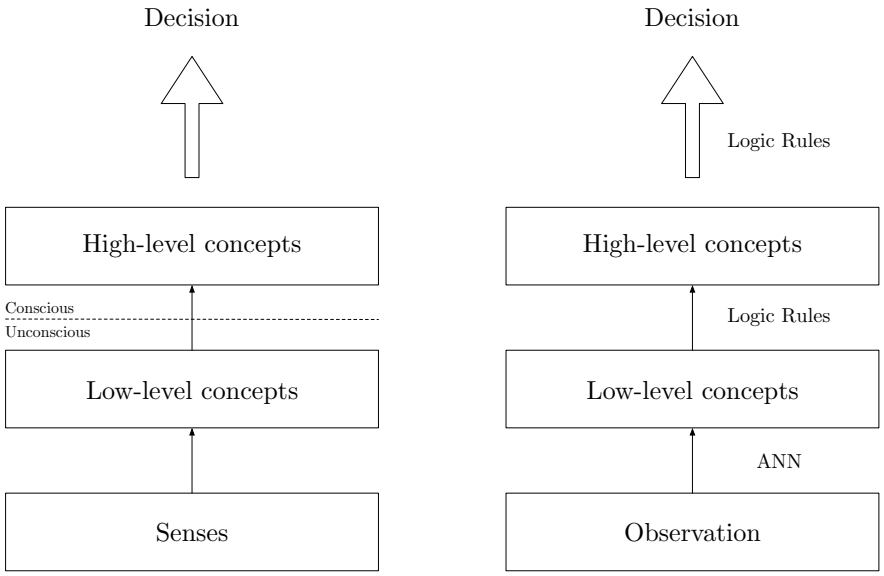
Artificial Intelligence (AI) (Minsky, 1961) has proved itself a valuable tool in many different aspects of society. It provides us with more accurate translations (Gehring et al., 2017), helps provide medical diagnoses (Deo et al., 2017; Tan et al., 2015; Cireşan et al., 2013), and enables self-driving cars (Bojarski et al., 2016). These applications are often underpinned by artificial neural networks (ANNs), which we know to be powerful function approximators (Hornik et al., 1989). A major drawback, however, is that ANNs lack any form of expressibility compared to, for example, logic-based machine learning techniques (Muggleton and De Raedt, 1994), where the learned objective is expressed in a logic language easily interpretable by humans. At the same time, the lack of explanatory power in deep learning applications forms a major obstacle for a deeper integration of these techniques into our lives. Lawmakers are wary of fully automated "black box" decision processes and have started to adopt legislation to address critics' concerns (Council of European Union, 2016; Goodman and Flaxman, 2016).

Efforts to better understand the working of neural networks are prevalent in the literature. They focus on finding a human-understandable interpretation of trained neural networks, either by deriving an end-to-end rule or showing how the outcome changes with different inputs (Samek et al., 2017). These methods are often hard to interpret and do not scale well with the size of the network. Instead of trying to explain the working of a neural network trained to perform a certain task, we propose a hybrid neural/symbolic learning method that produces a human-readable hypothesis.

The human decision process is characterised by reasoning over known constructs. Reasoning is encoded in natural language and expressed by *inner speech* in case of a thought process, or *external speech* in case of a debate or discussion. A complex, unconscious interplay between sensory organs and particular areas in the brain forms the critical link between an observation and the imminent identification of a concept. As a concept is less familiar to the observer, the thought process is more conscious, whereas when the concept is more familiar, the identification or classification is rather unconscious and immediate. For example, for a child to learn the meaning of the hours past 12 in the 24-hour clock is a merely conscious task: 12 is substracted to obtain the more familiar speech label associated with the actual time. Over time this conscious step is no longer necessary and the mapping is implicitly "wired" into the brain. It is this conscious reasoning procedure that we would like any automated decision process to exhibit, and that state-of-the-art deep learning techniques are lacking.

In Figure 1.1 we identify what a more human-like automated decision process could look like. We argue it is sufficient to expose only those areas where information is encoded in natural language, i.e. the conscious reasoning procedure. Attempts to describe the unconscious procedure in some sort of formal language are merely irrelevant, as this procedure precedes consciousness. Hence, the mapping from observations to low-level concepts can effectively be implemented and masked by an artificial neural network. The conscious part of the procedure can be encoded in a formal logic language, as its syntax provides interpretability and the semantics allow for modeling of relations between concepts.

It was found that Variational Autoencoders (VAEs) are capable of isolating and quantifying basic visual concepts from images (Higgins et al., 2017a). By introducing a constraint on one of



(a) Human decision process.

(b) Proposed machine decision process.

Figure 1.1: The observed decision process in the human brain vs. the proposed automated reasoning procedure. The automated procedure is less obscure than an end-to-end artificial neural network as the logic rules are easily interpretable by humans.

the two terms in the marginal log likelihood lower bound, disentanglement performance surpasses any other state-of-the-art technique. The concepts encoded by β-VAE are not directly usable in a logic context, since they are encoded as continuous random variables. In this work, we seek a statistical model that has the same properties as β-VAE but is inherently discrete. The discrete concepts obtained by such a model can then be encoded as logical atoms and combined with an expert opinion to inductively learn the definition of higher level constructs. We use a state-of-the-art inductive learning framework called ILASP (Law et al., 2014, 2016) to perform inductive inference over the extracted visual factors. An additional benefit of VAEs compared to other latent variable models is their generative nature. The generative capacity can become an explicit part of the learning process by visualising intermediate rules to the expert, and as such guiding the search for a better rule.

The discrete model is deployed in a proof-of-concept learning task, as illustrated in Figure 1.2. A user assesses a set of images and orders them according to their preference. Each image is decomposed into a set of low-level concept classes X obtained from a discrete latent variable model. The recorded preferences define an ordering over pairs of X_i. A Learning to Rank Answer Sets (LTR) task then aims to find the rule that defined the ordering over the X_i in terms of low-level concept classes. The end result is a human-readable hypothesis describing the user preference in terms of low-level visual concepts.

1.1 Contributions

Our contribution is threefold: we introduce an end-to-end learning framework modelled after the human reasoning process and demonstrate its viability on a preference learning task (1), we derive a discrete latent variable model that achieves disentanglement performance on par with a state-of-the-art continuous model (2), we show how the same discrete model in a slightly different formulation can achieve a better disentanglement performance and discovers more relevant visual factors than any state-of-the-art technique (3). More specifically, we explore two discrete latent variable models that rely on the novel Gumbel-Softmax distribution (Jang et al., 2016) for a variational approximation of posterior distributions, and show that the second model, an array of univariate mixture models, achieves performance comparable with the state-of-the-art β-VAE (Higgins et al., 2017a).

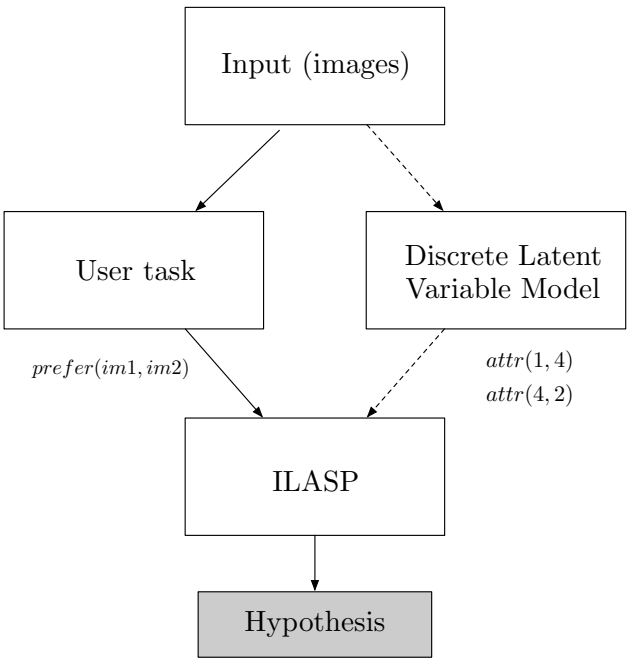


Figure 1.2: The proposed learning framework. The path formed by dotted arrows is hidden from the user.

In a particular formulation, the mixture model surpasses the state-of-the-art and discovers more meaningful visual factors. Finally, the overall concept of learning from visual factors has been verified in a preference learning setting: with relatively high accuracy, an unwritten preference denoting *comfortability* can be expressed in terms of visual factors.

1.2 Related Work

A significant number of different research areas converge into the final learning framework. The most closely related topics are presented here. First, a common alternative to obtaining a transparent decision process is to explain the predictions given by a trained neural network. Secondly, there are other learning methods that combine neural and symbolic learning, although to the best of our knowledge no previous method combines ILP and VAEs. Finally, discrete models have been explored in a VAE setting before, but for different purposes.

As mentioned earlier, methods that aim to explain the working of trained neural networks can be broadly divided into two areas: those that derive an equivalent human-readable rule from the opaque network, and those that align small changes in input with changes in prediction (Samek et al., 2017). Concrete examples of the former category are the logic-based approaches from Zhou et al. (2003); Craven and Shavlik (1996); Thrun (1994). The obtained rule is a logic rule, and interpretability is restricted to symbolic learning tasks. Explanation of networks trained for visual tasks often involves analysis of activation spectra of convolutional layers, e.g. Bau et al. (2017); Zeiler and Fergus (2014); Garnelo et al. (2016). We advocate a different approach, where we prefer the neural network to remain opaque and use its capacity in the same way the biological counterpart forms the unconscious part of the human reasoning procedure.

The idea of a learning framework that by its very formulation is less obscure than an end-to-end neural network is not new. Garnelo et al. (2016) proposes a hybrid neural/symbolic learning framework applied to a reinforcement learning context. A convolutional autoencoder is trained on images representing the states of a simple game. The activation spectra of a single convolutional layer is used to identify distinct objects placed on the playing board. The extracted objects then form the input to a reinforcement learning algorithm that gives the optimal next move. The high-level idea of the framework is similar as the framework proposed here, but it lacks a purely logic-based learning component, and inspects a trained network to acquire concepts as opposed to

obtaining those from the low-level representation given by a VAE. Inspecting activation spectra does not scale well with the size of the network. A more similar framework is given by Higgins et al. (2017c), which was published during the final stages of this work. It also builds on β-VAE (Higgins et al., 2017a), but includes some ground-truth labels of concepts rendering the approach semi-supervised, and lacks a logic-based learning component. They introduce another class of multivariate Gaussian distributions whose components learn to capture the building blocks of a visual factor. Our approach differs in that we modify the model itself in order to fit a univariate Gaussian mixture model to each component of the latent code. Additionally, we introduce a purely logic-based learning component, which provides us with predicate invention and an extra layer of robustness against noisy input data.

Discrete random variables have been considered in a VAE setting before. A common application is unsupervised clustering. Dilokthanakul et al. (2016) and Jiang et al. (2016) describe a multivariate Gaussian mixture model, where the mixture components form clusters in latent space. Both approaches derive a marginal log likelihood lower bound without requiring a variational approximation of the posterior distribution over the discrete latent variable. Such a variational approximation is not possible for discrete random variables; only recently it was found that a continuous approximation of a discrete distribution forms a good alternative (Jang et al., 2016; Maddison et al., 2016). In this project, we are not interested in finding object classes based on clusters in latent space. Instead, since individual dimensions in latent space encode different concepts, we aim to cluster those variations into distinct visual concept classes. Such a model consists of an array of univariate Gaussian mixture models, as opposed to a single multivariate mixture model. To the best of our knowledge, there is no mention of a univariate mixture model formulation in the VAE literature. We derive an inference setting over the discrete random variable both using a variational approximation (Jang et al., 2016), as well as by using an analytical equivalence present in the optimum (Jiang et al., 2016). The former approach incorporates posterior inference, which allows us to identify informative latents by looking at the posterior variance of the univariate Gaussian mixtures.

1.3 Thesis outline

The remainder of this work is organised as follows. In Chapter 2, we introduce topics from the literature that are essential for a good understanding of the remaining chapters. An introduction to logic is given, as well as details of inductive inference in particular forms of logic programs called Answer Set Programs (ASP), which will be used to encode the learning task in Figure 1.2. Chapter 3 provides a more detailed overview of the entire project, and how each chapter fits in the final, novel framework presented in Figure 1.2. Chapter 4 introduces two discrete statistical models for visual concept extraction from images, Chapter 5 is concerned with the evaluation of those models. Finally, the best performing model relative to criteria critical for our application is deployed in an actual application of the framework in Chapter 6. In conclusion, novel work is presented in Chapters 4-6, while Chapters 2-3 expand on relevant topics from the literature.

1.4 Terminology

A brief overview of the most important terms used throughout this work is given in Table 1.1.

Visual factor / visual concept	A common visual aspect present in a range of images. It is assumed that images are created by sampling particular values from visual factors (Higgins et al., 2017a). In that case the visual factors are referred to as *generative* visual factors. Additionally, since this process is hidden from the observer, the factors are called *latent*. Examples are hair length, hair style, facial hair, etc. in a dataset of facial images.
Visual concept class	A distinct class within a visual factor. Examples are short hair length, long hair length, brown eye color, etc.
$d\beta$-VAE	A discrete adjustment of the continuous β-VAE where the Gaussian curve encoding individual visual factors is split up into K evenly-sized bins, assumed to each cover a different concept.

Table 1.1: A definition of terms used repeatedly throughout this work.

Chapter 2

Background

In this chapter we set out the topics required for a thorough understanding of the following chapters. We discuss relevant topics from logic-based machine learning, probability theory, and artificial neural networks. We finalise this chapter with a formal treatment of Variational Autoencoders.

2.1 Inductive Logic Programming

Inductive Logic Programming (ILP) is a generic term covering the intersection of machine learning and logic programming (Muggleton, 1991). The goal for any of these techniques is to inductively derive a rule or hypothesis from observations or examples (Muggleton and De Raedt, 1994). More specifically, any hypothesis H that forms a solution to the learning task has to cover all of the positive examples and none of the negative examples. A possible explanation of the covers relation is given by the notion of entailment. In this section we briefly iterate over the two fundamental logic languages, propositional and first-order logic. Next we extend to normal logic programs and ASP programs. We define the notion of entailment in the context of definite programs and ASP programs using the stable model semantics, which in the context of ASP requires a reduced representation called the *reduct*. Finally we illustrate a particular learning task in the context of ASP programs called Learning to rank Answer Sets (LTR), which will form the basis for the learning tasks performed in Chapter 6.

2.1.1 Principles of Logic

Propositional logic is the logic language restricted to propositional variables (lowercase alphanumerical characters) and the logical connectives $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$. The semantics of the language, or meaning, is defined by the meaning of the logical connectives (their truth table) and an interpretation I. I is the set of propositional variables that are assigned *true*. A *sentence* is a sequence of constants and logical connectives composed in accordance with the language syntax. We say that an interpretation *satisfies* a sentence if that interpretation results in the sentence evaluating to *true*. In that case the interpretation is called a *model* of the sentence. An *unsatisfiable* sentence is a sentence for which no interpretation exists that can make it true, a *valid* sentence is a sentence that is satisfied by every possible interpretation. A *model* of a set of sentences S is an interpretation that satisfies all sentences in S. S *logically entails* a sentence ϕ if every interpretation that satisfies all sentences in S, also satisfies ϕ. We write $S \models \phi$.

Example 2.1.1 (Valid sentence)**.** $\phi = p \vee \neg p$. Possible interpretations are $I_1 = \emptyset$ and $I_2 = \{p\}$. Either interpretation results in ϕ evaluating to true.

Example 2.1.2 (Unsatisfiable sentence)**.** $\phi = p \wedge \neg p$. Neither I_1 nor I_2 is a model of ϕ, hence ϕ is unsatisfiable.

Example 2.1.3 (Logical entailment)**.** $p \wedge q \models p$. For any interpretation I satisfying $p \wedge q$, it holds that $p \in I$ by the semantics of the logical connective $\wedge$. Hence it follows that p holds as well, and we have $p \wedge q \models p$.

Predicate logic, also called first-order logic, extends propositional logic with constants, variables, functions, predicates, and quantifiers $\forall$ and $\exists$. Predicates define a mapping from *terms* to true/false. A term could be a *function* of one or more terms, which returns another term, or a constant or variable. A *ground* sentence is a sentence that does not contain any variables. We refer to Russell and Norvig (2010) for a formal definition of predicate logic in Backus-Naur form.

Propositional and predicate logic sentences have a canonical form called Conjunctive Normal Form (CNF), clausal representation, or clausal theory. It is a conjunction of clauses. A set of clauses is also referred to as a *logic program*. A *clause* is a disjunction ($\vee$) of literals. A *literal* is an atomic sentence or its negation. An atomic sentence is a sentence that cannot be reduced to a smaller sentence. In predicate logic, an atomic sentence is a predicate, in propositional logic, it is a single propositional variable. There is a systematic way to convert propositional and first-order sentences to CNF (e.g. Russell and Norvig (2010)), which we will not discuss here.

Example 2.1.4 (Clausal theory)**.** The following clausal theory is composed of five predicates:

$$(married(X,Y) \vee single(X) \vee widowed(X)) \wedge person(X) \wedge person(Y)$$

The *Herbrand domain* of a clausal theory Th is the set of all constants and ground function symbols that appear in Th. The *Herbrand base* of Th is the set of all ground predicates whose arguments are terms from the Herbrand domain. The *grounding* of Th is the set of all ground instances of clauses in Th, where variables are replaced by terms from the Herbrand domain. Note that in the presence of functions, the Herbrand domain could be an infinite set, and as a result, the grounding could be infinite as well.

Example 2.1.5. Assume the following clausal theory:

$$\begin{aligned} Th &= \{workFromHome(X) \leftarrow sick(X); sick(bob); sick(marga)\} \\ &\equiv (\neg sick(X) \vee workFromHome(X)) \wedge sick(bob) \wedge sick(marga) \end{aligned}$$

The $\equiv$ symbol denotes logical equivalence, and the equivalence follows from the semantics of logical implication: $p \rightarrow q \equiv \neg p \vee q$ (for an overview of additional logical equivalences, see Chapter 1.3 of Rosen (2007)). The following statements hold for Th:

- Its Herbrand domain is $\{bob, marga\}$.
- Its Herbrand base is $\{workFromHome(bob), workFromHome(marga), sick(bob), sick(marga)\}$.
- Its grounding is $\{sick(bob), sick(marga), workFromHome(bob) \vee \neg sick(bob), workFromHome(marga) \vee \neg sick(marga)\}$.

A *Horn* clause is a clause with at most one positive literal. If there is precisely one positive literal, the clause is called a *definite* clause. If there is zero positive literals, then the clause is a *denial*, also called a constraint. A definite clause with no negative literals is called a *fact*, otherwise the definite clause is a *rule*. The positive literal in a rule is referred to as the *head* of the rule, the negative literals form the *body*.

Example 2.1.6 (Rule)**.** If we convert the sentence $p \leftarrow (q \wedge b \wedge a)$ to its clausal representation, it becomes clear that there is just one positive literal p and the resulting clause is a definite clause:

$$p \leftarrow (q \wedge b \wedge a) \equiv \neg(q \wedge b \wedge a) \vee p \equiv \underbrace{\neg q \vee \neg b \vee \neg a \vee p}_{\text{Conjunctive Normal Form (CNF)}}$$

At the same time, this definite clause is a rule, as there is a nonzero number of negative literals.

Example 2.1.7 (Denial/constraint)**.** A denial has zero positive literals: $\neg p \equiv \neg p \vee \bot \equiv \bot \leftarrow p$.

Example 2.1.8 (Fact)**.** An example of a definite clause that is a fact is p. There is no negative literals, and precisely one positive literal p.

Example 2.1.9 (Prolog)**.** Prolog is a programming language that allows for reasoning over sets of Horn clauses (Bratko, 1986). In Prolog one would write the Horn clause from Example 2.1.6 as `p :- q, b, a.`

2.1.2 Answer Set Semantics

A *Herbrand interpretation* is a subset of the Herbrand base that is assigned true, while its complement evaluates to false. A *Herbrand model* is a Herbrand interpretation I such that for every rule R in the grounding of Th it holds that if I satisfies the body of R, I also satisfies the head. Th is called satisfiable if it has at least one Herbrand model. Any satisfiable set Th of definite clauses has a unique minimal Herbrand model called the *Least Herbrand Model*. The least Herbrand model of a definite logic program is the set of ground predicates that are logically entailed by the program (De Raedt, 2008).

Example 2.1.10 (Herbrand model of a definite logic program)**.** Taking the clausal theory from Example 2.1.5, we can define a Herbrand interpretation $I_1 = \{sick(bob)\}$. Recall that facts are rules with no body literals: $p \equiv p \vee \bot \equiv \top \vee p \equiv p \leftarrow \top$. The body of a fact is satisfied by any interpretation I, as $\top$ is true by definition. Hence any Herbrand model of a definite logic program that contains facts, has to include the facts. In the theory from Example 2.1.5 there are two facts: $sick(bob)$ and $sick(marga)$. Since $sick(marga) \notin I_1$, I_1 cannot be a Herbrand model of Th.

Example 2.1.11 (Least Herbrand model of a definite logic program)**.** In Example 2.1.10 we saw that any Herbrand model of a definite logic program includes fact rules. Additionally, the Herbrand model M includes head predicates of ground rules whose bodies are satisfied by M. The fact $sick(bob)$ satisfies the body of the ground rule $workFromHome(bob) \leftarrow sick(bob)$. Similarly, $sick(marga)$ entails $workFromHome(marga)$. A Herbrand model of Th is thus given by $\{sick(bob), sick(marga), workFromHome(bob), workFromHome(marga)\}$. As there is no other Herbrand model of Th, this Herbrand model is at the same time the least Herbrand model.

A *normal* clause extends definite clauses with the negation as failure operator, written as *not*. The literal $not\, p$ evaluates to false if and only if p holds. For any normal logic program P we can find the *reduct* of its grounding $ground(P)$ with respect to a set of atoms X (Gelfond and Lifschitz, 1988):

- Remove any rule from P whose body contains the negation as failure of an atom in X.

- From all remaining rules, remove any negation as failure atoms.

We write $ground(P)^X$ for the reduct of $ground(P)$ with regard to X (Law, 2017). Any interpretation X that is the least Herbrand model of $ground(P)^X$ is called a *stable model* or *answer set*.

Example 2.1.12 (Reduct of a normal logic program)**.** We extend the clausal theory from Example 2.1.5 with the normal rule $workFromOffice(X) \leftarrow not\, workFromHome(X)$. The final logic program P looks as follows:

```
workFromHome(X) :- sick(X).
workFromOffice(X) :- not workFromHome(X).
sick(bob).
sick(marga).
```

The grounding of P is given by:

```
workFromHome(bob) :- sick(bob).
workFromHome(marga) :- sick(marga).
workFromOffice(bob) :- not workFromHome(bob).
workFromOffice(marga) :- not workFromHome(marga).
sick(bob).
sick(marga).
```

If we take $X = \{sick(bob), workFromHome(bob)\}$, the reduct of $ground(P)$ w.r.t. X becomes:

```
workFromHome(bob) :- sick(bob).
workFromHome(marga) :- sick(marga).
workFromOffice(marga).
sick(bob).
sick(marga).
```

X is not the least Herbrand model of $ground(P)^X$, as $sick(marga) \notin X$.

Example 2.1.13 (Stable model of a normal logic program)**.** If we take $X = \{sick(bob), sick(marga), workFromHome(bob), workFromHome(marga)\}$ and the program P from Example 2.1.12, the reduct of $ground(P)$ w.r.t. X becomes:

```
workFromHome(bob) :- sick(bob).
workFromHome(marga) :- sick(marga).
sick(bob).
sick(marga).
```

This is identical to the grounding of the original definite logic program, of which we found the least Herbrand model to be X. Since X is the least Herbrand model of $ground(P)^X$, it is a stable model or answer set of normal logic program P.

Answer Set Programming (ASP) has extended normal logic with *constraints* and *choice rules*. Recall that normal clauses are definite clauses with an extended construct (*not*) in the body. The head is unchanged, and remains formed by one positive literal. ASP does allow clauses with an empty head: these are called constraints. For X to be a Herbrand model of an ASP program P, if X satisfies the body of a rule in $ground(P)^X$, it must also satisfy the head. If an interpretation satisfies the body of a constraint, it must contain $\bot$ (see Example 2.1.7), i.e. make the head true for it to be a Herbrand model. Since no interpretation contains $\bot$ ($\bot$ cannot be assigned true), X is not a Herbrand model and cannot be an answer set. Thus, no interpretation that satisfies the body of a constraint rule can be an answer set. Constraints provide a means for filtering out unwanted answer sets.

An interpretation X satisfies a choice rule $a\{h_1, \ldots, h_n\}b$ if $a \leq |\{h_1, \ldots, h_n\} \cap X| \leq b$: the number of elements in the choice rule that are also in X is greater than or equal to integer a and less than or equal to integer b. To find the reduct of a ground ASP program with choice rules, an additional step is required (Law et al., 2015c):

- For each choice rule R with aggregate A in the head:
  - if X does not satisfy the aggregate, remove the head of R, turning it into a constraint.
  - if X satisfies A, eliminate R and generate a rule for each atom a of A that is true in X, with a as head and the body of R as body.

