Semi-Supervised Multimodal Learning with Generative Models

by

Punit Shah

Supervisors: Prof. David Barber and Hippolyt Ritter

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in

Machine Learning

Department of Computer Science
University College London

September 30, 2017

This report is submitted as part requirement for the MSc in Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text.

Abstract

Multimodal learning is about discovering relationships across multiple information sources. We investigate the multimodal learning problem between two image modalities. A shared representation of these modalities is efficiently learned using a probabilistic model in a semi-supervised setting where only a few paired examples are required. A novel formulation of the model is proposed which can scale more effectively to larger numbers of modalities. We investigate the extent to which the shared representation captures the data, and we also assess the model's ability to infer one modality from the other.

For accompanying code, see:
`https://github.com/punit-haria/multimodal-learning`

Acknowledgements

I would like to thank Prof. David Barber and Hippolyt Ritter for their unwavering support and invaluable insight without which this would not have been possible.

Contents

List of Figures

List of Tables

LIST OF TABLES

1. Introduction

The world is a multitude of coexisting modalities for both animals and machines. Animals use their many sensory organs in synchrony to learn from and react to the environment, and increasingly, artificial agents must also leverage multiple information sources to make predictions and take actions. The problem of usefully merging these information sources is called *multimodal learning*, and it is a formidable problem when the sources are highly structured, e.g. images, audio, text.

In this thesis, we investigate the multimodal learning problem between two visual modalities. We can think of each modality as a different view of some object in the world. Each view may only paint a partial picture of the underlying concept, and the challenge is then to merge all modalities to attain a complete picture. We do this by learning a joint representation of the image modalities which can be effectively used for downstream decision making. The advantage of our approach is that it is *semi-supervised*; very few paired examples of the two modalities are required, and instead a large number of unpaired examples are drawn upon. Moreover, the learning algorithm is efficient and scales to large datasets. This setting is highly characteristic of real-world scenarios where annotating data is expensive and yet vast amounts of unlabelled data exist, motivating the need for scalable algorithms.

Our model is a generative model, which means that it can also simulate the underlying data generating process. This is a powerful capability which can be used to imagine possible configurations of these modalities. The generative mechanism can also be used to infer one modality from the other, which is desirable in situations with missing information. We propose a novel variation of this model which improves the quality of these translations and easily scales to a large number of modalities. Our approach has a probabilistic underpinning which means that uncertainty can be captured and expressed in situations where information sources are ambiguous or noisy.

In our experiments, we make two key evaluations:

1. Visual evaluation of the model's ability to infer one modality from the other.
2. Evaluation of the extent to which the joint representation simultaneously captures the two modalities.

These evaluations are made on a variety of datasets, and we find promising results in both these contexts.

1.1 Our Contribution

Our contribution is in the application of an efficient probabilistic approach to learning from multiple image modalities in a semi-supervised setting. We also propose a variant to our model which more effectively scales to a large number of modalities.

1.2 Thesis Outline

Chapter 2 introduces foundational concepts and formalizes the problem in the language of probability theory. The learning and inference algorithms we use are described in chapter 3. In chapter 4, we contextualize the problem of multimodal learning within the probabilistic framework. In chapter 5, we introduce neural networks which are a core architectural component in our models and learning algorithms. Chapter 6 outlines related and competing methods in this area, and in chapter 7, we highlight the results of our experiments. Finally, we conclude and identify possible future directions in chapter 8.

2. Foundations

In this chapter, we describe basic elements of probability theory and introduce our notation. The language of probability is then used to formalize the problems of generative modeling and representation learning. Here we focus on modeling a single input modality, and defer the case of multiple modalities to chapter 4.

First, one may ask, why probability? Suppose we observe a sequence of inputs $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$, and are given no further information. We are uncertain about the underlying data generating process, so any subsequent reasoning adheres implicitly or explicitly to some kind of probabilistic framework. The laws of probability are correct in the sense that they can be derived from a small set of axioms which only enforce logical coherence. See Hájek et al. [2008] for one such argument. And so, this is the approach we maintain going forward. In the next section, we briefly introduce foundational concepts while setting our notation.