For a definite logic program P, an atom is entailed by P if it is in the least Herbrand model of P. In a normal logic context, answer sets provide us with a means to formulate entailment. The reduct of an ASP program yields a definite logic program, which has a unique least Herbrand model. The intuition behind the formulation of the reduct is as follows. Given an interpretation X, rules with $not\, p$ for any $p \in X$ in their body are removed because the body is never satisfied. Recall that the body is a conjunction of atoms, hence all atoms have to be satisfied for the body to be satisfied. Because p is in X, X satisfies p, and $not\, p$ does not hold. The remaining negation as failure literals evaluate to true, since they are not in X (because those have been removed), and by definition of the *not* operator, they evaluate to true. After removing those, the program does not contain any normal logic clauses anymore, and we can find the unique least Herbrand model. In an ASP context, an atom A is *bravely* entailed by a program P if it is true in at least one stable model of P (Law et al., 2014). It is written as $P \models_b A$. An atom A is *cautiously* entailed by a program P if it is true in every stable model of P: $P \models_c A$ (Law et al., 2014).

In addition to choice rules and constraints, ASP programs can also include *weak constraints* (Calimeri et al., 2012). They differ from *hard* constraints in that they do not exclude answer sets of a program, but define an ordering over them. The syntax of a weak constraint is similar to a

hard constraint. Weak constraints are denoted by a tilde (:∼) as opposed to a dash (:-) and are followed by a list of terms. The first term defines a *weight* and *level*, the rest of the terms in the list determine uniqueness of the constraint. Let $W(I, l)$ denote the sum of the weights of ground weak constraints with level l that are satisfied by an interpretation I. An answer set A of an ASP program P *dominates* an answer set A_2 if there is a level q for which $W(A, q) < W(A_2, q)$ and $W(A, p) = W(A_2, p)$ for all $p > q$ (Calimeri et al., 2012). In other words, A dominates A_2 if A minimises the weights of satisfied ground weak constraints compared to A_2 at the first (highest) level where the sum of these weights is different. Finally, an answer set A of an ASP program P is optimal if it is not dominated by any other answer set A_2 of P.

Example 2.1.14 (Weak constraints)**.** Consider the following ASP program listing the properties of a bike:

```
1{saddle(soft); saddle(hard)}1.
1{tire(thin); tire(thick)}1.

:~ saddle(hard). [1@2]
:~ tire(thick). [1@1]
```

The answer sets of this program are:

1. $\{saddle(hard); tire(thick)\}$ (1, 1)
2. $\{saddle(hard); tire(thin)\}$ (0, 1)
3. $\{saddle(soft); tire(thick)\}$ (1, 0)
4. $\{saddle(soft); tire(thin)\}$ (0, 0)

There are three weak constraints, two with weight 1 at level 2, and one with weight 1 at level 1. The weak constraints are already ground, since the program only contains propositional atoms. For each answer set A, the value of $W(A, l)$ is displayed in parentheses for each level l. Answer set 4 is preferred, then 3, then 2, then 1. Intuitively, the program says that bikes with soft saddles and thin tires are preferred. Then, bikes with a soft saddle, then bikes with a thin tire, and finally the last possible option.

In a definite logic context, Inductive Logic Programming aims to find an optimal hypothesis H that covers all positive examples E^+ and none of the negative examples E^-, given some background knowledge B (Muggleton and De Raedt, 1994):

$$\forall e^+ \in E^+ : B \cup H \models e^+$$
$$\forall e^- \in E^- : B \cup H \not\models e^+$$

Optimality could be determined by the length of the hypothesis: the less number of literals, the more optimal the solution. Note the use of the entailment symbol $\models$. Entailment for definite logic programs has been defined in the previous section. We saw that entailment for ASP programs is defined in terms of brave entailment and cautious entailment. Inductive logic programming for ASP programs similarly is defined in terms of *cautious induction* and *brave induction* (Sakama and Inoue, 2008). In cautious induction for ASP programs (ILP_c), the goal is to find a hypothesis H such that $B \cup H$ is satisfiable, and for all answer sets A of $B \cup H$ we have $\forall e^+ \in E^+ : e^+ \in A$ and $\forall e^+ \in E^+ : e^+ \notin A$. B is an ASP program providing background knowledge, i.e. rules that are known beforehand. An ASP program is *satisfiable* if it has at least one answer set. Brave induction for ASP programs aims to find a hypothesis H such that there is at least one answer set A that contains all positive examples and none of the negative examples: $\forall e^+ \in E^+ : e^+ \in A$ and $\forall e^+ \in E^+ : e^+ \notin A$.

2.1.3 ILASP

ILASP (Inductive Learning from Answer Set Programs) (Law et al., 2014, 2015a) is a framework capable of inductive inference from answer set programs. It can perform several types of learning tasks: Learning from Answer Sets (LAS), Learning from Ordered Answer Sets (LOAS), and Learning to Rank Answer Sets (LTR).

In Learning from Answer Sets (LAS), examples are *partial interpretations* (Law et al., 2014). A partial interpretation e is composed of two sets of atoms: the inclusions e^{inc} and the exclusions e^{exc} written $e =< e^{inc}, e^{exc} >$. A Herbrand interpretation I *extends* partial interpretation e if $e^{inc} \subseteq I$ and $e^{exc} \cap I = \emptyset$: all of the inclusions and none of the exclusions are in I. Let $AS(P)$ denote the answer sets of an ASP program P. Then H is a solution of a LAS task $T =< B, S_M, E^+, E^- >$ if $H \subseteq S_M$ and $\forall e^+ \in E^+ \exists A \in AS(B \cup H)$ s.t. A extends e^+ and $\forall e^- \in E^- : \nexists A \in AS(B \cup H)$ s.t. A extends e^- (Law et al., 2014). S_M is the search space and is defined by a set of head and body *mode declarations* $M =< M_h, M_b >$ which together form the *language bias* (Law et al., 2014). S_M restricts the search space of the hypothesis: the hypothesis has to be constructed using rules from S_M.

Example 2.1.15 (Learning from Answer Sets in ILASP)**.** We illustrate the syntax of a LAS task in ILASP using an example. Assume that we know $sick(bob)$ and $sick(marga)$ hold. Furthermore, we observe $workFromHome(bob)$ and $workFromHome(marga)$. We are trying to learn a rule that describes the relationship between our background knowledge (the two facts) and the observed facts $workFromHome$. In this case our input to ILASP would look as follows:

```
#modeh(1, workFromHome(var(person))).
#modeb(1, sick(var(person))).

sick(bob).
sick(marga).

#pos(e1, {sick(bob), workFromHome(bob)}, {}).
#pos(e2, {sick(marga), workFromHome(marga)}, {}).
```

The *#modeh* predicate defines the head mode declarations, while the *#modeb* defines the body mode declarations. The modeh statement says that we are trying to learn a rule with the predicate $workFromHome(X)$ in the head, where X is a variable. The integer 1 is the *recall* and indicates how often the predicate can occur in a rule. The *#pos* predicate is an arity 3 predicate describing a positive example, which takes as arguments an identifier, the inclusions, and the exclusions. The *#neg* predicate (not used in this example) is defined analougously. Upon execution of the learning task, we obtain the correct hypothesis `workFromHome(V0) :- sick(V0).`

Learning from Ordered Answer Sets (LOAS) (Law et al., 2015b) allows the specification of an ordering over positive partial examples E^+ which a hypothesis, containing weak constraints, is required to respect. The details are not discussed here. Instead, we focus on Learning to Rank Answer Sets (LTR) (Law et al., 2017) which forms a simplification of LOAS. The aim is to find a hypothesis made up of weak constraints that together with a background knowledge B respects orderings pairwise defined over sets of rules. An ordering example (e_1, e_2) is respected by an hypothesis H if $H \cup B \cup e_1$ and $H \cup B \cup e_2$ each have one answer set, A_1 and A_2 respectively, and A_1 dominates A_2 (Law et al., 2017). H is a solution to a LTR Task $T =< B, S_M, O >$ if H respects all ordering examples in O. S_M is the search space of weak constraints, B an ASP program providing the background knowledge, and O a set of pairwise orderings over sets of rules. If the examples are expected to be noisy, one can choose to maximise the number of ordering examples matched as opposed to matching all. In that case the notion of an optimal solution changes. The optimality metric for a hypothesis H is now given by the length of H plus the sum of weights assigned to the ordering examples that are not respected by H.

Example 2.1.16 (LTR task in ILASP)**.** Assume the setting of Example 2.1.14. We provide the orderings defined by the weak constraints in Example 2.1.14 along with the background knowledge as an ILASP task, and attempt to find the weak constraints that defined these orderings.

```
#modeo(1, saddle(const(s))).
#modeo(1, tire(const(t))).

#constant(s, soft).
#constant(s, hard).
#constant(t, thin).
#constant(t, thick).

#order({saddle(soft). tire(thin).}, {saddle(soft). tire(thick).}).
#order({addle(soft). tire(thick).}, {saddle(hard). tire(thin).}).
#order({saddle(hard). tire(thin).}, {saddle(hard). tire(thick).}).

#weight(1).
```

The *#modeo* predicate is similar to the *#modeh* predicate buit instead of guiding the search for a head predicate in a hypothesis, it defines the literals that can occur in a weak constraint hypothesis. The *#order* predicate forms a single ordering example, and defines an ordering over two sets of rules. The weight indicates the weights that should be considered for each constraint level. ILASP returns the following simplified hypothesis:

```
:~ saddle(hard).[1@2, 1]
:~ tire(thick).[1@1, 2]
```

The second terms in the list of terms following the weak constraints are identifiers defining their uniqueness.

2.2 Probability Theory

In this section, we provide an overview of the mathematical background required for the remainder of this work. We adopt the arrow notation $\vec{x}$ for vectors. Matrices are denoted by capital letters.

Definition 2.2.1 (Multivariate normal distribution)**.** A random vector $\vec{x} \in \mathbb{R}^D$ is multivariate Gaussian distributed with mean $\vec{\mu}$ and covariance matrix $\Sigma = AA^T$ if $\vec{x} = \vec{\mu} + A\vec{z}$ for some $A \in \mathbb{R}^{D \times N}$ and $\vec{z} \in \mathbb{R}^D$ where $\vec{z}$ is a vector of independently identically distributed (i.i.d.) standard normal random variables (Klebaner, 2005, p. 41).

Corollary 2.2.1. A vector $\vec{z}$ of N i.i.d. standard normal random variables is multivariate normal distributed with mean $\vec{\mu} = \vec{0}$ and covariance matrix $\Sigma = I$.

Proof. Take $A = I$ and $\vec{\mu} = \vec{0}$. Then $\vec{x} = \vec{\mu} + A\vec{z} = \vec{z}$ and $\Sigma = I$. □

Definition 2.2.2 (Expectation of a continous random variable)**.** The expectation of a continuous random variable $\vec{x}$ w.r.t. a probability distribution $p(\vec{x})$ is given by:

$$\mathbb{E}_{p(\vec{x})}[\vec{x}] = \int_{\vec{x}} p(\vec{x})\vec{x}\, d\vec{x} \tag{2.1}$$

Corollary 2.2.2.

$$\mathbb{E}_{p(\vec{x})}[af(\vec{x})] = a\mathbb{E}_{p(\vec{x})}[f(\vec{x})] \tag{2.2}$$

Corollary 2.2.3.

$$\mathbb{E}_{p(\vec{x})}[f(\vec{x}) + g(\vec{x})] = \mathbb{E}_{p(\vec{x})}[f(\vec{x})] + \mathbb{E}_{p(\vec{x})}[g(\vec{x})] \tag{2.3}$$

Definition 2.2.3 (Covariance)**.** The covariance between two scalar-valued random variables x_1 and x_2 is defined as (Deisenroth and Zafeiriou, 2017):

$$\mathrm{Cov}[x_1, x_2] = \mathbb{E}[x_1 x_2] - \mathbb{E}[x_1]\mathbb{E}[x_2] \tag{2.4}$$

Definition 2.2.4 (Monte-Carlo Sampling)**.** If an expectation cannot be calculated analytically, it can be approximated by for example Monte Carlo sampling. A Monte Carlo estimate of the expectation of a function $f(x)$ under a distribution $p(x)$ is given by:

$$\mathbb{E}_{p(x)}[f(x)] = \int_x f(x)p(x)\,dx \tag{2.5}$$

$$\approx \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}) \qquad \text{where } x^{(s)} \sim p(x) \tag{2.6}$$

Definition 2.2.5 (Convexity)**.** A twice differentiable function whose second derivative is everywhere positive is called a convex function (Bishop, 2006, p. 56).

Example 2.2.1. $-\ln(x)$ is convex:

$$\frac{\partial - \ln(x)}{\partial x} = -\frac{1}{x} \tag{2.7}$$

$$\frac{\partial^2 - \ln(x)}{\partial x^2} = \frac{\partial - x^{-1}}{\partial x} = x^{-2} > 0 \tag{2.8}$$

Definition 2.2.6 (Jensen's inequality)**.** If ϕ is a convex function, then $\phi(\mathbb{E}[x]) \leq \mathbb{E}[\phi(x)]$ where x is a random variable.

Definition 2.2.7 (Product rule)**.** For any two random variables x and y:

$$p(x, y) = p(y \mid x)p(x) \tag{2.9}$$

Definition 2.2.8 (Sum rule)**.** For any two random variables we have:

$$p(x) = \int_y p(x, y)\,dy \qquad (y \text{ is continuous}) \tag{2.10}$$

$$p(x) = \sum_y p(x, y) \qquad (y \text{ is discrete}) \tag{2.11}$$

Corollary 2.2.4 (Bayes' rule)**.** Bayes' rule follows from the product rule and sum rule:

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)} = \frac{p(x, y)}{\int_y p(x, y)\,dy}$$

(In the discrete case, the integral over y is replaced by a summation.)

Definition 2.2.9 (Kullback-Leibler Divergence)**.** The **Kullback-Leibler (KL) divergence** is a measure of similary between two probability distributions $p(x)$ and $q(x)$. For continuous random variables it is defined as:

$$D_{KL}(q(x)\,||\,p(x)) = \int_x q(x)\ln\frac{q(x)}{p(x)}\,dx \tag{2.12}$$

$$= \int_x q(x)\ln q(x)\,dx - \int_x q(x)\ln p(x)\,dx \tag{2.13}$$

$$= \mathbb{E}_{q(x)}[\ln q(x)] - \mathbb{E}_{q(x)}[\ln p(x)] \tag{2.14}$$

Similarly, for discrete distributions $g(x)$ and $h(x)$ we get:

$$D_{KL}(g(x)\,||\,h(x)) = \sum_x g(x)\ln\frac{g(x)}{h(x)} \tag{2.15}$$

The KL-divergence has two important properties: it is non-negative and remains invariant under invertible transformations.

Definition 2.2.10 (Determinant of a diagonal matrix)**.** The determinant of a diagonal matrix $A \in \mathbb{R}^{n \times n}$ is the product of its diagonal elements: $|A| = \prod_{i=0}^{n-1} a_{ii}$.

Definition 2.2.11 (Multivariate normal distribution pdf)**.** The probability density function of a multivariate normal (Gaussian) distribution with mean μ and covariance matrix Σ is given by:

$$p(\vec{x}) = (2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} \exp -\frac{1}{2}(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu}) \tag{2.16}$$

where $\vec{x}, \vec{\mu} \in \mathbb{R}^D$, $\Sigma \in \mathbb{R}^{D \times D}$

2.3 Neural Networks

In their most generic form, Artificial Neural Networks (ANNs) are composed of artificial neurons, grouped together in *layers*. Connectivity between layers is weighted using some weight matrix W, while there is no intra-layer connectivity. The output of a layer is computed by multiplying the *activation* (output) of incoming neurons with the layer-specific weight matrix W, adding a bias vector $\vec{b}$, and taking one of several *activation functions* over the resulting vector. A pre-defined loss function measures the loss incurred between the network's input and the output of the final layer. One of several optimisation techniques, often based on gradient descent, is used to find the optimal weights and biases, such as stochastic gradient descent (SGD) (Bottou, 2010) or variations (Kingma and Ba, 2014). Perhaps fueled by the availability of powerful GPUs and automatic differentiation frameworks (Abadi et al., 2015; Theano Development Team, 2016), ANNs have seen adoption across all branches of science and engineering (e.g. medicine (Baxt, 1991), civil engineering (Wu et al., 1992), and robotics (Bateux et al., 2017)) in different forms and shapes (e.g. LSTMs (Hochreiter and Schmidhuber, 1997) or CNNs (LeCun et al., 1989)). We illustrate the concepts introduced in this section by deriving the backpropagation algorithm for *feedforward neural networks*.

2.3.1 Feedforward Neural Networks

Feedforward neural networks are a particular class of ANNs where the weighted connections between neurons form a directed acyclic graph (Murphy, 2012). This differs from recurrent neural networks, where the weighted connections between neurons form a cycle.

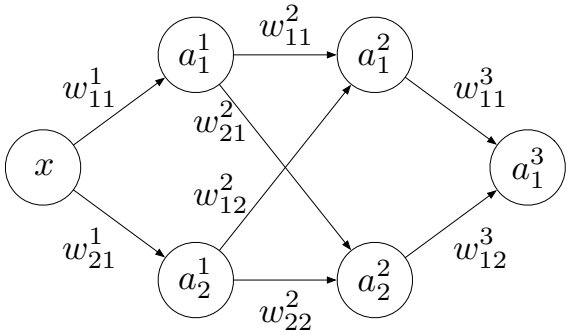


Figure 2.1: A simple feedforward neural net with one input variable x, in this case also called a multi-layer perceptron (MLP) (Murphy, 2012) due to the presence of more than one hidden layer. Arrows denote connections, weight w_{ij}^k denotes the weight on the connection from neuron j in hidden layer $k-1$ to neuron i in hidden layer k. Similarly, a_i^k denotes the activation of neuron i in layer k. Layers are number 1 to $N+1$, where N is the number of hidden layers. Neurons are numbered from top to bottom, starting at 1.

The *activation* at each node in a Figure 2.1 is given by the output of an *activation function* taking the *field* at the node. The field at node j in layer l is defined by: $z_j^l = \sum_k w_{jk}^l \cdot a_k^{l-1} + b_j^l$ (Nielsen, 2015). In vector form, it is a dot product of all incoming weights and activations plus a bias term b. Common activation functions are ReLU (Glorot et al., 2011), which is defined as $f(x) =$

$\max(0, x)$, and the hyperbolic tangent function $f(x) = tanh(x)$. The choice of activation function impacts the gradient calculations in the backpropagation phase, and thus impacts performance (Karlik and Olgac, 2011).

The values of the weights and biases are determined in a process called *training*. Before the training starts, the weights and biases of the network are initialised using some initialisation method. Possible initialisation methods include uniform random samples from some interval, Glorot initialisation (Glorot and Bengio, 2010), or He initialisation (He et al., 2015). The initialisation method plays an important role in the convergence of stochastic gradient descent methods; local minima may be avoided by careful initialisation (Wessels and Barnard, 1992). At each training step, the activations at all neurons are calculated (*forward phase*), after which the gradients are computed starting at the output layer (*backward phase*) with regard to a *loss* function to determine the values of the weights and biases in the next iteration. The loss function is often a function of the output of the final layer and the input of the network x. Choice of loss function depends on the nature of the data and the application. For example, in autoencoders as we will see shortly, the loss function is an approximation of the lower bound of the marginal log likelihood. We derive the gradients for any MLP in Example 2.3.1.

Example 2.3.1 (Backpropagation)**.** This example is based on Nielsen (2015). Suppose we have a multi-layer perceptron with L layers, inputs $\vec{x}$ and labels $y(\vec{x})$, and we want to train a classifier: a function to predict the labels for unseen input. We define a cost function C, which minimises the distance between the output of the net a^L (the predicted labels) and the actual labels: $C = \frac{1}{2}||\vec{a}^L - \vec{y}(\vec{x})||_2^2$. The scaling factor $\frac{1}{2}$ is included for convenience as we will see shortly in the gradient calculations. The training objective is then to minimise this cost w.r.t. the parameters of the neural network b_j^l and w_{ji}^l.

The contribution of node j in layer L to the total cost C is given by:

$$\begin{aligned}&\frac{1}{2} \cdot (a_j^L - y_j(x))^2) \\ &= \frac{1}{2}(\sigma(z_j^L)^2 - 2\sigma(z_j^L)y_j(x) + y_j(x)^2)\end{aligned} \tag{2.17}$$

, where σ denotes the activation function. As we are training a classifier, the activation function at the output layer will be of the logistic family (e.g. softmax or softplus) in order to normalise the unnormalised class probabilities (also called *logits*) over the available class labels. Differentiating with respect to the field z_j^L we get:

$$\begin{aligned}\frac{\partial C}{\partial z_j^L} &= \frac{1}{2} \cdot 2\sigma(z_j^L)\sigma'(z_j^L) - \frac{1}{2} \cdot 2\sigma'(z_j^L)y_j(x) \\ &= \sigma(z_j^L)\sigma'(z_j^L) - \sigma'(z_j^L)y_j(x) \\ &= (\sigma(z_j^L) - y_j(x))\sigma'(z_j^L) \\ &= (a_j^L - y_j(x))\sigma'(z_j^L) \\ &:= \delta_j^L\end{aligned} \tag{2.18}$$

Next we find the contribution from the field of node z_j^l for $l < L$ to C. Since we are calculating these quantities backwards (hence the term *backpropagation*), we know the contribution of the fields z_k^{l+1} for all nodes k in layer $l+1$ to C. Since all z_k^{l+1} are a function of z_j^l, we can use the chain rule to find the contribution of z_j^l to C in terms of z_k^{l+1}:

$$\begin{aligned}\frac{\partial C}{\partial z_j^l} &= \sum_k \frac{\partial C}{\partial z_k^{l+1}} \cdot \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l}\end{aligned}$$

$$
\begin{aligned}
&= \sum_k \delta_k^{l+1} \frac{\partial \sum_i w_{ki}^{l+1} a_i^l}{\partial z_j^l} \\
&= \sum_k \delta_k^{l+1} \frac{\partial \sum_i w_{ki}^{l+1} \sigma(z_i^l)}{\partial z_j^l} \\
&= \sum_k \delta_k^{l+1} w_{kj}^{l+1} \sigma'(z_j^l) \qquad (2.19) \\
&:= \delta_j^l
\end{aligned}
$$

Given $\frac{\partial C}{\partial z_j^l}$ for $1 \leq l \leq L$ we can find the derivative of C w.r.t. weights and biases using the chain rule:

$$
\begin{aligned}
\frac{\partial C}{\partial w_{ji}^l} &= \frac{\partial C}{\partial z_j^l} \cdot \frac{z_j^l}{\partial w_{ji}^l} = \delta_j^l \cdot \frac{\partial \sum_p w_{jp}^l a_p^{l-1} + b_j^l}{\partial w_{ji}^l} = \delta_j^l \cdot a_i^{l-1} \\
\frac{\partial C}{\partial b_j^l} &= \frac{\partial C}{\partial z_j^l} \cdot \frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \cdot 1 = \delta_j^l
\end{aligned}
$$

A naive gradient descent algorithm would then update the parameters $\theta \in \{w_{ji}^l, b_j^l\}$ according to $\theta^{n+1} = \theta^n - \gamma \frac{\partial C}{\partial \theta}$, where γ is the *learning rate*. In reality we use *minibatches* to obtain smoother gradients: the gradient is averaged over all input vectors x in batch M of size $|M| = m$: $\theta^{n+1} = \theta^n - \frac{\gamma}{m} \sum_{x \in M} \frac{\partial C(x)}{\partial \theta}$.

2.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (LeCun et al., 1989) include layers that operate on spatial inputs. In this section we focus on convolutional layers, as other layers from CNNs, such as the pooling layer, are not used in this work. For a comprehensive overview of convolutional networks, we refer to the lecture notes of CS231n (Karpathy, 2017).

Let V be an input image of dimension $H \times W \times D$. A convolution layer slides one or more filters F_r of size M with a step size of S (the stride) along the width and height dimension of the input volume. Each filter considers the entire input volume, including the depth dimension, and produces a two-dimensional spatial ordering of output neuron activations. The depth dimension D' of the output volume of a convolution layer is simply the number of filters; its width $W' \in \mathbb{Z}^+$ and height $H' \in \mathbb{Z}^+$ depend on the stride $S \in \mathbb{Z}^+$ and potentially applied zero-padding $P \in \mathbb{Z}^+$ to accommodate for the stride:

$$
\begin{aligned}
W' &= \frac{1}{S}(W - M + 2P) + 1 \\
H' &= \frac{1}{S}(H - M + 2P) + 1
\end{aligned}
$$

The learnable parameters of filter F_r are its weights $w_{pq}^d(r)$ where $0 \leq p, q < M$ and bias $b(r) \in \mathbb{R}$. Given the activation a_{nm}^d of the input neuron at position n, m, d, the activation $z_{ij}(r)$ of the hidden neuron at position i, j, r is given by:

$$
z_{ij}(r) = \sum_{d=1}^{D} \sum_{p=0}^{M-1} \sum_{q=0}^{M-1} w_{pq}^d(r) a_{(Si+p)(Sj+q)}^d + b(r)
$$

2.4 Variational Autoencoders

Variational Autoencoders (VAEs) learn to capture a low-dimensional representation from a high-dimensional input. They are different from other so-called *latent variable models* in that the mapping from high to low-dimensional space is non-linear. Other techniques, such as Principal Component Analysis (PCA), Probabilistic Principal Component Analysis (PPCA), Factor Analysis, and

Independent Component Analysis (ICA) (Bishop, 2006) similarly aim to find a low-dimensional space, but assume that the observations are linear combinations of the low-dimensional basis vectors. VAEs are thus capable of modeling more complex relationships between the high and low-dimensional vectors.

In this section we denote the parametrisations of probability distributions with a subscript; in later sections we relax this notation and instead explicitly define the distributions and their parametrisations.

Let us assume a fully Bayesian setting where we have a vector of observed random variables $\vec{x}$, a vector $\vec{z}$ of latent variables, a generative parameter vector $\vec{\theta}$ with prior $p(\vec{\theta})$, and deterministic model parameters $\vec{\alpha}$. The corresponding graphical model is displayed in Figure 2.2.

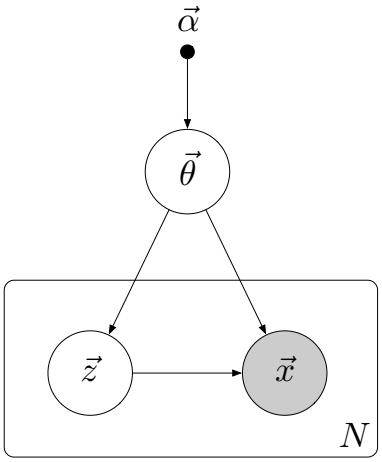


Figure 2.2: Graphical model of the Bayesian setting under consideration. This is similar to the model of a mixture of experts (Murphy, 2012, p. 343). For a comprehensive introduction to graphical models, we refer to Chapter 8 of Bishop (2006).

The complete data likelihood $p(X \mid \vec{\alpha})$, which gives the probability of the observations under some model parametrised by $\vec{\alpha}$, can be written as follows:

$$p(X \mid \vec{\alpha}) = \int_{\vec{\theta}} \int_{Z} p(X, Z, \vec{\theta} \mid \vec{\alpha}) \, dZ \, d\vec{\theta} \tag{2.20}$$

We assume that the N observations are independently and identically distributed (i.i.d.). Then the likelihood factorises over the individual samples:

$$p(X \mid \vec{\alpha}) = \prod_{i=1}^{N} \int_{\vec{\theta}} \int_{\vec{z_i}} p(\vec{x_i}, \vec{z_i}, \vec{\theta} \mid \vec{\alpha}) \, d\vec{z_i} \, d\vec{\theta} \tag{2.21}$$

For brevity, from now on we will omit the product index i from the samples. In maximum likelihood estimation, the goal is the find a model that explains the data well. We aim to find the model $\vec{a}$ that maximises the *likelihood* (probability) of the observations X under the model defined by $\vec{a}$: $p(X|\vec{a})$. Once such a model has been found, one thing we can do is perform *inference* over the latent variable $\vec{z}$: obtain a probable low-dimensional representation $\vec{z}$ from a high-dimensional observation $\vec{x}$. VAEs assume that the true posterior distribution $p(\vec{z} \mid \vec{x}, \vec{\theta})$ is intractable. Hence an approximatation of the posterior $q_\phi(\vec{z}, \vec{\theta} \mid \vec{x}) \approx p(\vec{z}, \vec{\theta} \mid \vec{x})$ with some parametrisation ϕ is introduced (Beal, 2003):

$$p(X \mid \vec{\alpha}) = \prod \int_{\vec{\theta}} \int_{\vec{z}} q_\phi(\vec{z}, \vec{\theta} \mid \vec{x}) \frac{p(\vec{x}, \vec{z}, \vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{z}, \vec{\theta} \mid \vec{x})} \, d\vec{z} \, d\vec{\theta} \tag{2.22}$$

$$= \mathbb{E}_{q_\phi(\vec{z}, \vec{\theta} | \vec{x})} \left[\frac{p(\vec{x}, \vec{z}, \vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{z}, \vec{\theta} \mid \vec{x})} \right] \tag{2.23}$$

Taking the logarithm:

$$\ln p(X \mid \vec{\alpha}) = \sum \ln \int_{\vec{\theta}} \int_{\vec{z}} q_\phi(\vec{z}, \vec{\theta} \mid \vec{x}) \frac{p(\vec{x}, \vec{z}, \vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{z}, \vec{\theta} \mid \vec{x})} \, d\vec{z} \, d\vec{\theta} \tag{2.24}$$

Since $-\ln(x)$ is convex, we can apply Jensen's inequality (Theorem 2.2.6) to Equation (2.24) in order to obtain a lower bound on the data log likelihood:

$$\begin{aligned}
\ln p(X \mid \vec{\alpha}) &\geq \sum \int_{\vec{\theta}} \int_{\vec{z}} q_\phi(\vec{z}, \vec{\theta} \mid \vec{x}) \ln \left[\frac{p(\vec{x}, \vec{z}, \vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{z}, \vec{\theta} \mid \vec{x})} \right] d\vec{z} \, d\vec{\theta} \\
&= \sum \int_{\vec{\theta}} \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) q_\phi(\vec{\theta} \mid \vec{x}) \ln \left[\frac{p(\vec{x}, \vec{z}, \vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) q_\phi(\vec{\theta} \mid \vec{x})} \right] d\vec{z} \, d\vec{\theta} \\
&= \sum \int_{\vec{\theta}} q_\phi(\vec{\theta} \mid \vec{x}) \, d\vec{\theta} \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \ln \left[\frac{p(\vec{x}, \vec{z}, \vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) q_\phi(\vec{\theta} \mid \vec{x})} \right] d\vec{z} \\
&= \sum \int_{\vec{\theta}} q_\phi(\vec{\theta} \mid \vec{x}) \, d\vec{\theta} \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \ln \left[\frac{p(\vec{x}, \vec{z} \mid \vec{\theta}, \vec{\alpha}) p(\vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) q_\phi(\vec{\theta} \mid \vec{x})} \right] d\vec{z} \\
&= \sum \int_{\vec{\theta}} q_\phi(\vec{\theta} \mid \vec{x}) \, d\vec{\theta} \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \left(\ln \frac{p(\vec{x}, \vec{z} \mid \vec{\theta}, \vec{\alpha})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})} + \ln \frac{p(\vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{\theta} \mid \vec{x})} \right) d\vec{z} \\
&= \sum \int_{\vec{\theta}} q_\phi(\vec{\theta} \mid \vec{x}) \, d\vec{\theta} \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \left(\ln \frac{p(\vec{x} \mid \vec{z}, \vec{\theta}, \vec{\alpha}) p(\vec{z} \mid \vec{\theta}, \vec{\alpha})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})} + \ln \frac{p(\vec{\theta} \mid \vec{\alpha})}{q_\phi(\vec{\theta} \mid \vec{x})} \right) d\vec{z} \\
&= \sum \int_{\vec{\theta}} q_\phi(\vec{\theta} \mid \vec{x}) \, d\vec{\theta} \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \Big(\ln p(\vec{x} \mid \vec{z}, \vec{\theta}, \vec{\alpha}) + \ln p(\vec{z} \mid \vec{\theta}, \vec{\alpha}) \\
&\quad - \ln q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) + \ln p(\vec{\theta} \mid \vec{\alpha}) - \ln q_\phi(\vec{\theta} \mid \vec{x}) \Big) \, d\vec{z}
\end{aligned} \tag{2.25}$$

This lower bound is referred to as the Evidence Lower Bound (ELBO). If there is no prior on θ, we get instead:

$$\begin{aligned}
p(X \mid \vec{\theta}) &= \prod_i \int_{\vec{z_i}} p(\vec{x_i}, \vec{z_i} \mid \vec{\theta}) \\
&= \prod_i \int_{\vec{z_i}} q_\phi(\vec{z_i} \mid \vec{\theta}, \vec{x_i}) \frac{p(\vec{x_i}, \vec{z_i} \mid \vec{\theta})}{q_\phi(\vec{z_i} \mid \vec{\theta}, \vec{x_i})} \, d\vec{z} \\
\ln p(X \mid \vec{\theta}) &= \sum \ln \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \frac{p(\vec{x}, \vec{z} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})} \, d\vec{z} \qquad (2.26) \\
&\geq \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \ln \frac{p(\vec{x}, \vec{z} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})} \\
&= \sum \mathbb{E}_{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})} \left[\ln \frac{p(\vec{x}, \vec{z} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})} \right] \\
&= \sum \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \ln \frac{p(\vec{x} \mid \vec{z}, \vec{\theta}) p(\vec{z} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})} \, d\vec{z} \\
&= \sum \int_{\vec{z}} q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \Big(\ln p(\vec{x} \mid \vec{z}, \vec{\theta}) + \ln p(\vec{z} \mid \vec{\theta}) - \ln q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \Big) \, d\vec{z} \qquad (2.27)
\end{aligned}$$

Using the **Kullback-Leibler divergence** between the approximate and true posteriors, we can show the exact relationship between the log marginal likelihood and its lower bound (Equation (2.27)):

$$D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta}, \vec{x})) = \mathbb{E}_q \left[\ln \frac{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})}{p(\vec{z} \mid \vec{\theta}, \vec{x})} \right]$$

$$
\begin{aligned}
&= \mathbb{E}_q\Big[\ln q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) - \ln p(\vec{z} \mid \vec{\theta}, \vec{x})\Big] \\
&= \mathbb{E}_q\Big[\ln q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) - \ln p(\vec{z}, \vec{x} \mid \vec{\theta}) - \ln p(\vec{x} \mid \vec{\theta})\Big] \\
&= \mathbb{E}_q\Big[\ln q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) - \ln p(\vec{z}, \vec{x} \mid \vec{\theta})\Big] - \ln p(\vec{x} \mid \vec{\theta}) \\
-D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta}, \vec{x})) &= -\mathbb{E}_q\left[\ln \frac{p(\vec{z}, \vec{x} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})}\right] + \ln p(\vec{x} \mid \vec{\theta}) \\
\ln p(x \mid \vec{\theta}) &= -D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta}, \vec{x})) + \mathbb{E}_q\left[\ln \frac{p(\vec{z}, \vec{x} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})}\right]
\end{aligned}
$$

$$
\begin{aligned}
\ln p(X \mid \vec{\theta}) &= \sum \ln p(x \mid \vec{\theta}) \\
&= \sum -D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta}, \vec{x})) + \sum \mathbb{E}_q\left[\ln \frac{p(\vec{z}, \vec{x} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})}\right] \qquad (2.28)
\end{aligned}
$$

The difference between Equation (2.26) and Equation (2.27) becomes apparent in Equation (2.28): it is the KL-divergence between approximate and true posteriors q and p.

Using Bayes rule again, Equation (2.28) can be further reduced:

$$
\begin{aligned}
\ln p(X \mid \vec{\theta}) &= \sum -D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta}, \vec{x})) + \sum \mathbb{E}_q\left[\ln \frac{p(\vec{x} \mid \vec{z}, \vec{\theta}) p(\vec{z} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})}\right] \\
&= \sum -D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta}, \vec{x})) + \sum \mathbb{E}_q\left[\ln \frac{p(\vec{z} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})} + \ln p(\vec{x} \mid \vec{z}, \vec{\theta})\right] \\
&= \sum -D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta}, \vec{x})) + \sum \mathbb{E}_q\left[\ln \frac{p(\vec{z} \mid \vec{\theta})}{q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})}\right] + \sum \mathbb{E}_q\Big[\ln p(\vec{x} \mid \vec{z}, \vec{\theta})\Big] \\
&= -\sum D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta}, \vec{x})) - \sum D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta})) \\
&\quad + \sum \mathbb{E}_q\Big[\ln p(\vec{x} \mid \vec{z}, \vec{\theta})\Big] \\
&\geq \sum \mathbb{E}_q\Big[\ln p(\vec{x} \mid \vec{z}, \vec{\theta})\Big] - \sum D_{KL}(q_\phi(\vec{z} \mid \vec{\theta}, \vec{x}) \,||\, p(\vec{z} \mid \vec{\theta})) \qquad (2.29)
\end{aligned}
$$

The first term is referred to as the expected *reconstruction* error, since it is the expected value of the input with regard to the posterior distribution over the latent variable $\vec{z}$. If the expectations in $\ln p(X \mid \vec{\alpha})$ cannot be solved analytically, they could be estimated using a Monte Carlo estimator (see Theorem 2.2.4).

In Figure 2.3 a simple Variational Autoencoder is pictured. The network on the left is called the encoder and implements the parametrisation ϕ of the approximate posterior distribution $q_\phi(\vec{z} \mid \vec{x})$. The network on the right is the decoder network and implements the parametrisation of the distribution $p(\vec{x} \mid \vec{z}, \vec{\theta})$. The decoder relies on $\vec{z}$, which is sampled from the distribution $q_\phi(\vec{z} \mid \vec{x})$. The *reparametrisation trick* is a function $f(\vec{\epsilon}, \phi)$ over the parametrisation ϕ of the posterior distribution over $\vec{z}$ and an auxiliary random variable $\vec{\epsilon}$. It allows to backpropagate weights and biases from the network's output layer to the input layer, despite the presence of a probabilistic node sampling $\vec{z}$ in between.

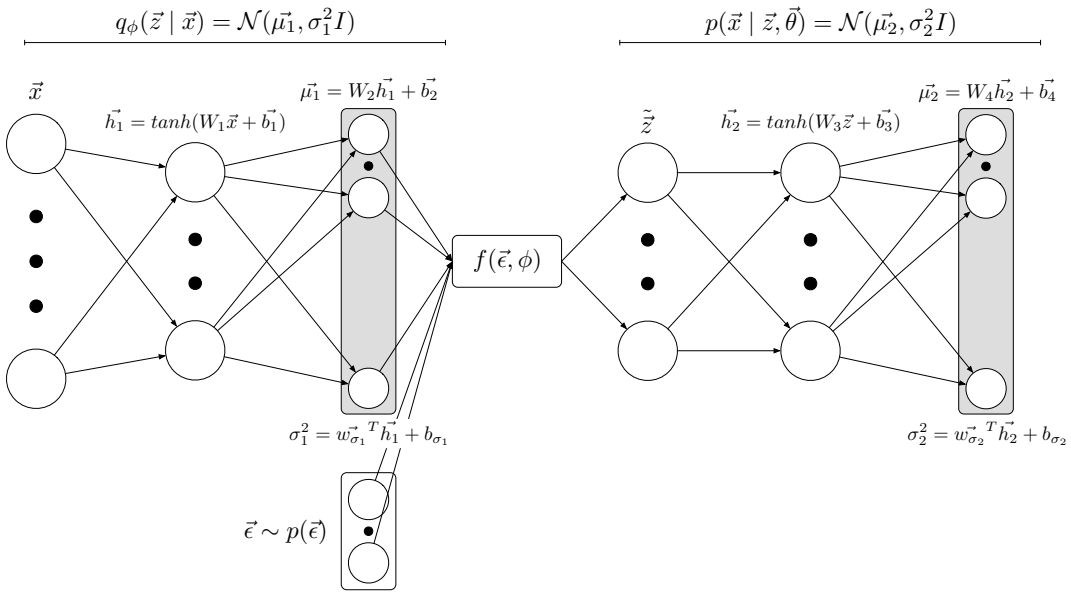


Figure 2.3: A simple Variational Autoencoder with no prior on $\vec{\theta}$. The left part is the approximate posterior $q_\phi(\vec{z} \mid \vec{x})$, also called an *encoder*: it encodes $\vec{x}$ into a latent code $\vec{z}$. The differentiable transformation $g_\phi(\vec{\epsilon}, \vec{x})$ yields the input to the *decoder* $p(\vec{x} \mid \vec{z}, \vec{\theta})$. To find the optimal parameters $\vec{\theta}$ and $\vec{\phi}$ that maximise the likelihood of $\vec{x}$ we can perform gradient descent using backpropagation, where $\phi = \{W_1, W_2, \vec{w_{\sigma_1}}, \vec{b_1}, \vec{b_2}, b_{\sigma_1}\}$ and $\vec{\theta} = \{W_3, W_4, \vec{w_{\sigma_2}}, \vec{b_3}, \vec{b_4}, b_{\sigma_2}\}$.

Example 2.4.1 (Reparametrisation trick)**.** Suppose $\vec{z}$ is Gaussian distributed with mean $\vec{\mu}$ and diagonal covariance matrix $\text{diag}(\vec{\sigma}^2)$. Then if we choose $\vec{\epsilon} \sim \mathcal{N}(0, I)$ and differentiable transformation $\vec{z} = \mu + \vec{\sigma} \odot \vec{\epsilon}$, then as a consequence of Theorem 2.2.1, $\vec{z}$ is multivariate normal distributed with mean $\vec{\mu}$ and covariance matrix $\text{diag}(\sigma)\text{diag}(\sigma)^T = \text{diag}(\sigma)^2$, which is the original distribution of $p(\vec{z})$. In other words, a differentiable sample from $p(\vec{z})$ was created by sampling from an auxiliary distribution $p(\vec{\epsilon})$. A comprehensive overview of distributions and suitable reparametrisation transformations is given in Kingma and Welling (2013).

Chapter 3

Project overview

In this chapter, we provide an overview of the project as a whole, and expand on the individual components that are required for the final deliverable. It will become clear how the remaining chapters fill in on each component.

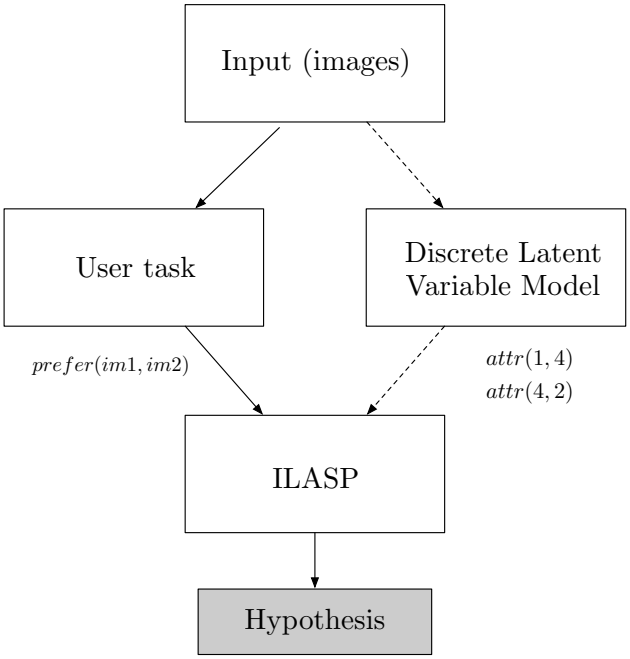


Figure 3.1: The final learning framework as outlined in Chapter 1.

Recall that in Chapter 1, we devised a learning setup based on a high-level flow chart of human reasoning. See Figure 3.1. In the process leading to the final product, the learning framework was broken down into smaller parts to verify the correctness and viability of each part. Initially, the learning setup involved a purely symbolic task called the Monk's problem. The idea was that if ILASP could solve the various tasks bundled in the Monk's problems, then the final stage of the larger hybrid neural/symbolic framework has been verified to work. What is left then is to verify that the discrete latent variable model (Chapter 4) manages to discover meaningful visual factors with high accuracy (Chapter 5), and that those factors contribute to explaining the outcome of a user task (e.g. preference learning) (Chapter 6). In the final product, the symbolic Monk's problem task is replaced by a task consisting of input decomposed into low-level concepts given by the discrete latent variable model, and labels assigned to those inputs. We now briefly recap the Monk's problem and show how it can be solved with ILASP, then settle on a learning task that will form the proof-of-concept application of the learning framework, and finally discuss the requirements of the discrete latent variable model. Sections 3.1-3.3 concern background research, while Section 3.4 leads to the novel work introduced in Chapters 4-6.

3.1 Monk's problems

In the Monk's problems (Thrun et al., 1991), the problem domain is a set of robots that each have six attributes and a positive or negative class label. The six attributes are `head_shape`, `body_shape`, `is_smiling`, `holding`, `jacket_color`, and `has_tie`. Depending on the variation of the problem being studied (Monk's 1, 2, or 3), the classifications of each robot are provided by a particular set of logic rules. For example, in Monk's 1, a positive classification requires the head and body shapes to be equal or the jacket color to be red. In Monk's 3, a positive classification is contingent upon a green jacket color and holding a sword, or a jacket color that is not blue and body shape that is not octagon. Monk's 3 additionally includes noise: some of the given classifications are wrong. The aim is, given the classes for a set of robots, to uncover the rule that determines to which class a robot belongs. In the final learning framework, one can expect noise to be present as well, as user-provided labels may not be consistent. Thus, Monk's 3 forms a valid task to solve before applying ILASP in the final setting. A Learning from Answer Sets (LAS) task solved by ILASP returns the correct hypothesis for each problem attempted; Monk's 1 and 3. The language bias and background knowledge for each task are given in Appendix D. For task 3, the noise weight assigned to each example is 3. Task 1 was solved without noise. It was confirmed that the same tasks can be performed by a multi-layer perceptron as well, with equivalently 100% accuracy on the test set. The issue of obscurity however rises again: while ILASP has provided us with a human-readable logic rule explaining the input to the learning algorithm, the neural net is a black box: it does not explain *how* it determines the label of each robot.

3.2 The learning task

In the left path in Figure 3.1, input images are given to a user who then performs a classification task on those images. The outputs are the class labels. The task pictured to be ideal for validation of the framework was the classification of fruits, a task more frequently mentioned in VAE literature (Higgins et al., 2017a,c). It is easily seen that different classes of fruits share the same visual factors, which together determine the fruit class. Examples are color, size, and shape. It would be a perfect application of the learning framework to uncover the factors that determine the fruit class that is perceived by humans. Unfortunately, VAEs, which will implement the discrete latent variable component in Figure 3.1, require large amounts of training data. Ready available datasets such as CIFAR-100 (Krizhevsky and Hinton, 2009) do not contain a sufficient number of fruits. Several thousands of fruits were scraped from Bing, Google Images and Flickr, but most were deemed unsuitable because they deviate in one way or another from the desired real-world image of a single fruit. It has been considered to photograph several thousands of fruits at a wholesaler, but this idea was dropped due to time constraints. Different datasets of a suitable single object class with its variations were considered. Faces (Liu et al., 2015) and Chairs (Aubry et al., 2014) are large datasets of a single object class (a face and a chair) with variations within. Faces however involves photographs of real human faces and thus quickly introduces an ethical barrier. Chairs are more suitable, but the dataset does not distinguish any classes within the chairs. Hence, we settled on a preference learning task: a user is presented with a pair of chairs and indicates which one they prefer. Once can imagine that preferences are decided upon based on low-level concepts and their variations present across all inputs. The goal is then to find weak constraints that describe the recorded preferences in terms of a number of visual concept classes (Chapter 6).

3.3 Understanding VAEs

As mentioned in Chapter 1, we have chosen to base the extraction of concepts from images on a particular formulation of VAEs called β-VAE (Higgins et al., 2017a). β-VAE has been chosen because it is the current state-of-the-art in disentangled factor learning, outperforming other recent GAN-based techniques such as DC-IGN (Kulkarni et al., 2015) and InfoGAN (Chen et al., 2016). Before we see how β-VAE fits in the final framework, it is crucial to understand how β-VAE and

VAEs in general can be used to infer visual concepts from input images. We note that VAEs are a highly active area of research. New insights are rapidly developed, improving upon previously established findings. For example, Higgins et al. (2017b) found that Denoising Autoencoders (DAE) (Vincent et al., 2010) improve on input reconstruction, and Higgins et al. (2017c) was published while this work was well underway.

3.3.1 Interpreting the low-dimensional representation

In practise, it turns out that the low-dimensional representation $\vec{z}$ obtained by the encoder network of a VAE encodes visual factors present in the inputs X (Higgins et al., 2017a). In Figure 3.2 and Figure 3.3 we take a closer look at the boundary between encoder and decoder in order to understand how the latent code $\vec{z}$ can be interpreted in terms of visual factors. As mentioned before, the encoder network *parametrises* the approximate posterior distribution $q(\vec{z} \mid \vec{x})$. In the particular case of Figure 3.2 its output layers $\vec{\mu}$ and $\vec{\sigma}^2$ together parametrise a multivariate Gaussian distribution. It is assumed that $q(\vec{z} \mid \vec{x})$ has a diagonal covariance matrix; the diagonal components are given by $\vec{\sigma}^2$. In a diagonal covariance matrix the off-diagonal elements are zero; there is assumed no covariance between individual components of $\vec{z}$. As a result, each scalar component z_i of $\vec{z}$ is univariate Gaussian distributed with mean μ_i and variance σ_i^2. From the parametrisation of $q(\vec{z} \mid \vec{x})$ we obtain a differentiable sample $\vec{z} \sim q(\vec{z} \mid \vec{x})$ using the reparametrisation trick. Most samples will be centered around the mean vector $\vec{\mu}$ by definition of a Gaussian distribution.

The encoder essentially is a function mapping an observation $\vec{x}$ to the parametrisation of $q(\vec{z} \mid \vec{x})$. The KL-divergence term in the VAE objective function (Equation (2.29)) aims to match $q(\vec{z} \mid \vec{x})$ for each $\vec{x} \in X$ as close as possible to a predefined prior $p(\vec{z})$. If for an observation $\vec{x}$ and a component i of z_i the posterior deviates from the assigned prior $p(z_i)$, it means that the VAE was unable to default the posterior to the prior and had to accommodate for variations in the input data. These variations are due to the latent visual factors; in this case z_i is a component encoding a visual factor present in the inputs X. To visualise the inferred quantities of variation, we can input $\vec{\mu}$ directly to the decoder network to obtain a deterministic reconstruction of the input. Across the entire dataset, the posterior over the components z_i that encode visual factors will be approximately matching the prior $p(z_i)$. Hence, by modifying μ_i over the support interval of the prior, we can iterate over the visual factor. A common choice of prior $p(\vec{z})$ is $\mathcal{N}(0, I)$ (e.g. Kingma and Welling (2013); Higgins et al. (2017a); Kingma et al. (2014)).

Example 3.3.1 (Inferring visual factors)**.** Assume a VAE setting with prior $p(\vec{z}) = \mathcal{N}(0, I)$. Suppose we obtain $\vec{\mu}$ and $\vec{\sigma}^2$ for an input image $\vec{x}$. Component i of $\vec{z}$ turns out to be univariate Gaussian distributed with a variance lower than 1, and a mean around -2 (i.e., it is relatively dissimilar from the unit Gaussian prior). We do the following for each $\tilde{z} \in [-3, -2, -1, 0, 1, 2, 3]$, which corresponds to the 99.7% confidence interval of a unit Gaussian:

- Change μ_i to $\tilde{z}$ while keeping all other components of $\vec{\mu}$ fixed.
- Input the altered mean vector to the decoder network to obtain a reconstruction from the observation $\vec{x}$.

For each $\tilde{z}$, in the corresponding reconstruction we will notice a change in appearance of the visual factor encoded by z_i. For example, if z_i encoded background color, the obtained set of reconstructions could form a transition from a white to a black background (or reversed). Disentanglement performance can be measured by observing changes in other visual factors in the reconstruction. A good disentanglement performance results in just the background of the reconstruction changing, while all other visual aspects remain unchanged.

3.3.2 β-VAE

In any variational setting, the marginal log-likelihood lowerbound can be decomposed into a KL-divergence term and an expected reconstruction error term (see Equation (2.29)). The value of

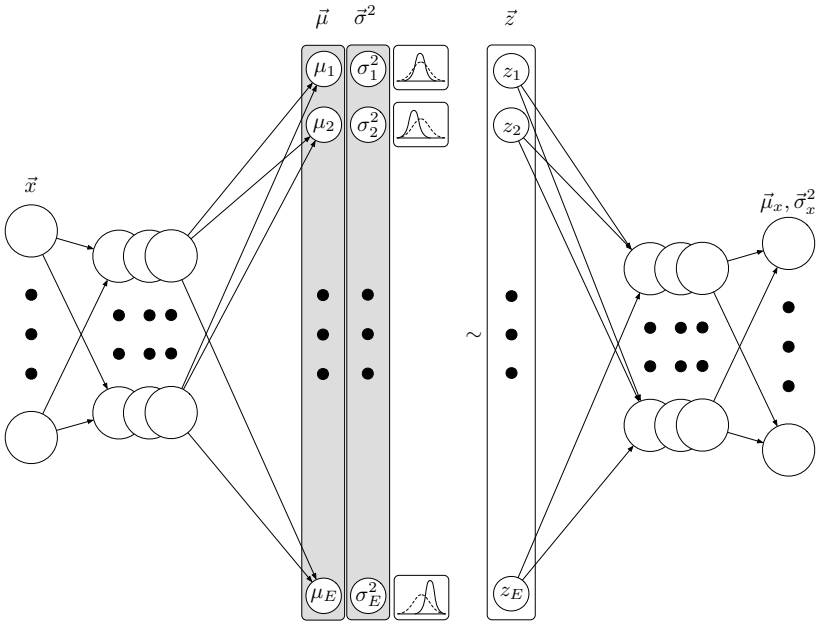
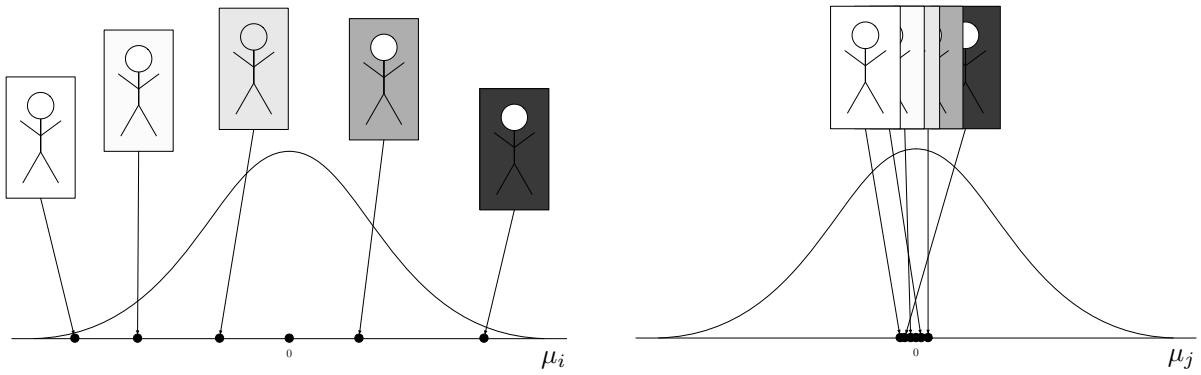


Figure 3.2: A closer look at a VAE and its latent channel $\vec{z}$. In this particular example the output of the decoder network is a multivariate Gaussian distribution, which can be visualised by plotting the mean $\vec{\mu}_x$. For an input image $\vec{x}$, the mean vector $\vec{\mu}_x$ conforms to the reconstructed input. The dotted Bell curves indicate the prior unit Gaussian distribution over $\vec{z}$, the solid curves indicate the inferred posterior distribution for a certain input $\vec{x}$.



(a) Latent encoding background tint (light-dark).

(b) Noninformative latent.

Figure 3.3: Inferred means given input images $\vec{x}$ of an informative component i of $\vec{z}$ (left) vs. a noninformative component (right). The inferred posterior mean of noninformative components, i.e. components not encoding a visual factor, is closer to the prior mean of 0. The inferred means of informative latents are more spread out. The images at the top are images input to the encoder network. The Bell curve is a schematic depiction of the standard Gaussian with mean 0 and variance 1, not to scale.

the KL-divergence term indicates how well the approximate posterior distributions match the prior distributions. In the case of one latent random variable, it was found that adding a constraint on the KL-divergence term $D_{KL}(q \,||\, p)$ between approximate and true posteriors $q_\phi(\vec{z} \mid \vec{\theta}, \vec{x})$ and $p(\vec{z} \mid \vec{\theta})$ improves disentanglement performance in the latent channel $\vec{z}$, at a cost of a lower reconstruction fidelity (Higgins et al., 2017a). The optimal value for this β constraint depends on the application. It can be determined using a disentanglement metric or through visual inspection (Higgins et al., 2017a).

3.4 The discrete latent variable model

The visual concepts encoded by the univariate standard normal distributions are not directly usable in an ILP context. The most common concept classes are centered around the mean, while concepts that occur less frequent in the input data are encoded further away from the mean. In order to perform logical inference, the continuous values under the curve need to be discretised to concrete concept classes. One, naive possibility is to split the support of the univariate Gaussian (the interval $[-3, 3]$) into equal-sized bins, and assign a concept class label to each bin. We refer to this technique as **dβ-VAE**, *discretised* β-VAE. Such a discretisation is not ideal, especially in case where a concept has three or more classes. They may all occur equally frequently in the input data, but if the curve is split into three equal sized intervals, the cumulative probabilities covered by the intervals are not equal, causing the labels to be misaligned with the actual occurrence of each label. However, in the case of two concept classes that are estimated to occur equally frequently, splitting the curve into two halves at the mean is expected to result in a good match between ground truth labels and assigned labels due to the inherent symmetry of the Gaussian distribution (see Figure 3.4). Hence, the performance of a $d\beta$-VAE classifier of visual concepts in the 2-class case is considered the baseline for any other statistical model to be introduced in the next chapters. Its performance is only dependent on the suitability of VAEs in general for extracting visual factors, which was demonstrated by Higgins et al. (2017a).

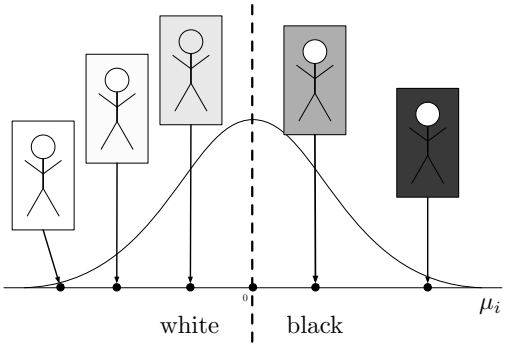


Figure 3.4: $d\beta$-VAE with 2 categories applied to the curve from Figure 3.3a encoding background color.

The remainder of this work is organised as follows. In Chapter 4, two alternatives to $d\beta$-VAE are derived that are inherently discrete. The first model has one latent random variable approximately discretely distributed that generates the observations. It is expected that his model is not powerful enough, due to the fact that a restricted discrete range of values will not be capable of capturing as much variation as a continuous range. The second model is an adaptation of the default model in VAE setting. Variations in the components of the latent vector $\vec{z}$ are modelled as Gaussian mixtures: each component of $\vec{z}$ is sampled from a univariate Gaussian mixture model, where each mixture component is expected to cover a different visual concept class. The idea is that if a visual factor can be modelled by a Gaussian distribution, then individual concept classes can be as well. In Chapter 5, the two models and slight variations on them are evaluated on two different datasets. A list of disired properties of the final model is devised: it should have a number of visual factors discovered comparable with the state-of-the-art β-VAE, a comparable

disentanglement performance, a classification accuracy comparable in the 2-category case with $d\beta$-VAE, and does not require human intervening if applied to the final learning framework. In Chapter 6, the statistical model that performed best is integrated with a preference learning task to demonstrate the performance of the final, complete framework.

Chapter 4

Derivation of statistical models

In this chapter, our aim is to find an inherently discrete latent variable model to compete with $d\beta$-VAE, as in the previous chapter we mentioned that $d\beta$-VAE is not ideal. We claim two models are worth investigating. The first model is a naive discrete variant of the standard 2-variable model. We hypothesise that the model is not powerful enough to capture the latent structure of the observations, due to its simplistic and discrete nature. It is worth investigating since there is no mention of such a model in the literature in the context of disentangled factor learning. The second model is identical to the standard VAE model, but introduces a third, discrete random variable controlling the assignment of a continuous latent variable to univariate clusters. The latent channel remains multivariate Gaussian distributed, but the variations within the individual components are grouped together on the real line in Gaussian-shaped concentrations. A third, discrete random variable determines the mixture from which a component is sampled, and as such defines the desired "categories" within visual factors.

4.1 Discrete Model

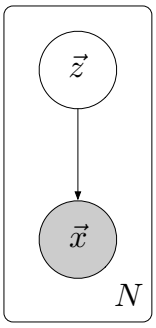


Figure 4.1: Generative graphical model under consideration.

We reiterate the standard graphical model in the VAE setting from Section 2.4 in Figure 4.1. A common assumption in this setting is that $\vec{z}$ is a continuous variable, whose posterior and prior distributions are both multivariate Gaussians (e.g. Kingma and Welling (2013); Higgins et al. (2017a); Kingma et al. (2014)). The prior $p(\vec{z})$ is often chosen to be the unit Gaussian $\mathcal{N}(0, I)$.

In this section, we consider a model where $\vec{z}$ is a discrete random variable. Each component of $\vec{z}$ is an integer indicating the selected category for that component. The reparametrisation trick, however, is not defined for discrete distributions. It was independently shown that a continuous approximation to the categorical distribution is an effective and efficient way of modelling a discrete probability distribution (Jang et al., 2016; Maddison et al., 2016). Since this Gumbel-Softmax distribution is a transformation of the Gumbel distribution for which the reparametrisation trick is defined, we can use the Gumbel-Softmax distribution to model the approximate posterior distribution $q(\vec{z} \mid \vec{x})$. A brief overview of the Gumbel-Softmax distribution follows.

4.1.1 The Gumbel-Softmax distribution

The Gumbel-Softmax distribution is based on the Gumbel-Max trick. The Gumbel-Max trick adds random noise from a Gumbel distribution to unnormalised class probabilities, and takes the argmax to obtain a discrete sample from the categorical distribution parametrised by the unnormalised class probabilities (Jang et al., 2016). Differentiability is ensured by approximating the argmax function using the softmax function with temperature hyperparameter τ. The probability density function (pdf) of the Gumbel-Softmax distribution is obtained by transforming the pdf of the Gumbel-distributed random noise variable, the details of which can be found in Jang et al. (2016). Hyperparameter τ controls the uniformity of the one-hot Gumbel-Softmax samples. A small value of τ results in nearly one-hot samples, while a larger value of τ yields a uniformly distributed sample. A large τ additionally negatively affects the variance of the gradients. An annealing schedule is suggested to decrease the value of τ as training progresses: every N steps, τ is updated according to:

$$\tau = \max(0.5, \exp(-rt)) \tag{4.1}$$

where r is the rate of descend and t is the global training step. We choose $r = 1 \times 10^{-5}$ and $N = 1000$ (Jang et al., 2016).

4.1.2 Prior belief

We can now define the distributions over the latent variable $\vec{z}$ in the revised, (approximately) discrete model:

$$p(\vec{z}) = \text{Gumbel-Softmax}(\Pi, K) \tag{4.2}$$
$$q(\vec{z} \mid \vec{x}) = \text{Gumbel-Softmax}(\Phi, K) \tag{4.3}$$

Since $\vec{z}$ is technically a continuous distribution, we cannot use a discrete prior but have to use the Gumbel-Softmax there as well. The prior class probabilities are given by $\Pi \in \mathbb{R}^{E \times K}$, the posterior class probabilities by $\Phi \in \mathbb{R}^{E \times K}$. E defines the number of components of $\vec{z}$, K defines the number of categories per component. The distribution over $\vec{x}$ depends on the nature of the input data, e.g. in case of greyscale inputs a common choice is a Bernoulli distribution, and in case of 3-channel RGB inputs a common choice is to model each channel by one multivariate Gaussian distribution (e.g. Higgins et al. (2017a)).

Before this model can be used in practise, the prior class probabilities Π remain to be decided on. In a perfectly balanced world, all visual concepts are equally likely to occur. Such belief is embodied by a uniform prior. If we encode $\vec{z}$ as a one-hot matrix $Z \in \mathbb{R}^{E \times K}$, where each row $\vec{z_i} \in \mathbb{R}^K$ is a one-hot encoding of the category selected for the i-th component of $\vec{z}$, then the uniform prior $p(Z)$ is defined as:

$$p(Z) = \overbrace{\begin{bmatrix} \frac{1}{K} & \cdots & \frac{1}{K} \\ \vdots & & \vdots \\ \frac{1}{K} & \cdots & \frac{1}{K} \end{bmatrix}}^{K} \tag{4.4}$$

where K is the number of categories, and E the number of latents. A one-hot vector is a vector of 0s and 1s of length K where $K - 1$ values are equal to 0, and precisely one value is 1. It can be used to represent categorical assignments, which simplifies calculations in mixture models (e.g. see Bishop (2006, p. 431)). In that case the component equalling 1 corresponds to the selected category.

In practise, however, many quantities tend to be normally distributed (Lyon, 2013). We could formulate the prior such that the prior probabilities correspond to support intervals of a univariate unit Gaussian $\mathcal{N}(0, 1)$. Such a prior would, compared to the $\mathcal{N}(0, I)$ prior that was found to work well in Higgins et al. (2017a), give us a shape similar to the standard Bell curve, but it lacks the

statistical independence embodied by a diagonal covariance matrix. Despite this shortcoming, we argue it is of interest to pursue the Gaussian-shaped prior to see if its disentanglement performance approaches the baseline set by β-VAE or not. The cumulative distribution function (CDF) of a univariate standard normal distribution is given by:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left[-\frac{t^2}{2}\right] dt \tag{4.5}$$

The values within 3 standard deviations of the mean account for 99.7% of all values drawn from the distribution according to the three sigma rule (Hazewinkel, 2001). If we split this area into 5 equal-sized intervals, we obtain $[-3, -1.8)$, $[-1.8, -0.6)$, $[-0.6, 0.6)$, $[0.6, 1.8)$, $[1.8, 3]$. Expanding to the full support of a Gaussian yields $[-\infty, -1.8)$, $[-1.8, -0.6)$, $[-0.6, 0.6)$, $[0.6, 1.8)$, $[1.8, +\infty]$. Using the Gaussian CDF we can calculate the probability associated with each interval, and obtain, subject to rounding:

$$p(Z) = \begin{bmatrix} 0.036 & 0.2385 & 0.451 & 0.2385 & 0.036 \\ & & \vdots & & \\ 0.036 & 0.2385 & 0.451 & 0.2385 & 0.036 \end{bmatrix} \tag{4.6}$$

We will refer to this prior as the *Gaussian-CDF-like* prior, or GCDF in short.

4.1.3 The ELBO

The model remains unchanged from the original formulation in Section 2.4, hence the evidence lower bound (ELBO) $\mathcal{L}$ remains as previously shown:

$$\begin{aligned} \mathcal{L} &= \int_{\vec{z}} q(\vec{z} \mid \vec{x})(\ln p(\vec{z}) + \ln p(\vec{x} \mid \vec{z}) - \ln q(\vec{z} \mid \vec{x})) \\ &= -D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z})) + \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] \end{aligned} \tag{4.7}$$

The same model is described in Asai and Fukunaga (2017), but is applied to a different context. Additionally, it is restricted to a uniform prior, and does not consider the impact of the β-constraint in the VAE objective. Since the 2-variable model where the approximate posterior over the latent variable is modelled by a Gumbel-Softmax distribution has not been attempted in a disentangled factor learning setting before, we argue it forms a valuable contribution to the literature. Apart from two different priors, we will also evaluate the model introduced in this section with different values of β.

4.2 Univariate Gaussian Mixtures

In this section, we expand the standard graphical model from Section 2.4 with an additional, discrete random variable $\vec{y}$. The idea is that each component of $\vec{y}$ selects a mixture component from a univariate Gaussian mixture model. Depending on the chosen mixture components, the latent code $\vec{z}$ is sampled component-wise from univariate Gaussians. By Corollary 2.2.1, this vector itself is multivariate Gauss distributed, where the mean of the distribution is given by the means of the assigned clusters. The covariances between the individual components are 0 and the covariance matrix diagonal is given by the variances of the selected mixture components, which are assumed significantly smaller than 1. Within a component of $\vec{z}$, the univariate Gaussians cluster variations in the input data together on the real line - we hypothesise that these variations capture the visual concepts present within a visual factor. For example, if a component i of $\vec{z}$ encodes the factor *size*, then the mixture components for z_i may encode *small*, *medium*, and *large* and the small variations within those classes across the entire dataset. Using posterior inference over the discrete vector $\vec{y}$ we can obtain the cluster assignment for each component of $\vec{z}$, and as such obtain the inferred visual categories. In Figure 4.2 the real line with 4 Gaussians forming a mixture of Gaussians (red)

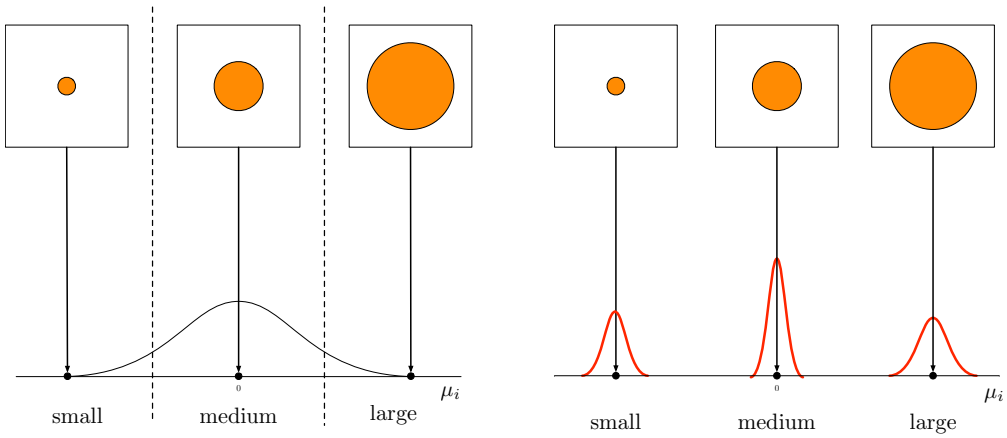


Figure 4.2: A single Gaussian encoding the size of a fruit (left) v.s. a mixture of Gaussians (right) where each mixture constitutes a different size class. The dotted lines in the figure on the left indicate a possible discretisation of the continuous visual factor *size*.

and the original, single Gaussian distribution (black) is displayed. Figure 4.3 shows the graphical model in the new setting.

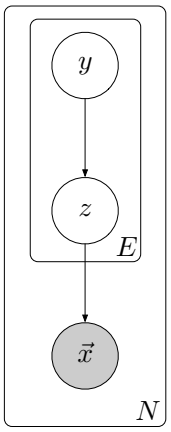


Figure 4.3: The expanded generative graphical model, where E is the number of latents and N the number of samples. y and z are scalars.

4.2.1 The ELBO

We now define the prior and approximate posterior distributions in the new setting. Let Y be a matrix where each row y_i is the one-hot encoding of the discrete category selected for the i-th component of $\vec{z}$, and let $M^{E\times K}$ and $\Sigma^{E\times K}$ be matrices holding the means and variances of $E \times K$ univariate Gaussians, where $\vec{z} \in \mathbb{R}^E$ and K is the number of categories per latent. Then:

$$p(\vec{y}) = \text{Cat}(\Pi^{E\times K}) \tag{4.8}$$

$$p(\vec{z} \mid \vec{y}) = \mathcal{N}((Y^T M)\vec{1}, (Y^T \Sigma)\vec{1}) \tag{4.9}$$

$$p(\vec{x} \mid \vec{z}) = \mathcal{N}(\vec{\mu}_{p_x}, \text{diag}(\vec{\sigma}^2_{p_x})) \tag{4.10}$$

This generative process is similar to the processes described in Dilokthanakul et al. (2016) and in particular Jiang et al. (2016). A fundamental difference, however, is that both assume a multivariate mixture model, as opposed to the univariate (component-wise) model under consideration here. Jiang et al. (2016) showed an efficient way of inferring the posterior distribution $q(\vec{y} \mid \vec{x})$ in case the joint posterior distribution over latent variables $\vec{y}$ and $\vec{z}$ is assumed to factorise over the latents: $q(\vec{z}, \vec{y} \mid \vec{x}) \approx q(\vec{z} \mid \vec{x})q(\vec{y} \mid \vec{x})$. This is called a mean-field approximation. The contribution from Jiang et al. (2016) results in a more efficient model, because it does only require a variational

approximation of $q(\vec{z} \mid \vec{x})$ and $p(\vec{x} \mid \vec{z})$: they show that $q(\vec{y} \mid \vec{x})$ can be calculated analytically. We first show that their observation also applies to the array of univariate mixture models introduced in this section, and formulate a second derivation that does require three variational approximations but allows for posterior inference over the mixture components, which plays a key role in automatically identifying descriptive latents.

Mean-field approximation

Let us assume the mean-field approximation $q(\vec{z}, \vec{y} \mid \vec{x}) \approx q(\vec{z} \mid \vec{x})q(\vec{y} \mid \vec{x})$, and the following approximate posterior:

$$q(\vec{z} \mid \vec{x}) = \mathcal{N}(\vec{\mu}_{q_z}, \text{diag}(\vec{\sigma}^2_{q_z})) \tag{4.11}$$

The ELBO is given by:

$$\begin{aligned}
p(X) \geq & \sum_{\vec{y}} \int_{\vec{z}} q(\vec{z}, \vec{y} \mid \vec{x}) \ln \left[\frac{p(\vec{x}, \vec{y}, \vec{z})}{q(\vec{z}, \vec{y} \mid \vec{x})} \right] d\vec{z} \\
= & \sum_{\vec{y}} \int_{\vec{z}} q(\vec{z}, \vec{y} \mid \vec{x}) \ln \left[\frac{p(\vec{y})p(\vec{z} \mid \vec{y})p(\vec{x} \mid \vec{z})}{q(\vec{z}, \vec{y} \mid \vec{x})} \right] d\vec{z} \\
= & \sum_{\vec{y}} \int_{\vec{z}} q(\vec{z}, \vec{y} \mid \vec{x}) \ln \left[\frac{p(\vec{z}, \vec{y})p(\vec{x} \mid \vec{z})}{q(\vec{z}, \vec{y} \mid \vec{x})} \right] d\vec{z} \\
= & \mathbb{E}_{q(\vec{z},\vec{y}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] - D_{KL}(q(\vec{z}, \vec{y} \mid \vec{x}) \,||\, p(\vec{z}, \vec{y})) \qquad (4.12) \\
= & \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] - \sum_{\vec{y}} \int_{\vec{z}} q(\vec{z}, \vec{y} \mid \vec{x}) \ln \left[\frac{p(\vec{y})p(\vec{z} \mid \vec{y})}{q(\vec{z}, \vec{y} \mid \vec{x})} \right] d\vec{z} \\
\approx & \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] - \sum_{\vec{y}} \int_{\vec{z}} q(\vec{z} \mid \vec{x})q(\vec{y} \mid \vec{x}) \ln \left[\frac{p(\vec{y})p(\vec{z} \mid \vec{y})}{q(\vec{z} \mid \vec{x})q(\vec{y} \mid \vec{x})} \right] d\vec{z} \\
= & \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] - \sum_{\vec{y}} \int_{\vec{z}} q(\vec{z} \mid \vec{x})q(\vec{y} \mid \vec{x}) \ln \left[\frac{p(\vec{z} \mid \vec{y})}{q(\vec{z} \mid \vec{x})} \right] d\vec{z} - D_{KL}(q(\vec{y} \mid \vec{x} \,||\, p(\vec{y})) \\
= & \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] - \mathbb{E}_{q(\vec{y}|\vec{x})}[D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z} \mid \vec{y}))] - D_{KL}(q(\vec{y} \mid \vec{x} \,||\, p(\vec{y})) \qquad (4.13)
\end{aligned}$$

By rewriting the ELBO using Bayes' rule, Jiang et al. (2016) showed that in the optimum, $q(\vec{y} \mid \vec{x}) = p(\vec{y} \mid \vec{z})$ holds:

$$\begin{aligned}
p(X) \geq & \sum_{\vec{y}} \int_{\vec{z}} q(\vec{z}, \vec{y} \mid \vec{x}) \ln \left[\frac{p(\vec{x}, \vec{y}, \vec{z})}{q(\vec{z}, \vec{y} \mid \vec{x})} \right] d\vec{z} \\
\approx & \sum_{\vec{y}} \int_{\vec{z}} q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{x}) \ln \left[\frac{p(\vec{y})p(\vec{z} \mid \vec{y})p(\vec{x} \mid \vec{z})}{q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{x})} \right] d\vec{z} \\
= & \sum_{\vec{y}} \int_{\vec{z}} q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{x}) \ln \left[\frac{p(\vec{y}) \frac{p(\vec{y}|\vec{z})p(\vec{z})}{p(\vec{y})} p(\vec{x} \mid \vec{z})}{q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{x})} \right] d\vec{z} \\
= & \sum_{\vec{y}} \int_{\vec{z}} q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{x}) \ln \left[\frac{p(\vec{y} \mid \vec{z})p(\vec{z})p(\vec{x} \mid \vec{z})}{q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{x})} \right] d\vec{z} \\
= & \sum_{\vec{y}} \int_{\vec{z}} q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{x}) \left[\ln \frac{p(\vec{z})p(\vec{x} \mid \vec{z})}{q(\vec{z} \mid \vec{x})} + \ln \frac{p(\vec{y} \mid \vec{z})}{q(\vec{y} \mid \vec{x})} \right] d\vec{z} \\
= & \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln \frac{p(\vec{z})p(\vec{x} \mid \vec{z})}{q(\vec{z} \mid \vec{x})} \, d\vec{z} + \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \sum_{\vec{y}} q(\vec{y} \mid \vec{x}) \ln \frac{p(\vec{y} \mid \vec{z})}{q(\vec{y} \mid \vec{x})} \, d\vec{z} \\
= & \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln \frac{p(\vec{z})p(\vec{x} \mid \vec{z})}{q(\vec{z} \mid \vec{x})} \, d\vec{z} - \int_{\vec{z}} q(\vec{z} \mid \vec{x}) D_{KL}(q(\vec{y} \mid \vec{x}) \,||\, p(\vec{y} \mid \vec{z})) \, d\vec{z} \qquad (4.14)
\end{aligned}$$

The second term in Equation (4.14) is maximised when $q(\vec{y} \mid \vec{x}) = p(\vec{y} \mid \vec{z})$ by definition of the KL-divergence, and the first term does not depend on $\vec{y}$. Hence, in the optimum, it holds that $q(\vec{y} \mid \vec{x}) = p(\vec{y} \mid \vec{z})$ (Jiang et al., 2016). We now take this result and show that $q(\vec{y} \mid \vec{x}) = p(\vec{y} \mid \vec{z})$ can be calculated analytically in our case, where we do not have a multivariate mixture model but an array of univariate mixture models. By doing so we assume that the y_i are independent:

$$\begin{aligned}
q(\vec{y} \mid \vec{x}) = p(\vec{y} \mid \vec{z}) &= \frac{p(\vec{y})p(\vec{z} \mid \vec{y})}{p(\vec{z})} \\
&= \frac{p(y_1, \ldots, y_E) \prod_{i=0}^{E-1} p(z_i \mid y_i)}{p(z_1, \ldots, z_E)} \\
&= \frac{\prod_i p(y_i)p(z_i \mid y_i)}{\prod_i p(z_i)} \\
&= \frac{\prod_i p(y_i)p(z_i \mid y_i)}{\prod_i \sum_{j=0}^{K} p(y_{ij}, z_i)} \\
&= \prod_i \frac{p(y_i)p(z_i \mid y_i)}{\sum_{j=0}^{K} p(y_{ij})p(z_i \mid y_{ij})} \qquad (4.15) \\
&= \prod_i p(y_i \mid z_i) \qquad (4.16)
\end{aligned}$$

The denominator has a computational complexity of $\mathcal{O}(E \times K)$, which is tractable. Note that if we do not assume that y_i are independent, we get an intractable summation over K^E terms:

$$q(\vec{y} \mid \vec{x}) = \frac{p(\vec{y})p(\vec{z} \mid \vec{y})}{\sum_{y_0} \cdots \sum_{y_E} p(y_0, \ldots, y_E)p(\vec{z} \mid y_0, \ldots, y_E)} \qquad (4.17)$$

Given an input $\vec{x}$, $\vec{z}$ can be inferred using $q(\vec{z} \mid \vec{x})$. The categorical assignments $\vec{y}$ are given by $p(\vec{y} \mid \vec{z})$: the i-th component of z is assigned category y_{ik} if $\frac{p(y_{ik})p(z_i|y_{ik})}{\sum_{j=0}^{K} p(y_{ij})p(z_i|y_{ij}))}$ is maximal for $0 \leq k < K$. Using the prior distribution $p(\vec{z} \mid \vec{y})$ (Equation (4.9)) it is possible to visualise a different category $\vec{y}$. In this approach there is no way of inferring the true posterior $p(\vec{z} \mid \vec{y}, \vec{x})$.

Some components of the ELBO can be integrated analytically, others can be approximated by one Monte-Carlo sample:

$$\ln p(X) \geq \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] - \mathbb{E}_{q(\vec{y}|\vec{x})}[D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z} \mid \vec{y}))] - D_{KL}(q(\vec{y} \mid \vec{x} \,||\, p(\vec{y}))$$

$$\begin{aligned}
\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] &= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln p(\vec{x} \mid \vec{z}) \, d\vec{z} \\
&\approx \ln p(\vec{x} \mid \vec{z^{(s)}}) \qquad \text{where } \vec{z^{(s)}} \sim q(\vec{z} \mid \vec{x}) \\
\mathbb{E}_{q(\vec{y}|\vec{x})}[D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z} \mid \vec{y}))] &= \sum_{\vec{y}} q(\vec{y} \mid \vec{x}) \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln q(\vec{z} \mid \vec{x}) \, d\vec{z} \\
&\quad - \sum_{\vec{y}} q(\vec{y} \mid \vec{x}) \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln p(\vec{z} \mid \vec{y}) \, d\vec{z} \\
&= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln q(\vec{z} \mid \vec{x}) \, d\vec{z} - \mathbb{E}_{q(\vec{y}|\vec{x})}[\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{z} \mid \vec{y})]] \\
&= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln q(\vec{z} \mid \vec{x}) \, d\vec{z} - \mathbb{E}_{p(\vec{y}|\vec{z})}[\mathbb{E}_{q(\vec{z}|\vec{x})}[\sum_{i=0}^{E-1} \ln p(z_i \mid y_i)]] \\
&= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln q(\vec{z} \mid \vec{x}) \, d\vec{z} - \sum_{i=0}^{E-1} \mathbb{E}_{p(\vec{y}|\vec{z})}[\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)]] \\
&= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln q(\vec{z} \mid \vec{x}) \, d\vec{z} - \sum_{i=0}^{E-1} \mathbb{E}_{p(y_i|z_i)}[\mathbb{E}_{q(z_i|\vec{x})}[\ln p(z_i \mid y_i)]]
\end{aligned}$$

$$= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln q(\vec{z} \mid \vec{x})\, d\vec{z} - \sum_{i=0}^{E-1} \sum_{j=0}^{K} p(y_{ij} \mid z_i) \int_{z_i} q(z_i \mid \vec{x}) \ln p(z_i \mid y_{ij})\, dz_i$$

$$\begin{aligned} D_{KL}(q(\vec{y} \mid \vec{x} \,||\, p(\vec{y}))) &= \sum_{\vec{y}} q(\vec{y} \mid \vec{x}) \ln q(\vec{y} \mid \vec{x}) - \sum_{\vec{y}} q(\vec{y} \mid \vec{x}) \ln p(\vec{y}) \\ &= \sum_{\vec{y}} p(\vec{y} \mid \vec{z}) \ln p(\vec{y} \mid \vec{z}) - \sum_{\vec{y}} p(\vec{y} \mid \vec{z}) \ln p(\vec{y}) \\ &= \sum_{i=0}^{E-1} \sum_{j=0}^{K} p(y_{ij} \mid z_i) \ln p(y_{ij} \mid z_i) - \sum_{i=0}^{E-1} \sum_{j=0}^{K} p(y_{ij} \mid z_i) \ln p(y_{ij}) \end{aligned}$$

The equality $\sum_{i=0}^{E-1} \mathbb{E}_{p(\vec{y}|\vec{z})}[\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)]] = \sum_{i=0}^{E-1} \mathbb{E}_{p(y_i|z_i)}[\mathbb{E}_{q(z_i|\vec{x})}[\ln p(z_i \mid y_i)]]$ follows from the assumed independences:

$$\begin{aligned} p(\vec{y} \mid \vec{z}) &= \prod_{i}^{E-1} p(y_i \mid z_i) \\ q(\vec{z} \mid \vec{x}) &= \prod_{i}^{E-1} q(z_i \mid \vec{x}) \qquad \text{(due to the diagonal covariance matrix in } q(\vec{z} \mid \vec{x})) \\ p(\vec{z} \mid \vec{y}) &= \prod_{i}^{E-1} p(z_i \mid y_i) \end{aligned}$$

The proof is as follows:

$$\begin{aligned} \sum_{i=0}^{E-1} \mathbb{E}_{p(\vec{y}|\vec{z})}[\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)]] &= \sum_{i} \sum_{y_1} \cdots \sum_{y_E} p(y_1, \ldots, y_E \mid z_1, \ldots, z_E) \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)] \\ &= \sum_{i} \sum_{y_1} \cdots \sum_{y_E} \prod_{j} p(y_j \mid z_j) \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)] \\ &= \sum_{i} \sum_{y_1} p(y_1 \mid z_1) \sum_{y_2} p(y_2 \mid z_2) \cdots \sum_{y_E} p(y_E \mid z_E) \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)] \\ &= \sum_{i} \sum_{y_i} p(y_i \mid z_i) \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)] \underbrace{\prod_{j \neq i} \sum_{y_j} p(y_j \mid z_j)}_{1} \\ &= \sum_{i} \sum_{y_i} p(y_i \mid z_i) \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)] \\ &= \sum_{i} \mathbb{E}_{p(y_i|z_i)}[\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)]] \end{aligned}$$

In a similar way it can be shown that $\sum_i \mathbb{E}_{p(y_i|z_i)}[\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(z_i \mid y_i)]] = \sum_i \mathbb{E}_{p(y_i|z_i)}[\mathbb{E}_{q(z_i|\vec{x})}[\ln p(z_i \mid y_i)]]$.

Full posterior inference

Alternatively, the true factorisation of the approximate posterior over latents can be considered instead of the mean-field approximation:

$$q(\vec{z}, \vec{y} \mid \vec{x}) = q(\vec{y} \mid \vec{x}) q(\vec{z} \mid \vec{x}, \vec{y}) \tag{4.18}$$

Both variational approximations q can be parametrised by neural networks. The advantage of this approach over the previous, analytical result is that the network modelling $q(\vec{z} \mid \vec{x}, \vec{y})$ allows for posterior inference over the Gaussian mixture components. A low variance is typical of descriptive

latents (Higgins et al., 2017c), and as such the mixtures with lowest inferred posterior variance could be interpreted as those matching visual concepts.

We assume the following (prior) distributions:

$$p(\vec{y}) = \text{Cat}(\Pi) \tag{4.19}$$
$$p(\vec{z} \mid \vec{y}) = \mathcal{N}((Y^T M)\vec{1}, (Y^T \Sigma)\vec{1}) \tag{4.20}$$
$$p(\vec{x} \mid \vec{z}) = \mathcal{N}(\vec{\mu}, \text{diag}(\vec{\sigma}^2)) \tag{4.21}$$

as well as the following approximate posteriors:

$$q(\vec{y} \mid \vec{x}) = \text{ST Gumbel-Softmax}(\Phi) \tag{4.22}$$
$$q(\vec{z} \mid \vec{y}, \vec{x}) = \mathcal{N}((Y^T M_q)\vec{1}, (Y^T \Sigma_q)\vec{1}) \tag{4.23}$$

where Π are the prior class probabilities, and Φ the posterior class probabilities. The Straight-Through (ST) Gumbel-Softmax is a particular application of the Gumbel-Softmax distribution where the argmax is taken over the continuous approximation to obtain a discrete value, while the gradients are still taken with respect to the underlying continuous approximation (Jang et al., 2016). This is different from Section 4.1, where the variable modelled by a Gumbel-Softmax distribution remained a continuous approximation at all times. Here, a one-hot discrete value is required to select the mixture components.

The ELBO changes with a different factorisation of the posterior over latent variables:

$$\begin{aligned}
p(X) &\geq \int_{\vec{y}} \int_{\vec{z}} q(\vec{z}, \vec{y} \mid \vec{x}) \ln \left[\frac{p(\vec{x}, \vec{y} \mid \vec{x})}{q(\vec{z}, \vec{y} \mid \vec{x})} \right] d\vec{z}\, d\vec{y} \\
&= \int_{\vec{y}} \int_{\vec{z}} q(\vec{y} \mid \vec{x}) q(\vec{z} \mid \vec{y}, \vec{x}) \ln \left[\frac{p(\vec{y}) p(\vec{z} \mid \vec{y}) p(\vec{x} \mid \vec{z})}{q(\vec{y} \mid \vec{x}) q(\vec{z} \mid \vec{y}, \vec{x})} \right] d\vec{z}\, d\vec{y} \\
&= \mathbb{E}_{q(\vec{z}, \vec{y} \mid \vec{x})}[\ln p(\vec{x} \mid \vec{z})] - D_{KL}(q(\vec{y} \mid \vec{x}) \,||\, p(\vec{y})) - \mathbb{E}_{q(\vec{y} \mid \vec{x})}[D_{KL}(q(\vec{z} \mid \vec{y}, \vec{x}) \,||\, p(\vec{z} \mid \vec{y}))]
\end{aligned}$$

Individual terms can be approximated by one Monte Carlo sample:

$$\begin{aligned}
\mathbb{E}_{q(\vec{z}, \vec{y} \mid \vec{x})}[\ln p(\vec{x} \mid \vec{z})] &= \int_{\vec{y}} \int_{\vec{z}} q(\vec{z} \mid \vec{x}, \vec{y}) q(\vec{y} \mid \vec{x}) \ln p(\vec{x} \mid \vec{z})\, d\vec{y}\, d\vec{z} \\
&\approx \int_{z} q(\vec{z} \mid \vec{x}, \vec{y^{(s)}}) \ln p(\vec{x} \mid \vec{z})\, d\vec{z} \qquad \text{where } \vec{y^{(s)}} \sim q(\vec{y} \mid \vec{x}) \\
&\approx \ln p(\vec{x} \mid \vec{z^{(s)}}) \qquad \text{where } \vec{z^{(s)}} \sim q(\vec{z} \mid \vec{x}, \vec{y^{(s)}})
\end{aligned} \tag{4.24}$$

$$D_{KL}(q(\vec{y} \mid \vec{x} \,||\, p(\vec{y}))) = \sum_{i=0}^{E-1} \sum_{j=0}^{K-1} q(y_{ij} \mid \vec{x}) \ln \frac{q(y_{ij} \mid \vec{x})}{p(y_{ij})} \tag{4.25}$$

$$\begin{aligned}
\mathbb{E}_{q(\vec{y} \mid \vec{x})}[D_{KL}(q(\vec{z} \mid \vec{y}, \vec{x}) \,||\, p(\vec{z} \mid \vec{y}))] &= \int_{\vec{y}} q(\vec{y} \mid \vec{x}) \int_{\vec{z}} q(\vec{z} \mid \vec{y}, \vec{x}) \ln \left[\frac{q(\vec{z} \mid \vec{y}, \vec{x})}{p(\vec{z} \mid \vec{y})} \right] d\vec{z}\, d\vec{y} \\
&\approx \int_{\vec{z}} q(\vec{z} \mid \vec{y^{(s)}}, \vec{x}) \ln \left[\frac{q(\vec{z} \mid \vec{y^{(s)}}, \vec{x})}{p(\vec{z} \mid \vec{y^{(s)}})} \right] d\vec{z} \qquad \text{where } y^{(s)} \sim q(\vec{y} \mid \vec{x}) \\
&= \int_{\vec{z}} q(\vec{z} \mid \vec{y^{(s)}}, \vec{x}) \ln q(\vec{z} \mid \vec{y^{(s)}}, \vec{x})\, d\vec{z} \\
&\quad - \int_{\vec{z}} q(\vec{z} \mid \vec{y^{(s)}}, \vec{x}) \ln p(\vec{z} \mid \vec{y^{(s)}})\, d\vec{z}
\end{aligned} \tag{4.26}$$

The integrals in Equation (4.26) can be calculated analytically, as both distributions under the integral are multivariate Gaussians in the same variable $\vec{z}$ (see Section A.3).

Chapter 5

Evaluation of statistical models

In this chapter we evaluate the two statistical models introduced in the previous chapter with different priors on a dataset of celebrity faces (Liu et al., 2015) and computer-generated chairs (Aubry et al., 2014). The faces dataset is made up of 202,599 rectangular RGB images, the chairs dataset consists of 86,366 square images. For the chairs dataset, the images are converted to greyscale. For the faces dataset, the images are cropped to a square. For both datasets, images are downsized to 64x64 pixels, and pixel values are scaled down from the original discrete range $[0, 255]$ to a continuous value between $[0, 1]$. Samples from the chairs dataset are interpreted as Bernoulli probabilities, while *each color channel* $\vec{x}_c$ in an image from the faces dataset can be modelled as a sample from a multivariate Gaussian distribution (Higgins et al., 2017a):

$$p_{chairs}(\vec{x} \mid \vec{z}) = \text{Bernoulli}(\vec{\pi})$$
$$p_{faces}(\vec{x}_c \mid \vec{z}) = \mathcal{N}(\vec{\mu}_c, \text{diag}(\vec{\sigma}_c^2))$$

With the original purpose in mind, the final model that will be selected for a learning task has to perform well with regard to the following quantitative and qualitative criteria:

1. The number of relevant visual factors discovered. *(quantitative)*

2. Disentanglement performance. *(qualitative)*

3. Classification accuracy of concepts within visual factors, compared to the $d\beta$-VAE baseline in the 2 category case. *(quantitative)*

4. The aptitude for integration in an unsupervised, automated decision process. For example, a model that requires a manual procedure to identify descriptive latents does not scale well with the size of the latent channel. *(qualitative)*

The first three criteria are measured against the baseline, β-VAE. The models are implemented in TensorFlow (Abadi et al., 2015). Training is performed using the Adam optimiser (Kingma and Ba, 2014) with a learning rate of 10^{-4} over 500,000 randomly drawn minibatches of size 16[1]. Evaluation is on the faces dataset first, as it is more visually expressive and rich, allowing for an easier visual interpretation of the results. Network architectures (see Chapter C) as well as the number of latents $E = 32$ are chosen as in Higgins et al. (2017a) for a fair comparison of the newly introduced underlying statistical models. The number of categories considered is initially $K = 5$ as it allows to see how well the variation within a factor is spread over the different categories. A derivation of some expectations in the ELBO of the models from the previous chapter for the faces and chairs dataset is given in Appendix A.

[1]To accurately reproduce β-VAE results, hyperparameters are chosen identical as in Higgins et al. (2017a). The number of minibatches and the minibatch size were conveyed through personal communication with Irina Higgins, the learning rate and optimiser are listed in the β-VAE paper.

5.1 Discrete Model

In Section 4.1 two possible priors on the discrete random variable $\vec{z}$ were identified: a uniform prior and the Gaussian-CDF-like prior. In addition to these two priors, we consider the impact of two different values for the β-constraint in the ELBO: $\beta = 1$ (equivalent to the original VAE formulation), and $\beta = \beta_{high}$, where β_{high} depends on the dataset the model is evaluated on. β_{high} is chosen to be the value found in Higgins et al. (2017a) to improve disentanglement performance on the same datasets. For the faces dataset, $\beta_{high} = 250$, for the chairs dataset, $\beta_{high} = 5$. In summary, the following setups are under consideration within the model that assumes a discrete distribution over $\vec{z}$:

- A uniform prior assigning equal probability to each class, and a $\beta = 1$ constraint applied to the KL-divergence term between approximate posterior $q(\vec{z} \mid \vec{x})$ and prior $p(\vec{z})$ in the ELBO.
- A uniform prior, $\beta = \beta_{high}$.
- A Gaussian-CDF-like (GCDF) prior, $\beta = 1$.
- A GCDF prior, $\beta = \beta_{high}$.

Faces

In Figure 5.2, the visual information encoded by two latents and their categories is displayed for three of the setups. Before analysing the results, we briefly explain how the figure was created. Each "row" in the table displays the reconstruction of 5 concept classes within a single visual factor. The columns indicate from which model the reconstruction was obtained. For example, in the top row, the second column, the reconstructions of factor *fringe* are displayed as obtained by $d\beta$-VAE. For each technique and each input image, 5 reconstructions are listed. Each reconstruction is a representation of a different concept class within a visual factor. If we take the very first input face for factor *fringe*, we see 20 reconstructions of this face: 5 from each technique. We expect each concept class to result in slightly different fringe reconstruction: e.g. across the 5 classes, fringe gradually increases in length. For $d\beta$-VAE the five iterations are over the univariate unit Gaussian as explained in Example 3.3.1. The encoder network provides a parametrisation μ of $q(\vec{z} \mid \vec{x})$. Let $\tilde{\mu}$ be μ where the inferred mean μ_i for latent i encoding fringe has been replaced by one of five equally spaced values over the interval $[-3, 3]$. Then a *reconstruction* image is given by the parametrisations μ_c for all color channels c of $p(\vec{x_c} \mid \vec{z})$, where $\vec{z} = \tilde{\mu}$. For the other three discrete models, reconstructions are obtained similarly, but the inputs to the decoder network are different. From the encoder network for an input $\vec{x}$, the class probabilities $\Phi \in \mathbb{R}^{32\times5}$ of the Gumbel-Softmax posterior over $\vec{z}$ is obtained. For the row $\vec{\phi_i}$ in Φ where i is the index of the latent encoding fringe, alternatingly a different class (column) probability is set to 1, while all other columns have value 0. The 5 resulting Φ are provided to the decoder network to obtain 5 different reconstructions. The class with largest posterior probability in row $\vec{\phi_i}$ of Φ is indicated with a red rectangle. A good method is characterised by the highlighted reconstruction most closely matching the input image amongst all 5 concept classes.

The fourth setup, a uniform prior with high β constraint, has been omitted since it did not result in the encoding of any generative information at all: across all latents and categories, the same reconstruction was obtained (see Figure 5.1). It seems to be an average of all faces across the entire dataset. This result illustrates the impact of the β constraint on the ELBO. It is an obvious consequence of enforcing a close match between posterior distribution $q(\vec{z} \mid \vec{x})$ and a noninformative uniform prior $p(\vec{z})$. Since no generative information is encoded at all, this model is discarded for the remaining experiments.

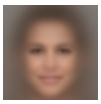


Figure 5.1: Reconstructed face for a uniform prior with $\beta = 250$.

For the three remaining setups, the informative capacity remains limited. Many factors discovered by β-VAE are not discovered at all, while in addition no new factors are found either; an overview is given in Table 5.1. The limited number of factors discovered suggests that the latent vector $\vec{z}$ forms a capacity bottleneck. To confirm this hypothesis, the performance of the three setups is cross-validated on the chairs dataset.

Factor	β-VAE	GCDF, $\beta = 1$	GCDF, $\beta = \beta_{high}$	Uniform, $\beta = 1$
Smile/sunglasses	✓	✓	-	✓✓
Fringe	✓	✓	✓	✓
Skin color	✓	-	✓	-
Azimuth	✓	-	✓	-
Hair parting	✓	-	-	-
Hue/saturation	✓	-	-	-
Background	✓	-	-	-
Age/gender	✓	-	-	-

Table 5.1: Comparison of visual factors discovered by each variation on the model that assumes a discrete distribution over $\vec{z}$. A double checkmark indicates that both factors are discovered separately.

Chairs

On the chairs dataset, latent factor discovery remains problematic. Out of the four factors listed in the β-VAE paper, two out of three setups fail to discover variations in width/size in the input images (Table 5.2). Inspecting the reconstructions across categories given by informative latents, an additional issue arises. In Figure 5.3, it can be seen that information is lost and instead, duplicative information is encoded. For example, in the case of azimuth, the three models consistently fail to encode more than two different angles. Similarly, each setup allocates just one category for a tall back height, while the other four categories give a reconstruction with identical back heights. Such information loss is intolerable for the envisioned application, especially if visualisation becomes an integral part of the learning framework. In conclusion, the discrete model fails to discover as much factors as the continuous β-VAE, and for factors that are discovered, not all possible variations of the factor in the input are captured. It is assumed that the limited expressivity of the discrete model can be attributed to its discreteness. An unrestricted range of continuous values is by definition capable of expressing more variation than a restricted discrete range (the categories per latent).

Factor	β-VAE	GCDF, $\beta = 1$	GCDF, $\beta = \beta_{high}$	Uniform, $\beta = 1$
Back height	✓	✓	✓	✓
Width/size	✓	-	-	✓
Leg style	✓	✓	✓	✓
Azimuth	✓	✓	✓	✓

Table 5.2: Comparison of visual factors discovered by each alternative formulation of the model that assumes a discrete $\vec{z}$.

Figure 5.2: The 2-variable discrete model applied to the Faces dataset with three different priors. See text for explanation of the figure.

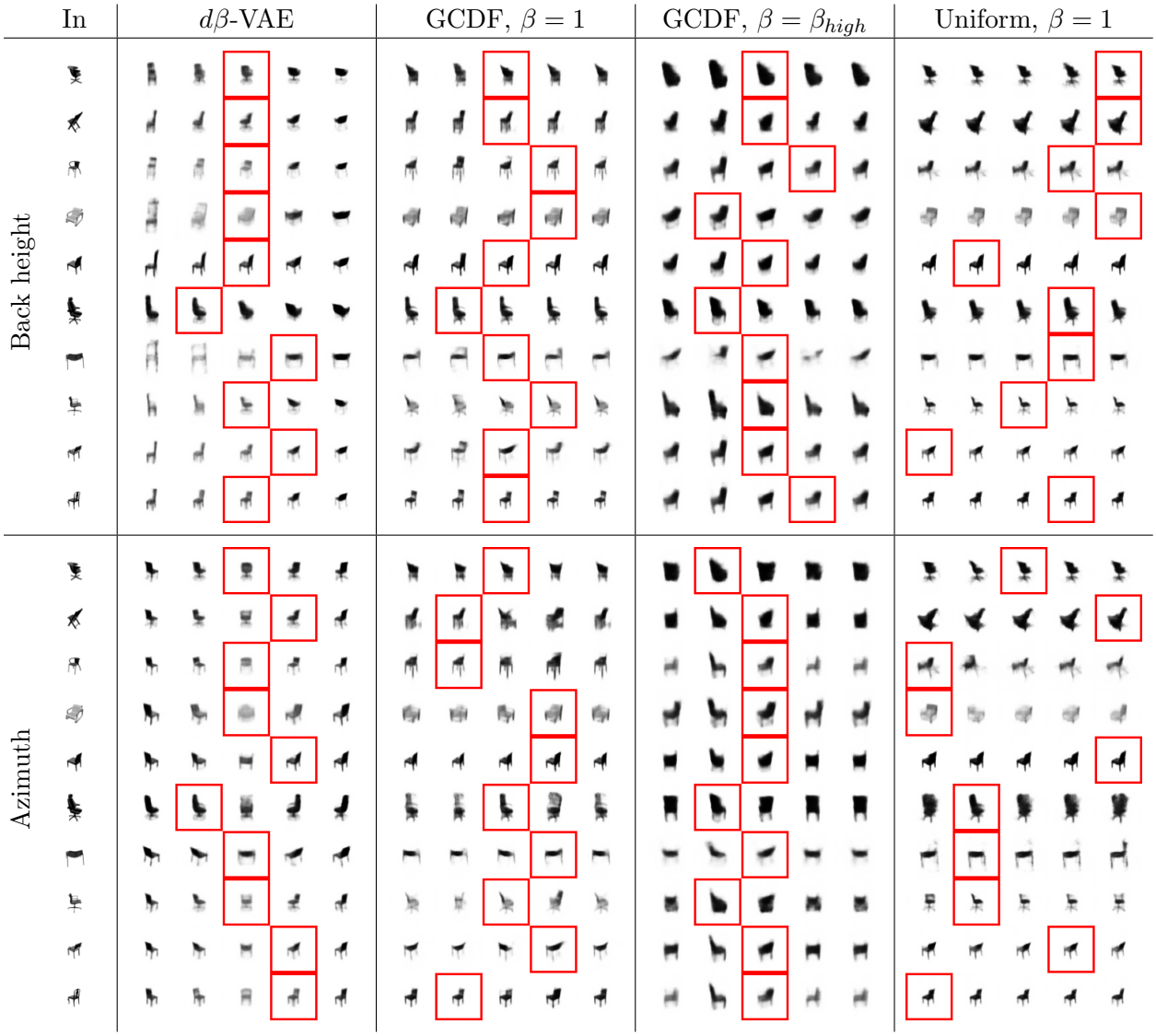


Figure 5.3: The 2-variable discrete model applied to the Chairs dataset with three different priors. The figure can be explained similarly as Figure 5.2, but the reconstructions are now given by the parametrisation $\vec{\pi}$ of the Bernoulli distribution $p(\vec{x} \mid \vec{z})$.

5.2 Univariate Gaussian Mixtures

Within the mixture model, we identified two different derivations of an evidence lower bound. The first is based on a mean-field approximation of the approximate joint posterior over latents, the second takes the full factorisation. We claimed that the derivation with mean-field approximation is less suited for our setting, since it does not incorporate posterior inference over the Gaussian mixtures. In this section, we evaluate the following variations on those models:

1) **Mean-field approximation** We assume the mean-field factorisation $q(\vec{z}, \vec{y} \mid \vec{x}) \approx q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{x})$ of the approximate joint posterior over latent variables. $q(\vec{y} \mid \vec{x})$ is not given by a variational approximation, but calculated analytically. The prior class probabilities Π are initialised and fixed at $\frac{1}{K}$, where $K = 5$ is the number of categories. The prior $p(\vec{z} \mid \vec{y})$ is parametrised by a matrix $M \in \mathbb{R}^{E \times K}$ holding the $E \times K$ univariate mixture means, and a matrix $\Sigma \in \mathbb{R}^{E \times K}$ holding the variances (Equation (4.9)). The columns of M are given by equal-spaced values over the interval [-3, 3]. Σ is initialised and fixed at 0.6^2, an arbitrary value since the encoder and decoder networks can scale the inputs with the prior variance.

2) **Mean-field approximation** All hyperparameters are as in 1, except for M and Σ. We choose an expected number of descriptive latents $D = 10$. Then the lower $E - D$ rows of M and Σ remain as in 1, while the first D rows of M are set to 0, and the first D rows of Σ to a larger variance of 1. It is expected that the first D components of $\vec{z}$ learn to encode descriptive latents. This model improves on 1 by allocating a fixed number of expected, descriptive latents; those latents qualify for translation to logical atoms. 1 does not expose a posterior over the mixtures, and as such identifying descriptive latents in 1 requires visual inspection, which does not scale well with the number of latents E.

3) **Full factorisation** The full factorisation of the joint approximate posterior over latent variables is assumed: $q(\vec{z}, \vec{y} \mid \vec{x}) = q(\vec{y} \mid \vec{x})q(\vec{z} \mid \vec{y}, \vec{x})$. This model improves on 2 by providing the posterior means and variances of the mixtures. A mixture with high posterior variances characterises a non-descriptive latent (Higgins et al., 2017c). All hyperparameters otherwise remain as in 1.

The mixture model was formulated in such a way that the latent vector $\vec{z}$ is unchanged from the original β-VAE formulation. It remains multivariate Gaussian distributed, but variations within its components are now grouped together in small univariate Gaussian concentrations. The reasoning behind the β constraints remains intact for disentanglement purposes, and we apply $\beta = \beta_{high}$ in all three setups listed above.

Faces

Figure 5.4 provides a comparison between iterations over five descriptive latents given by all three setups. More iterations are available in Appendix B. The $d\beta$-VAE reconstructions are obtained in the same way as in the previous section. We again define $\tilde{\mu}$ as the mean of the posterior distribution over $\vec{z}$ where one component μ_i for a latent factor i has been altered. For models 1, 2, and 3, μ_i is replaced with one of the five prior means from row i in matrix M. For model 3, $\vec{\mu}$ is not directly obtained from the encoder; instead, $\vec{\mu}$ is given by selecting those posterior mixture means from M according to the value of discrete vector $\vec{y}$. The category corresponding to the red rectangle in models 1 and 2 is given by the $\vec{y}$ for which the analytical computation of $q(\vec{y} \mid \vec{x})$ is maximal (Equation (4.16)), for the 3rd model it is directly given by $\vec{y}$ inferred using the $q(\vec{y} \mid \vec{x})$ neural net.

All three models manage to discover factors that are not isolated by β-VAE (see Table 5.3). Particular variations in those factors may be exhibited in some latents by β-VAE, but they are entangled with other factors. For example, in the iterations over skin tone, β-VAE reconstructions show a change in hair color alongside the skin tone change. These variations in hair color were isolated into a distinct latent by all three mixture models. Amongst those models, hair color was captured in the most disentangled manner by model 1. In the reconstructions, it is the only

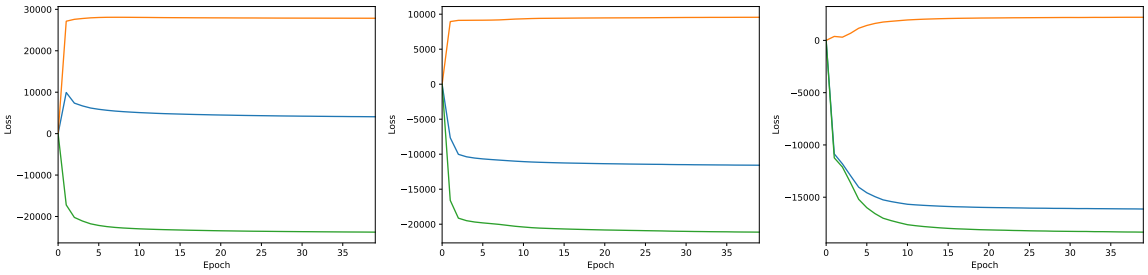
Figure 5.4: Overview of latent factors found by three different variations on the mixture model VAE, compared with the baseline β-VAE. See text for an explanation of the figure.

visual aspect changing, while for model 2 and 3 hair color seems to be entangled with gender. Overall, model 1 has the best performance in terms of reconstruction fidelity, disentanglement, and discovery of relevant visual factors. Unfortunately within model 1 there is no way of automatically identifying descriptive latents. The same is true for model 2: the number of expected descriptive latents ($D = 10$) does not match the actual number of descriptive latents identified through visual inspection (see Table 5.3). Model 3 is an acceptable alternative with a performance on par with β-VAE in latent factor discovery and disentanglement performance. Based on three of the four criteria outlined in the introduction of this chapter (latent factor discovery, disentanglement performance, and aptitude for integration in an automated decision process), model 3 is the best candidate out

of all models considered. It remains to be confirmed if model 3 also outperforms β-VAE in the classification of visual concepts, a critical requirement for success in a learning task based on the inferred visual concepts.

Factor	$d\beta$-VAE	1	2	3
Smile/sunglasses	✓	✓✓	✓	✓
Fringe	✓	✓	✓	✓
Skin color	✓	✓	✓	✓
Hair color	-	✓	✓	✓
Hair length	-	✓	✓	✓
Beard	-	✓	-	-
Tilt	-	✓	-	-
Azimuth	✓	-	-	✓
Hair parting	✓	✓	✓	✓
Hue/saturation	✓	-	-	-
Background	✓	-	-	✓
Age/gender	✓	-	-	-

Table 5.3: Comparison of factors discovered by the original β-VAE formulation (Higgins et al., 2017a) compared to three variations on the mixture model formulation. A double checkmark indicates that the factors are encoded in two distinct latents, as opposed to being entangled in one.



(a) 1) (b) 2) (c) 3)

Figure 5.5: Training loss at each epoch for all three models, evaluated on the faces dataset. Total loss is indicated in blue, the contribution from the KL-divergence term in orange, the reconstruction term in green (see Equation (2.29)). The aim is to minimise the loss, which is the negated lower bound of the marginal log likelihood. Based on the loss graphs, the 3rd alternative is the model that explains the data best. It could reasonably be expected that the model where the joint approximate posterior over latents is not approximated by a mean-field assumption is capable of reaching a better optimum of the ELBO.

Chairs

Model 3, found to be most suitable for our purpose after evaluation on the faces dataset, is cross-validated on the chairs dataset. The impact of a different number of categories on the qualitative results is now considered: $K \in \{5, 3, 2\}$.

Figure 5.6 lists iterations over the latents width/size, back height, leg style and azimuth. Although the model seems to learn two more low variance mixtures which appear to be encoding leg height and back style, we choose to focus on those factors that are listed in Higgins et al. (2017a) in order to compare classification accuracy, the fourth criterion from the introduction of this chapter, with β-VAE.

In the case of $K = 5$, the difference between a discretisation of the β-VAE Gaussian and the class assignments within the mixture model becomes apparent. In the β-VAE column, row 3 (leg style), all values are clustered around 0, and the chair with wheels (5th chair) is assigned the same class as

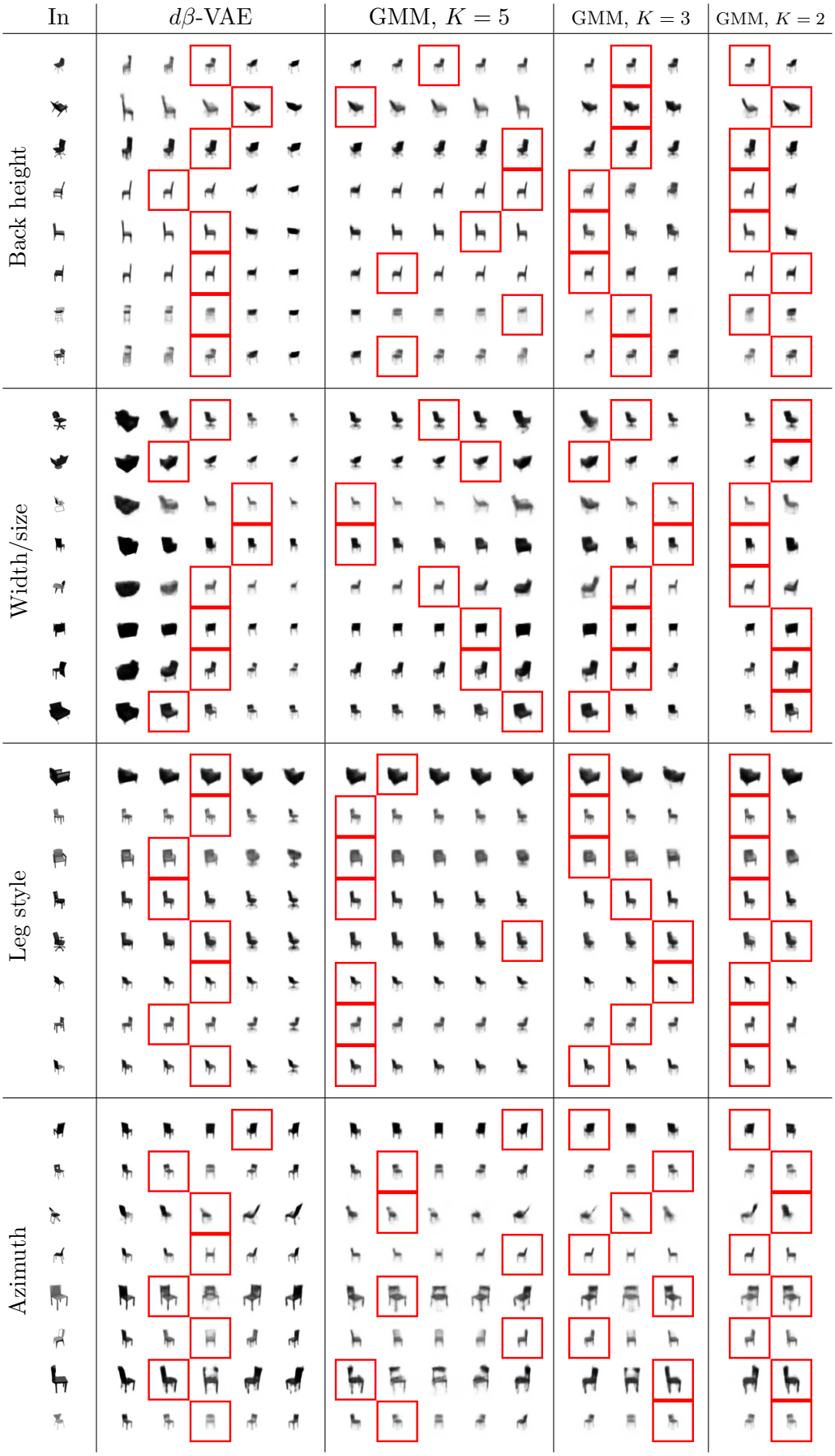


Figure 5.6: Overview of latent factors found by three different variations on the mixture model VAE, compared with the baseline β-VAE. See text for an explanation of the figure. β-VAE assigns most chairs the middle category, which shows the limitation of the multivariate Gaussian prior.

some chairs with legs. In the mixture model, (GMM $K = 5$ column), there is a clear dividing line between those chairs with legs on the left, and the ones with wheels on the right. As the number of categories is reduced, this distinction remains in place. Additionally, for latent width/size, again most chairs are assigned the middle category, by definition of the Gaussian distribution. In the case of $K = 5$, size classes are more spread out, allowing for inference of more powerful and precise rules over the input.

Figure 5.6 clearly shows an advantage of the mixture model over β-VAE in case of a relatively large number of categories. It remains to be investigated how the mixture model compares to the classification accuracy of β-VAE in the 2-category case ($K = 2$), which is unclear from Figure 5.6 and Figure B.4 in the Appendix.

Classification accuracy

A limitation of any of the new models introduced in this work, is that the number of categories has to be devised beforehand, and is fixed to the same number across all latents (see Section 7.1 for a proposed solution). As such, the number of categories we will settle on for a learning task, is rooted in the linguistics literature. In *structuralism*, understanding is contingent upon an ordered system of relationships (O'Sullivan et al., 1994). Individual concepts do not convey meaning, but the relationships within do. In binary opposition (De Saussure, 1959), a theme within structuralism, concepts are defined in terms of two opposites: good/bad, long/short, big/small. Following binary opposition, we assume each visual factor to take one of two opposite values. For example, for width/size the two possible classes are small and large. To compare categorical assignments between a naive discretisation of the continuous β-VAE ($d\beta$-VAE) and the mixture model, a random set Y of 200 chairs is drawn from the dataset and labelled. The 200 chairs are inspected **three times** in succession; the label that was assigned most frequently amongst the three instances is chosen as its ground-truth label. By repeating the inspection three times, the risk of human error and subjectivity, in particular for latents describing a relative ordering such as width/size, is reduced. A larger set of chairs would allow to draw stronger conclusions from the results, but due to time constraints and the labor-intensity of the manual classification, we have settled on the number 200. The ground-truth labels are subsequently compared with the labels assigned by $d\beta$-VAE and the mixture model to determine how well either discretisation technique performs. Both models assume two categories ($K = 2$). Through visual inspection (see Figure 5.6) it is determined which of the two extremas each category encodes (e.g. small or large in case of width/size). The classification accuracy of each model for each factor is displayed in Table 5.4.

	Attribute	Possible labels	Accuracy($K = 2$)		
			$d\beta$-VAE	GMM	
1	Back height	Short/tall	0.68	0.62	
2	Width/size	Small/large	0.74	0.78	
3	Leg style	Legs/other	0.8	0.74	
4	Azimuth	Facing left/facing right	0.98	0.95	
			0.80	0.77	(avg.)

Table 5.4: Broad distinction of classes within visual factors, and the classification accuracy of both models on a set of 200 labelled chairs.

On average, both approaches achieve approximately equal classification accuracy, although the accuracy per factor may differ. It is promising that the mixture model achieves a classification rate on par with $d\beta$-VAE in the 2 category case, given that a Gaussian curve is inherently symmetric - one would expect it to achieve top performance if it is split up in two parts at the center of symmetry (the mean). A 100% classification accuracy is not expected; the average classification accuracies are comparable to the values found in Jiang et al. (2016) and Dilokthanakul et al. (2016)

in a similar setting on other datasets, where instead of component-wise clustering, the entire latent vector $\vec{z}$ is assigned a class using a multivariate Gaussian mixture model.

5.3 Summary

It was shown that a discrete formulation of the simple VAE model with one latent random variable is incapable of achieving similar performance as the continuous counterpart in terms of the number of visual factors discovered, and the information encoded for each factor. On the visually rich faces dataset, many visual factors were not discovered by any of the alternative formulations of the discrete model. On the chairs dataset, just one setup managed to discover the same factors as the baseline, but failed to capture all variation within the factor present in the dataset. For example, in the case of azimuth, just 2 different angles were recorded, while the others are lost in the reconstructions. Such information loss is unacceptable.

The mixture model that introduces an additional latent random variable $\vec{y}$ forms a good alternative to $d\beta$-VAE. For larger values of K, information is better spread over the different classes, whereas for the Gaussian curve in $d\beta$-VAE all information is concentrated around the mean. For $K = 2$, classification performance of the mixture model is comparable with what is considered the baseline, $d\beta$-VAE. In a different formulation of the mixture model ELBO, the model discovers more relevant visual factors such as facial hair and tilt in the faces dataset than the baseline. This is a valuable contribution to the VAE literature. More experimentation with the same model on different datasets is left for future work, as we found the model is not suitable for our purpose due to a lack of posterior inference over the mixtures, which would enable automatic identification of descriptive latents.

Chapter 6

Learning chair preferences

The final step is to apply one of the discrete latent variable models to a learning task to see how well the training samples can be explained in terms of extracted visual concepts. The aim of the first learning task is to learn a *written* preference: one that is known beforehand, and according to which the training data will be ordered. Similar to the Monk's problems, the goal is then to uncover the preference(s) that created the ordering over the training examples. This is a simplistic task, and proofs the viability of the concept on a small scale (if it is known beforehand that the rule to be sought can be expressed in terms of the visual concepts that are extracted by the discrete latent variable model). The second learning task attempted is more advanced. Instead of learning a written preference, the aim is to find an explanation of comfortability in terms of low-level visual concepts. Throughout this chapter, again all factors are assumed to be composed of two concept classes ($K = 2$).

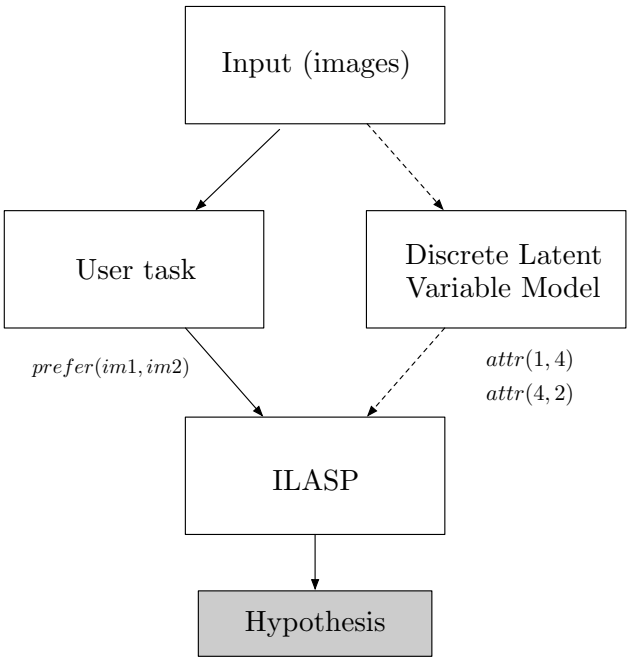


Figure 6.1: The final learning framework as outlined in Chapter 1.

Learning a written preference

For the first learning task, we apply the mixture model to extract visual concepts from the input images (the right path in Figure 6.1). A set of 16 pairs is randomly drawn from the Chairs dataset. Then, for each pair (X, Y):

1. The posterior class probabilities Φ are obtained for chair X and Y using the parametrisations of $q(\vec{y} \mid X)$ and $q(\vec{y} \mid Y)$. The categorical assignments $\vec{y}$ for the components of $\vec{z}$ are given by the classes with the largest posterior probabilities.

2. The categorical assignment of the latent encoding leg style for chair X is changed to the category encoding wheels, while for Y it is changed to legs.

3. The four latents back height, leg style, azimuth and width/size are encoded in an ILASP task with an ordering such that X is preferred over Y (*#order(X, Y)*).

The ordering of sets of extracted visual concepts forms the output of the left path in Figure 6.1, the concepts themselves are obtained through the right path. A concept is encoded as `attr(A, B)` where A is an integer identifying the visual factor, and B is the visual concept class within that factor. The atoms `attr(A, B)` are obtained from the mixture model; the discrete latent variable node in Figure 6.1 is implemented by the mixture model that was found to perform best in the previous chapter after $d\beta$-VAE. The full Learning to Rank Answer Sets (LTR) task encoding is given in Appendix D.

After 1.7 seconds[1], ILASP returns the following, correct weak constraint:

```
:~ not attr(3, 2).[1@1, 1]
```

Attribute 3 encodes leg style (see Table 5.4 for the mapping from integer to concept), while concept class 2 is formed by anything that is not four legs (see the last column in Figure 5.6). The hypothesis is correct, and the simple classification task has been performed successfully, yielding a human-readable hypothesis. A neural network would have been able to achieve the same 100% classification rate, but there is no straightforward way of inferring the mechanism according to which it classifies its input.

Learning an unwritten preference

Next, we aim to assess if any arbitrary visual learning task can be explained in terms of low-level visual concepts. This would expand the problem domain to which the learning framework is applicable from tasks where the solution is known beforehand to be defined in terms of visual factors. We return to the set Y of 200 labelled chairs from the previous chapter. The chairs are pairwise presented to a user, who is asked to select the chair they would prefer over the other in terms of *comfortability*. The pairwise orderings form the output of the left path in Figure 6.1, the concepts encoded in atoms present in each chair form the output of the right path. Three different ways of extracting the visual concepts from the input images are considered: the ground truth labels, $d\beta$-VAE, and the mixture model. It is not expected that the task for which the concepts were provided by the ground truth labels achieves 100% classification accuracy: one can imagine that comfortability is not just defined in terms of the four factors under consideration (azimuth, back height, leg style, width/size), but also by more detailed factors such as the presence or absence of armwrests and cushions. Extracting those factors are left for future work, but an alternative formulation of the mixture model with mean-field approximation has shown promising results on the faces dataset and managed to discover more detailed factors. The training set of 100 pairs is divided into 5 folds containing 20 pairs of chairs each; each time 4 different folds are selected for training, after which the found hypothesis is evaluated on the remaining validation fold. The final hypothesis is given by the solution that has the highest validation accuracy. The overall accuracy is averaged over all 5 validation folds. The results are displayed in Table 6.1.

It is not expected that azimuth plays a role in defining comfortability. Indeed, none of the found hypotheses include an `attr(4, X)` atom (azimuth). The ground truth accuracy is 88%, which means that on average 88% of the recorded preferences can be explained in terms of back height, size, and leg style. The second-best performing model is $d\beta$-VAE, which is expected since it achieved the highest accuracy in classifying concepts (Table 5.4). The mixture model follows with an accuracy of 79%.

In addition to the logic-based learning task, a multi-layer perceptron (MLP) was trained on the latent representation $\vec{z}$ of input chairs. It was decided to train on $\vec{z}$ as opposed to the full-size image inputs due to the small training set. If the inputs to the net were spatial (full size chair images),

[1]All learning tasks are performed on a Intel Core i7-6700 3.4 GHz Ubuntu machine.

	Classification	Accuracy	Learnt preferences
MLP	n/a	0.87	n/a

ILASP	Ground truth	0.88	`:~ not attr(1, 2), not attr(2, 2).[1@1, 2]`
			Back height is tall and size is large.
			`:~ not attr(3, 2).[1@2, 1]`
			Leg style is not legs.

ILASP	$d\beta$-VAE	0.86	`:~ not attr(2, 1).[1@2, 2]`
			Size is large.
			`:~ not attr(1, 1).[1@1, 1]`
			Back height is tall.

ILASP	GMM	0.79	`:~ not attr(2, 2).[1@1, 2]`
			Size is large.
			:
			`:~ not attr(3, 2).[1@2, 1]`
			Leg style is not legs.

Table 6.1: Results of the second learning task. See text for explanation. Some atoms may have a different class, but convey the same meaning English, due to the different models having learnt a reversed mapping from concepts classes to categories.

convolutional layers are required to robustly generalise over the pixel inputs, and those kernels have a large number of trainable parameters. The training set is too small to reliably obtain the optimal values of those parameters. The latent code $\vec{z}$ forms a good alternative since it encodes sufficient information to reconstruct the inputs with relatively high fidelity. The architecture of the MLP is listed in Table C.2 in Appendix C. The network is trained using the Adam optimiser (Kingma and Ba, 2014) with batch size 10 and learning rate 5×10^{-4}. The input is formed by the two $\vec{z}$ vectors of an input pair stacked on top of each other, the class labels are one-hot and indicate which of the two chairs in the input is preferred. The network's classification accuracy (0.87) is between the accuracy of a learning task based on ground-truth labels (0.88), and the accuracy of a learning task based on labels provided by $d\beta$-VAE (0.86). However, the MLP is obscure, and there is no straightforward method of finding out why it classifies the way it does.

In the task with ground truth labels, the highest priority constraint results in a penalty if the back height is not tall and the size is not large. At a lower priority level, a penalty is incurred if the leg style is four legs. $d\beta$-VAE finds a similar hypothesis, but omits the preference concerning leg style. The GMM finds just size and leg style. The differences in found hypotheses and classification accuracies can be attributed to the difference in classification accuracy of individual concepts (Table 5.4). Overall, however, the accuracy is relatively high compared to the ground truth, and the preferences convey similar rules.

Chapter 7

Conclusion

We have presented a proof-of-concept learning framework that can perform visual learning tasks based on extracted low-level visual concepts. Three extraction methods have been considered: a model with discrete latent variable, a model composed of a multitude of univariate mixture models, and a discrete adaption of the state-of-the-art model β-VAE. The model with a single discrete latent variable was found to be incapable of capturing as much visual factors as the baseline $d\beta$-VAE, and additionally incurred information loss in the reconstruction process. The mixture model approximates the state-of-the-art on all fronts, and additionally, it is inherently discrete and can be expanded to a larger number of concept classes where it is expected to outperform $d\beta$-VAE. In a particular formulation, the mixture model already outperforms $d\beta$-VAE: it discovers more detailed visual factors in a more disentangled manner. On a faces dataset, it found factors such as facial hair and glasses. This model is a promising addition to the VAE literature, both within the proposed learning framework and in other applications. It could improve performance of the hybrid neural/symbolic framework, but additional work is required to find an automated way of discovering descriptive latents, due to a lack of posterior inference over mixtures. Outside the proposed framework, an unsupervised model that discovers such detailed visual factors can for example be deployed at customs agencies around the world. Assuming a high reconstruction fidelity, which could possibly be improved by using Denoising Auto-Encoders (Vincent et al., 2010), the model allows for instant visualisation of changes in visual appearance of fugitives. Overall, the mixture model provides a promising foundation for future work, in which the model could be extended to automatically allocate an optimal number of components to each mixture.

The viability of the proposed framework has been demonstrated in a small scale proof-of-concept application expressing unwritten preferences in terms of low-level visual concepts. Ground truth concepts achieve an 88% prediction accuracy, while concept classes extracted by $d\beta$-VAE shortly follow with 86%, and the mixture model with 79%. $d\beta$-VAE is considered the baseline in the 2-category case. It is promising that the newly introduced mixture model approaches the classification accuracy of $d\beta$-VAE, while at the same time showing a better spread of information in case of a larger number of concept classes. A larger experimental setup is required to draw stronger conclusions from the two models.

The framework as a whole provides a clear advantage over the more traditional choice of an end-to-end neural network. First of all, we demonstrated that the learnt rule can be expressed in plain English. Secondly, the search for an optimal hypothesis could converge faster by involving the generative capacity of the model. The rule learnt so far could be used to reconstruct images with and without a particular concept, and based on the classification of both images, the search can be narrowed down. Overall, symbolic preference learning is more favorable than other methods, since there is a description of the preference. Its correctness can be validated by reading the rule, as opposed to validating a network's predictions on a large validation set.

7.1 Future Work

In future work, we wish to address two points. Firstly, in the current setup, each visual concept is allocated the same, fixed number of categories, and this number is determined before the training phase. Ideally, the number of categories varies per concept, such that most variation within a concept is captured by several low-variance mixture components. Stick-breaking processes (Ishwaran and James, 2001) have been explored in a VAE context to find the optimal dimensionality of the latent vector (Nalisnick and Smyth, 2016); we argue they could be explored in similar fashion within a GMM to find the number of components in each mixture. Secondly, the setup in this paper is entirely *unsupervised*, which has both advantages and disadvantages. Labelled data is scarse in the real world, but at the same time, unsupervised techniques require large amounts of training data. A semi-supervised approach would combine the advantages of both. In our setup, visual attributes are labelled with integers and visual inspection is required to convey the meaning of an attribute. A small amount of labelled data would allow for the naming of concepts, as demonstrated in Higgins et al. (2017c).

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Asai, M. and Fukunaga, A. (2017). Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. *CoRR*, abs/1705.00154.

Aubry, M., Maturana, D., Efros, A. A., Russell, B. C., and Sivic, J. (2014). Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3762–3769.

Bateux, Q., Marchand, É., Leitner, J., and Chaumette, F. (2017). Visual servoing from deep neural networks. *CoRR*, abs/1705.08940.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*.

Baxt, W. G. (1991). Use of an artificial neural network for the diagnosis of myocardial infarction. *Annals of internal medicine*, 115(11):843–848.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. University of London United Kingdom.

Bishop, C. M. (2006). Pattern recognition. *Machine Learning*, 128.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Bratko, I. (1986). *Prolog programming for artificial intelligence*. Addison-Wesley.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., and Schaub, T. (2012). Asp-core-2: Input language format. *ASP Standardization Working Group, Tech. Rep.*

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180.

Cireşan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2013). Mitosis detection in breast cancer histology images with deep neural networks. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 411–418. Springer.

Council of European Union (2016). General Data Protection Regulation (GDPR). `http://www.privacy-regulation.eu/en/22.htm`.

Craven, M. and Shavlik, J. W. (1996). Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30.

De Raedt, L. (2008). *Logical and relational learning*. Springer.

De Saussure, F. (1959). Course in general linguistics (1915). *New York: Philosophical Library.[JL]*.

Deisenroth, M. and Zafeiriou, S. (2017). Mathematics for Inference and Machine Learning Lecture Notes. `https://www.doc.ic.ac.uk/~mpd37/teaching/2016/496/notes.pdf`.

Deo, R. C., Zhang, J., Hallock, L. A., Gajjala, S., Nelson, L., Fan, E., Aras, M., Jordan, C., Fleischmann, K. E., Melisko, M., et al. (2017). An end-to-end computer vision pipeline for automated cardiac function assessment by echocardiography. *arXiv preprint arXiv:1706.07342*.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., and Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*.

Garnelo, M., Arulkumaran, K., and Shanahan, M. (2016). Towards deep symbolic reinforcement learning. *CoRR*, abs/1609.05518.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Goodman, B. and Flaxman, S. (2016). European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*.

Hazewinkel, M. (2001). Normal distribution. *Encyclopedia of Mathematics*, 4.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

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

Higgins, I., Pal, A., Rusu, A. A., Matthey, L., Burgess, C. P., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017b). Darla: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*.

Higgins, I., Sonnerat, N., Matthey, L., Pal, A., Burgess, C. P., Botvinick, M., Hassabis, D., and Lerchner, A. (2017c). SCAN: Learning Abstract Hierarchical Compositional Visual Concepts. *ArXiv e-prints*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Ishwaran, H. and James, L. F. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. (2016). Variational deep embedding: A generative approach to clustering. *CoRR*, abs/1611.05148.

Karlik, B. and Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122.

Karpathy, A. (2017). CS231n Convolutional Neural Networks for Visual Recognition. `http://cs231n.github.io`.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Klebaner, F. C. (2005). *Introduction to stochastic calculus with applications*. World Scientific Publishing Co Inc.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

Kulkarni, T. D., Whitney, W. F., Kohli, P., and Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547.

Law, M. (2017). Non monotonic Logic-Based Learning. `https://www.doc.ic.ac.uk/~ml1909/teaching/Non-monotonic%20Logic-based%20Learning,%20Summary.pdf`. [Lecture] CO-304 Logic-Based Learning, Imperial College London, Spring 2017.

Law, M., Russo, A., and Broda, K. (2014). Inductive learning of answer set programs. In *European Workshop on Logics in Artificial Intelligence*, pages 311–325. Springer.

Law, M., Russo, A., and Broda, K. (2015a). The ILASP system for learning answer set programs. `https://www.doc.ic.ac.uk/~ml1909/ILASP`.

Law, M., Russo, A., and Broda, K. (2015b). Learning weak constraints in answer set programming. *Theory and Practice of Logic Programming*, 15(4-5):511–525.

Law, M., Russo, A., and Broda, K. (2015c). Simplified reduct for choice rules in ASP. Technical report, Tech. Rep. DTR2015-2, Imperial College of Science, Technology and Medicine, Department of Computing.

Law, M., Russo, A., and Broda, K. (2016). Iterative learning of answer set programs from context dependent examples. *Theory and Practice of Logic Programming*, 16(5-6):834–848.

Law, M., Russo, A., and Broda, K. (2017). ILASP LTR Learning to Rank Answer Sets with ILASP.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Lyon, A. (2013). Why are normal distributions normal? *The British Journal for the Philosophy of Science*, 65(3):621–649.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.

Muggleton, S. (1991). Inductive logic programming. *New generation computing*, 8(4):295–318.

Muggleton, S. and De Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nalisnick, E. and Smyth, P. (2016). Stick-Breaking Variational Autoencoders. *ArXiv e-prints*.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

O'Sullivan, T., Hartley, J., Saunders, D., Montgomery, M., and Fiske, J. (1994). *Key concepts in communication and cultural studies*. Routledge London.

Rosen, K. H. (2007). Discrete mathematics and its applications. *AMC*, 10:12.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A modern approach*. Prentice-Hall, 3 edition.

Sakama, C. and Inoue, K. (2008). Brave induction. In *ILP*, volume 5194, pages 261–278. Springer.

Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Tan, J., Ung, M., Cheng, C., and Greene, C. S. (2015). Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, volume 20, page 132. NIH Public Access.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Thrun, S. B. (1994). Extracting symbolic knowledge from artificial neural networks. *Revised Version of Technical Research Report TR-IAI-93-5, Institut für Informatik III-Universität Bonn*.

Thrun, S. B., Bala, J. W., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K. A., Dzeroski, S., Fisher, D. H., Fahlman, S. E., et al. (1991). The monk's problems: A performance comparison of different learning algorithms. Technical report.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Wessels, L. F. and Barnard, E. (1992). Avoiding false local minima by proper initialization of connections. *IEEE Transactions on Neural Networks*, 3(6):899–905.

Wu, X., Ghaboussi, J., and Garrett, J. H. (1992). Use of neural networks in detection of structural damage. *Computers & structures*, 42(4):649–659.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zhou, Z.-H., Jiang, Y., and Chen, S.-F. (2003). Extracting symbolic rules from trained neural network ensembles. *Ai Communications*, 16(1):3–15.

Appendix A: Derivation of some expectations

Recall the ELBO for the 2-variable model:

$$p(X) \geq \mathbb{E}_{q(\vec{z}|\vec{x})}[p(\vec{x} \mid \vec{z})] - D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z})) \tag{A.1}$$

If we choose $p(\vec{z}) = \mathcal{N}(0, I)$, $q(\vec{z} \mid \vec{x}) = \mathcal{N}(\vec{z} \mid \vec{\mu}, \text{diag}(\vec{\sigma}^2))$ and Gaussian or Bernoulli decoder distributions $p(\vec{x} \mid \vec{z})$, both terms in Equation (A.1) can be solved analytically.

A.1 Bernoulli decoder

$$\begin{aligned}
\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] &= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln p(\vec{x} \mid \vec{z}) \, d\vec{z} \\
&= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln p(\vec{x} \mid \vec{z}) \, d\vec{z} \\
&= \ln p(\vec{x} \mid \vec{z}) \underbrace{\int_{\vec{z}} q(\vec{z} \mid \vec{x}) \, d\vec{z}}_{1} \\
&= \ln p(\vec{x} \mid \vec{z}) \\
&= \ln \text{Bernoulli}(\vec{y}) \\
&= \sum_{i=1}^{D} x_i \ln y_i + (1 - x_i) \ln(1 - y_i) \quad \text{where } \vec{x} \in \mathbb{R}^D
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z})) &= \mathbb{E}_{q(\vec{z}|\vec{x})}\left[\ln \frac{q(\vec{z} \mid \vec{x})}{p(\vec{z})}\right] \\
&= \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln q(\vec{z} \mid \vec{x}) - \ln p(\vec{z})] \\
&= \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln q(\vec{z} \mid \vec{x})] - \mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{z})] \\
&= \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln q(\vec{z} \mid \vec{x}) \, d\vec{z} - \int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln p(\vec{z}) \, d\vec{z}
\end{aligned} \tag{A.3}$$

$$\begin{aligned}
\int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln p(\vec{z}) \, d\vec{z} &= \int_{\vec{z}} \mathcal{N}(\vec{z} \mid \vec{\mu}, \text{diag}(\vec{\sigma}^2)) \cdot \ln \mathcal{N}(\vec{z} \mid 0, I) \, d\vec{z} \\
&= \int_{\vec{z}} \mathcal{N}(\vec{z} \mid \vec{\mu}, \text{diag}(\vec{\sigma}^2))(-\frac{E}{2} \ln 2\pi - \frac{1}{2} \vec{z}^T \vec{z}) \, d\vec{z} \quad \text{where } \vec{z} \in \mathbb{R}^E \\
&= -\frac{E}{2} \ln 2\pi - \frac{1}{2} \int_{\vec{z}} \mathcal{N}(\vec{z} \mid \vec{\mu}, \text{diag}(\vec{\sigma}^2)) \cdot \vec{z}^T \vec{z} \, d\vec{z} \\
&= -\frac{E}{2} \ln 2\pi - \frac{1}{2} \mathbb{E}_{q(\vec{z}|\vec{x})}\left[\vec{z}^T \vec{z}\right] \\
&= -\frac{E}{2} \ln 2\pi - \frac{1}{2} \mathbb{E}_{q(\vec{z}|\vec{x})}\left[\sum_{i=1}^{E} z_i^2\right] \\
&= -\frac{E}{2} \ln 2\pi - \frac{1}{2} \sum_{i=1}^{E} \mathbb{E}_{q(\vec{z}|\vec{x})}\left[z_i^2\right] \\
&= -\frac{E}{2} \ln 2\pi - \frac{1}{2} \sum_{i=1}^{E} (\mu_i^2 + \text{Cov}[z_i, z_i]) \qquad (\text{Cov}[x_i, x_j] = \mathbb{E}[x_i x_j] - \mu_i \mu_j)
\end{aligned}$$

$$= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}(\mu_i^2 + \sigma_i^2) \tag{A.4}$$

$$\begin{aligned}
\int_{\vec{z}} q(\vec{z} \mid \vec{x}) \ln q(\vec{z} \mid \vec{x})\, d\vec{z} &= \int_{\vec{z}} \mathcal{N}(\vec{z} \mid \vec{\mu}, \mathrm{diag}(\vec{\sigma}^2)) \cdot \ln \mathcal{N}(\vec{z} \mid \vec{\mu}, \mathrm{diag}(\vec{\sigma}^2))\, d\vec{z} \\
&= \int_{\vec{z}} \mathcal{N}(\vec{z} \mid \vec{\mu}, \mathrm{diag}(\vec{\sigma}^2)) \cdot \\
&\quad (-\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 - \frac{1}{2}(\vec{z}-\vec{\mu})^T \mathrm{diag}(\vec{\sigma}^2)^{-1}(\vec{z}-\vec{\mu}))\, d\vec{z} \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 \\
&\quad - \frac{1}{2}\int_{\vec{z}} \mathcal{N}(\vec{z} \mid \vec{\mu}, \mathrm{diag}(\vec{\sigma}^2)) \cdot ((\vec{z}-\vec{\mu})^T \mathrm{diag}(\vec{\sigma}^2)^{-1}(\vec{z}-\vec{\mu}))\, d\vec{z} \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 \\
&\quad - \frac{1}{2}\int_{\vec{z}} \mathcal{N}(\vec{z} \mid \vec{\mu}, \mathrm{diag}(\vec{\sigma}^2)) \cdot \mathrm{tr}[(\vec{z}-\vec{\mu})(\vec{z}-\vec{\mu})^T \mathrm{diag}(\vec{\sigma}^2)^{-1}]\, d\vec{z} \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 - \frac{1}{2}\mathbb{E}_{q(\vec{z}|\vec{x})}[\mathrm{tr}[(\vec{z}-\vec{\mu})(\vec{z}-\vec{\mu})^T \mathrm{diag}(\vec{\sigma}^2)^{-1}]] \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 - \frac{1}{2}\mathrm{tr}[\mathbb{E}_{q(\vec{z}|\vec{x})}[(\vec{z}-\vec{\mu})(\vec{z}-\vec{\mu})^T \mathrm{diag}(\vec{\sigma}^2)^{-1}]] \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 - \frac{1}{2}\mathrm{tr}[\mathbb{E}_{q(\vec{z}|\vec{x})}[(\vec{z}-\vec{\mu})(\vec{z}-\vec{\mu})^T]\mathrm{diag}(\vec{\sigma}^2)^{-1}] \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 - \frac{1}{2}\mathrm{tr}[\mathrm{diag}(\vec{\sigma}^2)\mathrm{diag}(\vec{\sigma}^2)^{-1}] \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 - \frac{1}{2}\mathrm{tr}[I] \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}\ln \sigma_i^2 - \frac{1}{2}\sum_{i=1}^{E} 1 \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}(1 + \ln \sigma_i^2)
\end{aligned} \tag{A.5}$$

Plugging Equation (A.4) and Equation (A.5) into Equation (A.3):

$$\begin{aligned}
D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z})) &= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}(1 + \ln \sigma_i^2) - (-\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{E}(\mu_i^2 + \sigma_i^2)) \\
&= -\frac{1}{2}\sum_{i=1}^{E}(1 + \ln \sigma_i^2 - \mu_i^2 - \sigma_i^2)
\end{aligned}$$

The full lower bound is then:

$$\begin{aligned}
\ln p(x) &\geq \mathbb{E}_q[\ln p(\vec{x} \mid \vec{z})] - D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z})) \\
&= \sum_{i=1}^{D} x_i \ln y_i + (1 - x_i)\ln(1 - y_i) + \frac{1}{2}\sum_{i=1}^{E}(1 + \ln \sigma_i^2 - \mu_i^2 - \sigma_i^2)
\end{aligned} \tag{A.6}$$

A.2 Gaussian decoder

Similarly, for a decoder with Gaussian outputs we obtain:

$$\begin{aligned}\mathbb{E}_{q(\vec{z}|\vec{x})}[\ln p(\vec{x} \mid \vec{z})] &= \ln p(\vec{x} \mid \vec{z}) = \ln \mathcal{N}(\vec{x} \mid \vec{\mu_2}, \mathrm{diag}(\vec{\sigma_2}^2)) \\ &= -\frac{D}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{D}\ln \sigma_{2i}^2 - \frac{1}{2}(\vec{x}-\vec{\mu_2})^T \mathrm{diag}(\vec{\sigma_2^2})^{-1}(\vec{x}-\vec{\mu_2}) \end{aligned} \tag{A.7}$$

$$D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z})) = -\frac{1}{2}\sum_{i=1}^{E}(1 + \ln \sigma_i^2 - \mu_i^2 - \sigma_i^2) \tag{A.8}$$

$$\begin{aligned}\ln p(X) &\geq \mathbb{E}_q[\ln p(\vec{x} \mid \vec{z})] - D_{KL}(q(\vec{z} \mid \vec{x}) \,||\, p(\vec{z})) \\ &= -\frac{D}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{D}\ln \sigma_{2i}^2 - \frac{1}{2}(\vec{x}-\vec{\mu_2})^T \mathrm{diag}(\vec{\sigma_2^2})^{-1}(\vec{x}-\vec{\mu_2}) \\ &+ \frac{1}{2}\sum_{i=1}^{E}(1 + \ln \sigma_i^2 - \mu_i^2 - \sigma_i^2) \end{aligned} \tag{A.9}$$

A.3 Expectation of log Gaussian w.r.t. Gaussian

Given two multivariate Gaussian distributions $p(\vec{z})$ and $q(\vec{z})$, with $\vec{z} \in \mathbb{R}^E$:

$$\begin{aligned} p(\vec{z}) &= \mathcal{N}(\vec{\mu}_1, \mathrm{diag}(\vec{\sigma}_1^2)) \\ q(\vec{z}) &= \mathcal{N}(\vec{\mu}_2, \mathrm{diag}(\vec{\sigma}_2^2)) \end{aligned}$$

we can calculate the expectation of the logarithm of one w.r.t. the other in closed form:

$$\begin{aligned}\int_{\vec{z}} p(\vec{z}) \ln q(\vec{z})\, d\vec{z} &= \int_{\vec{z}} p(\vec{z})[-\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln \sigma_{2,i}^2 - \frac{1}{2}(\vec{z}-\vec{\mu_2})^T \mathrm{diag}(\sigma_2^2)^{-1}(\vec{z}-\vec{\mu_2})]\, d\vec{z} \\ &= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln \sigma_{2,i}^2 - \frac{1}{2}\int_{\vec{z}} p(\vec{z})(\vec{z}-\vec{\mu_2})^T \mathrm{diag}(\sigma_2^2)^{-1}(\vec{z}-\vec{\mu_2})\, d\vec{z} \\ &= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln \sigma_{2,i}^2 - \frac{1}{2}\mathbb{E}_{p(\vec{z})}[(\vec{z}-\vec{\mu_2})^T \mathrm{diag}(\sigma_2^2)^{-1}(\vec{z}-\vec{\mu_2})] \\ &= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln \sigma_{2,i}^2 - \frac{1}{2}\mathbb{E}_{p(\vec{z})}[\vec{z}^T \mathrm{diag}(\sigma_2^2)^{-1}\vec{z} - 2\vec{\mu}_2^T \mathrm{diag}(\sigma_2^2)^{-1}\vec{z} \\ &\quad + \vec{\mu}_2^T \mathrm{diag}(\sigma_2^2)^{-1}\vec{\mu}_2] \\ &= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln \sigma_{2,i}^2 - \frac{1}{2}\mathbb{E}_{p(\vec{z})}\left[\sum_{i}^{E}\frac{z_i^2}{\sigma_{2,i}^2} - 2\sum_{i}^{E}\frac{\mu_{2,i}\cdot z_i}{\sigma_{2,i}^2} + \sum_{i}^{E}\frac{\mu_{2,i}^2}{\sigma_{2,i}^2}\right] \\ &= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln \sigma_{2,i}^2 - \frac{1}{2}\sum_{i}^{E}\mathbb{E}_{p(\vec{z})}\left[\frac{z_i^2}{\sigma_{2,i}^2} - 2\frac{\mu_{2,i}\cdot z_i}{\sigma_{2,i}^2} + \frac{\mu_{2,i}^2}{\sigma_{2,i}^2}\right] \\ &= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln \sigma_{2,i}^2 - \frac{1}{2}\sum_{i}^{E}\mathbb{E}_{p(\vec{z})}\left[\frac{z_i^2}{\sigma_{2,i}}\right] + \sum_{i}^{E}\mathbb{E}_{p(\vec{z})}\left[\frac{\mu_{2,i}\cdot z_i}{\sigma_{2,i}^2}\right] \\ &\quad - \frac{1}{2}\sum_{i}^{E}\mathbb{E}_{p(\vec{z})}\left[\frac{\mu_{2,i}^2}{\sigma_{2,i}^2}\right] \end{aligned}$$

$$
\begin{aligned}
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln\sigma_{2,i}^2 - \frac{1}{2}\sum_{i}^{E}\frac{\mathbb{E}_{p(\vec{z})}\left[z_i^2\right]}{\sigma_{2,i}^2} + \sum_{i}^{E}\frac{\mu_{2,i}\cdot\mathbb{E}_{p(\vec{z})}[z_i]}{\sigma_{2,i}^2} \\
&\quad - \frac{1}{2}\sum_{i}^{E}\frac{\mathbb{E}_{p(\vec{z})}\left[\mu_{2,i}^2\right]}{\sigma_{2,i}^2} \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln\sigma_{2,i}^2 - \frac{1}{2}\sum_{i}^{E}\frac{(\mu_{1,i}^2+\sigma_{1,i}^2)}{\sigma_{2,i}^2} + \sum_{i}^{E}\frac{\mu_{2,i}\cdot\mu_{1,i}}{\sigma_{2,i}^2} \\
&\quad - \frac{1}{2}\sum_{i}^{E}\frac{\mu_{2,i}^2}{\sigma_{2,i}^2} \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln\sigma_{2,i}^2 - \sum_{i}^{E}\frac{\sigma_{1,i}^2}{2\sigma_{2,i}^2} - \sum_{i}^{E}\frac{\mu_{1,i}^2 - 2\mu_{2,i}\cdot\mu_{1,i}+\mu_{2,i}^2}{2\sigma_{2,i}^2} \\
&= -\frac{E}{2}\ln 2\pi - \frac{1}{2}\sum_{i}^{E}\ln\sigma_{2,i}^2 - \sum_{i}^{E}\frac{\sigma_{1,i}^2+(\mu_{1,i}-\mu_{2,i})^2}{2\sigma_{2,i}^2}
\end{aligned}
\tag{A.10}
$$

Appendix B: Mixture model latent value iterations

Figure B.1: Overview of latent factors found by the mixture model VAE with different priors, compared with the baseline β-VAE.

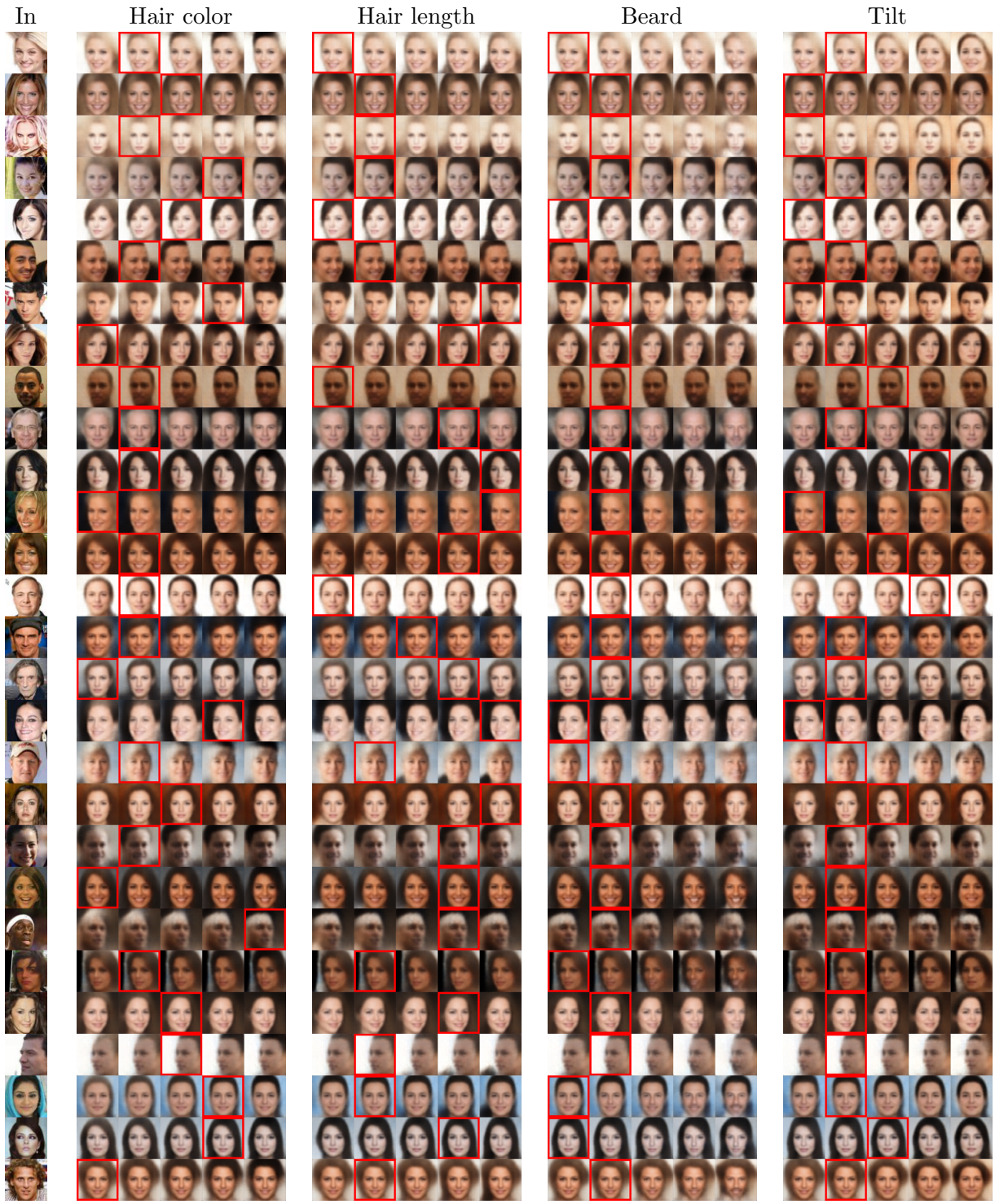


Figure B.2: Extra iterations over the 5 univariate Gaussian means on a model trained according to the first approach considered in Section 5.2.

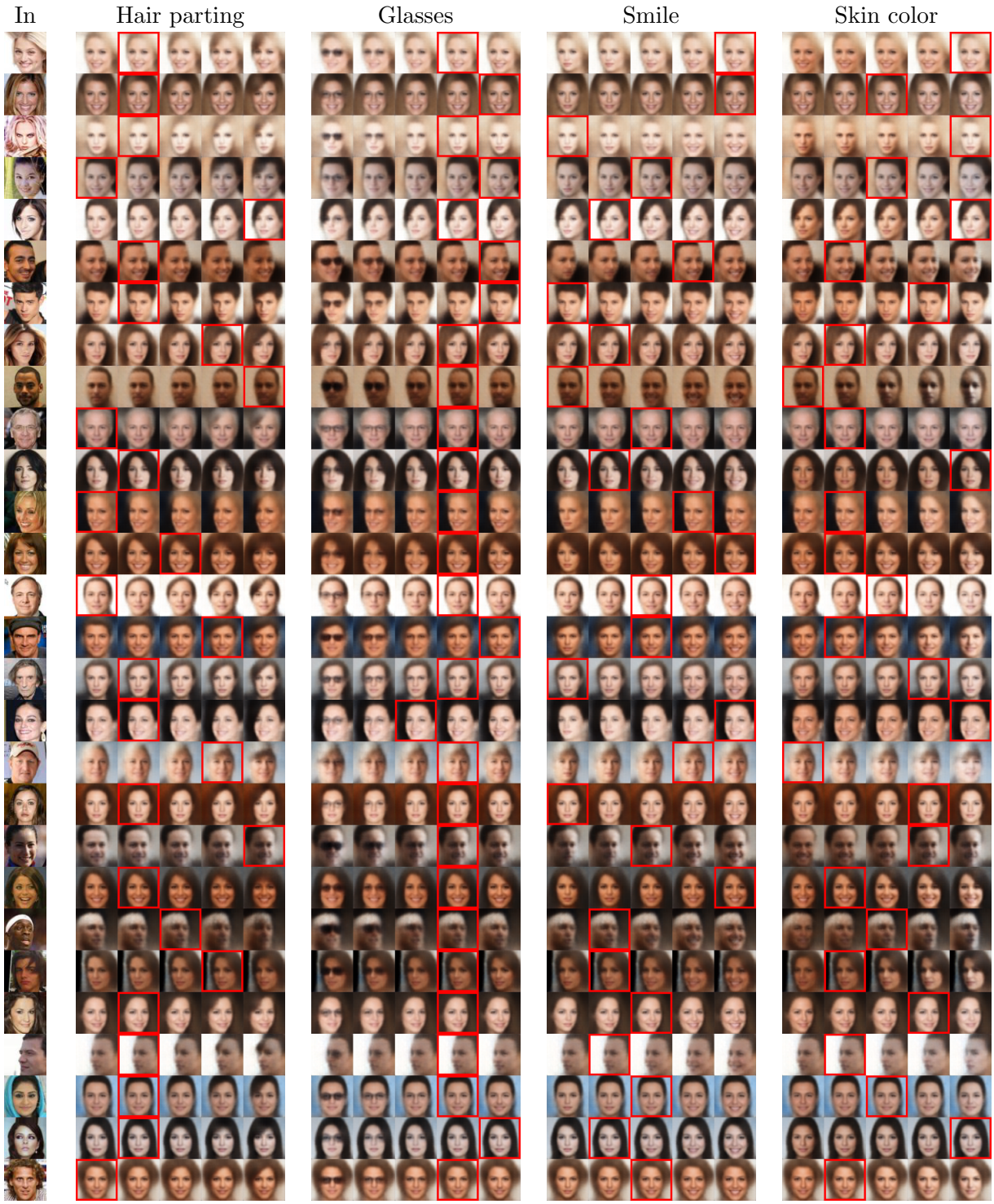


Figure B.3: Extra iterations over the 5 univariate Gaussian means on a model trained according to the first approach considered in Section 5.2.

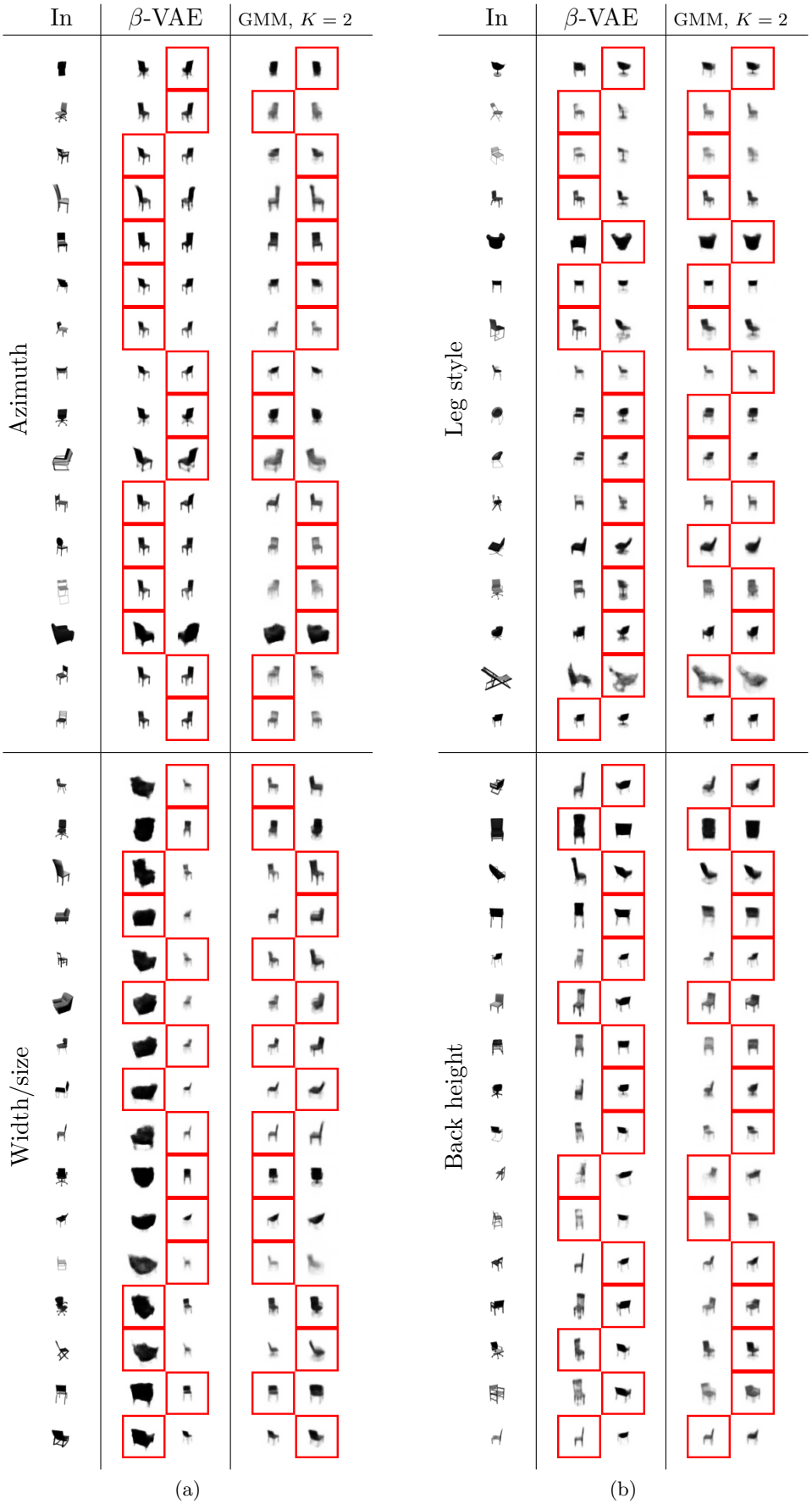


Figure B.4: Additional iterations over 2-category chairs.

Appendix C: Network architectures

	Distr.	Input	Hidden	Output
Faces (discr.)	$q(\vec{z} \mid \vec{x})$	64x64x3	Conv 32-4x4 (2), Conv 32-4x4 (2), Conv 64-4x4 (2), Conv 64-4x4 (2), Flatten, FC-256	• FC-(32x5) (softplus)
	$p(\vec{x} \mid \vec{z})$	32x5	FC-256, FC-1024, Reshape (4, 4, 64), Deconv 64-4x4 (2), Deconv 32-4x4 (2), Deconv 32-4x4 (2)	• Deconv 3-4x4 (2) (none) • Deconv 3-4x4 (2) (none)
Chairs (discr.)	$q(\vec{z} \mid \vec{x})$	64x64x1	Conv 32-4x4 (2), Conv 32-4x4 (2), Conv 64-4x4 (2), Conv 64-4x4 (2), Flatten, FC-256	• FC-(32x5) (softplus)
	$p(\vec{x} \mid \vec{z})$	32x5	FC-256, FC-1024, Reshape (4, 4, 64), Deconv 64-4x4 (2), Deconv 32-4x4 (2), Deconv 32-4x4 (2)	• Deconv 1-4x4 (2) (sigmoid)
Faces (mixt.)	$q(\vec{z} \mid \vec{x}, \vec{y})$	64x64x3	Conv 32-4x4 (2), Conv 32-4x4 (2), Conv 64-4x4 (2), Conv 64-4x4 (2), Flatten, FC-256	• FC-(32x5) (none) • FC-(32x5) (none)
	$q(\vec{y} \mid \vec{x})$	64x64x3	**, FC-256	• FC-(32x5) (softplus)
	$p(\vec{x} \mid \vec{z})$	32x1	FC-256, FC-1024, Reshape (4, 4, 64), Deconv 64-4x4 (2), Deconv 32-4x4 (2), Deconv 32-4x4 (2)	• Deconv 3-4x4 (2) (none) • Deconv 3-4x4 (2) (none)
Chairs (mixt.)	$q(\vec{z} \mid \vec{x}, \vec{y})$	64x64x1	Conv 32-4x4 (2), Conv 32-4x4 (2), Conv 64-4x4 (2), Conv 64-4x4 (2), Flatten, FC-256	• FC-(32x5) (none) • FC-(32x5) (none)
	$q(\vec{y} \mid \vec{x})$	64x64x1	**, FC-256	• FC-(32x5) (softplus)
	$p(\vec{x} \mid \vec{z})$	32x1	FC-256, FC-1024, Reshape (4, 4, 64), Deconv 64-4x4 (2), Deconv 32-4x4 (2), Deconv 32-4x4 (2)	• Deconv 1-4x4 (2) (sigmoid)

Table C.1: Network architectures for the VAE models used in Chapter 5. The output activation of each hidden layer is ReLU. Conv 32-4x4 (2) denotes a convolution layer with 32 filters each of size 4x4, and the filters are strided over the input with step size 2 both along the height and width dimension. Deconv denotes a deconvolution[1] layer. **The convolutional layers are shared with $q(\vec{z} \mid \vec{x}, \vec{y})$.

[1] `https://www.tensorflow.org/api_docs/python/tf/layers/conv2d_transpose`

Input	Hidden	Output
64	FC=10	FC-2 (softmax)

Table C.2: Network architecture for the MLP classifier from Table 6.1. Hidden layer activations are ReLU. The network is trained on mean vectors μ obtained from β-VAE instead of full chair inputs. Full chair inputs require convolutional layers, which are associated with a large number of parameters. The size of the training dataset (180 chairs) is insufficient to properly train those parameters.

Appendix D: Learning tasks

D.1 Learning Leg Style

```
#modeo(1, attr(1, const(val))).
#modeo(1, attr(2, const(val))).
#modeo(1, attr(3, const(val))).
#modeo(1, attr(4, const(val))).
#modeo(attr(const(att), var(v))).

#constant(att, 1).
#constant(att, 2).
#constant(att, 3).
#constant(att, 4).
#constant(val, 1).
#constant(val, 2).
#weight(1).
#weight(v).

#order(p1@5, {attr(1,2). attr(2,1). attr(3,2). attr(4,2).}, {attr(1,2).
    attr(2,1). attr(3,1). attr(4,1).}).
#order(p2@5, {attr(1,2). attr(2,1). attr(3,2). attr(4,1).}, {attr(1,1).
    attr(2,2). attr(3,1). attr(4,1).}).
#order(p3@5, {attr(1,2). attr(2,2). attr(3,2). attr(4,1).}, {attr(1,1).
    attr(2,2). attr(3,1). attr(4,2).}).
#order(p4@5, {attr(1,2). attr(2,2). attr(3,2). attr(4,1).}, {attr(1,1).
    attr(2,1). attr(3,1). attr(4,2).}).
#order(p5@5, {attr(1,2). attr(2,1). attr(3,2). attr(4,1).}, {attr(1,1).
    attr(2,1). attr(3,1). attr(4,1).}).
#order(p6@5, {attr(1,2). attr(2,1). attr(3,2). attr(4,2).}, {attr(1,1).
    attr(2,1). attr(3,1). attr(4,1).}).
#order(p7@5, {attr(1,1). attr(2,1). attr(3,2). attr(4,2).}, {attr(1,2).
    attr(2,2). attr(3,1). attr(4,2).}).
#order(p8@5, {attr(1,1). attr(2,1). attr(3,2). attr(4,1).}, {attr(1,2).
    attr(2,1). attr(3,1). attr(4,1).}).
#order(p9@5, {attr(1,2). attr(2,2). attr(3,2). attr(4,1).}, {attr(1,1).
    attr(2,2). attr(3,1). attr(4,1).}).
#order(p10@5, {attr(1,2). attr(2,2). attr(3,2). attr(4,2).}, {attr(1,1).
    attr(2,2). attr(3,1). attr(4,2).}).
#order(p11@5, {attr(1,2). attr(2,2). attr(3,2). attr(4,1).}, {attr(1,1).
    attr(2,1). attr(3,1). attr(4,1).}).
#order(p12@5, {attr(1,2). attr(2,2). attr(3,2). attr(4,2).}, {attr(1,2).
    attr(2,2). attr(3,1). attr(4,1).}).
#order(p13@5, {attr(1,1). attr(2,1). attr(3,2). attr(4,2).}, {attr(1,1).
    attr(2,2). attr(3,1). attr(4,2).}).
#order(p14@5, {attr(1,1). attr(2,1). attr(3,2). attr(4,1).}, {attr(1,1).
    attr(2,2). attr(3,1). attr(4,2).}).
#order(p15@5, {attr(1,1). attr(2,1). attr(3,2). attr(4,1).}, {attr(1,1).
    attr(2,2). attr(3,1). attr(4,2).}).
#order(p16@5, {attr(1,1). attr(2,2). attr(3,2). attr(4,1).}, {attr(1,2).
    attr(2,1). attr(3,1). attr(4,2).}).
```

D.2 Monk's 1

```
#modeh(monk).
#modeb(1,jacket_color(var(color)), (positive)).
```

```
#modeb(1,head_shape(var(shape)), (positive)).
#modeb(1,body_shape(var(shape)), (positive)).
#modeb(1,holding(var(item)), (positive)).
#modeb(1,is_smiling(var(truth)), (positive)).
#modeb(1,has_tie(var(truth)), (positive)).

#modeb(1,red(var(color))).
#modeb(1,yellow(var(color))).
#modeb(1,green(var(color))).
#modeb(1,blue(var(color))).

#modeb(2,round(var(shape))).
#modeb(2,square(var(shape))).
#modeb(2,octagon(var(shape))).

#modeb(2,yes(var(truth))).
#modeb(2,no(var(truth))).

#modeb(1,sword(var(item))).
#modeb(1,balloon(var(item))).
#modeb(1,flag(var(item))).

round(1). square(2). octagon(3).
yes(1). no(2).
sword(1). balloon(2). flag(3).
red(1). yellow(2). green(3). blue(4).
```

D.3 Monk's 3

```
#modeh(monk).
#modeb(1, head_shape(var(shape))).
#modeb(1, head_shape(const(shape))).

#modeb(1, body_shape(var(shape))).
#modeb(1, body_shape(const(shape))).

#modeb(1, jacket_color(var(color))).
#modeb(1, jacket_color(const(color))).

#modeb(1, is_smiling).
#modeb(1, has_tie).

#modeb(1, holding(var(item))).
#modeb(1, holding(const(item))).

#constant(color, green).
#constant(color, red).
#constant(color, blue).
#constant(color, yellow).

#constant(shape, round).
#constant(shape, square).
#constant(shape, octagon).

#constant(item, sword).
#constant(item, balloon).
#constant(item, flag).
```