2.1 Definitions and Notation

A random or stochastic variable $\mathbf{x}$ is a variable whose outcomes are governed by a random process. The variable $\mathbf{x}$ may be discrete or continuous, and will usually be multidimensional. Our notation does not distinguish between stochastic variables and observations, but this will be clear from context. We denote $p(\mathbf{x})$ to be a probability distribution over the possible values taken by $\mathbf{x}$. For continuous $\mathbf{x}$ this is a probability density function, and for discrete $\mathbf{x}$ this is a probability mass function. In the context of two random variables $\mathbf{x}$ and $\mathbf{y}$, the product rule of probability leads to *Bayes' theorem*, which states that:

$$p(\mathbf{y}|\mathbf{x}) \;=\; \frac{p(\mathbf{x}|\mathbf{y})p(\mathbf{y})}{p(\mathbf{x})}.$$

Here $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{y})$ are conditional distributions, and $p(\mathbf{x})$ and $p(\mathbf{y})$ are marginal distributions. If the two random variables are *independent*, then their joint distribution can be factorized, i.e. $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$.

The *expected value* of a random variable $\mathbf{x}$ is denoted:

$$\mathbb{E}_{p(\mathbf{x})}[\mathbf{x}] \;=\; \int \mathbf{x}\, p(\mathbf{x}) d\mathbf{x},$$

where the integral is replaced with a summation for discrete variables. Here we also introduce the *Gaussian* distribution, which is a continuous distribution over $\mathbf{x} \in \mathbb{R}^m$ defined as:

$$p(\mathbf{x}) \;=\; \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) \;=\; \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) \right),$$

where $\boldsymbol{\mu} \in \mathbb{R}^m$ and $\Sigma \in \mathbb{R}^{m \times m}$ are the mean and covariance parameters respectively. We also use the notation $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ to indicate the distribution of $\mathbf{x}$.

2.2 Problem Setting

Let $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be a dataset which we consider to be independent realizations of an m-dimensional stochastic variable $\mathbf{x}$. We assume each observation is instantiated via the following random process. An unobserved or *latent* variable $\mathbf{z} \in \mathbb{R}^k$ is drawn from distribution $p(\mathbf{z})$, and consequently an observation $\mathbf{x}$ is generated from conditional distribution $p(\mathbf{x}|\mathbf{z})$. The joint probability $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ is known as the generative model, and is depicted graphically in figure 2.1.

In this two-step generative process, we can interpret $\mathbf{z}$ as the explanatory factors of variation which underpin the observed data. Bayes' theorem can be used to further deconstruct this relationship:

$$p(\mathbf{z}|\mathbf{x}) \;=\; \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}. \tag{2.1}$$

Here the *prior* probability $p(\mathbf{z})$ reflects the modeler's initial beliefs about the latent space, and the *posterior* $p(\mathbf{z}|\mathbf{x})$ is an updated distribution after observing the evidence $\mathbf{x}$. The posterior is really an inversion of the generative process whereby we infer a latent representation $\mathbf{z}$ from the observation $\mathbf{x}$. Here, and throughout this thesis, $\mathbf{z}$ is

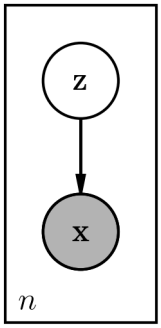


Figure 2.1: Generative model in graphical notation where the arrow expresses the conditional dependency structure of the model. The surrounding plate notation indicates n independent replications, one for each observation in dataset $\mathcal{D}$.

of lower dimension than $\mathbf{x}$, serving as a kind of compression of the information in $\mathbf{x}$. The distribution $p(\mathbf{x}|\mathbf{z})$ is called the *likelihood*, and

$$p(\mathbf{x}) \;=\; \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} \tag{2.2}$$

is known as the *marginal likelihood* or *model evidence*. For the n independent observations of dataset $\mathcal{D}$, this becomes $\prod_{i=1}^{n} p(\mathbf{x}_i)$.

In this exposition, we have unearthed all the elements required to both understand the generative process and to learn representations of the data. We are interested in deducing the properties of these distributions, and we do this by way of parameterization.

2.3 Learning and Inference

A parametric model is one that belongs to a family of distributions parameterized by a variable $\boldsymbol{\theta}$. This is to say that each distribution in the family is uniquely identified by a setting of the parameter $\boldsymbol{\theta}$. We denote this explicitly as $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ for the generative model of figure 2.1, but may sometimes omit the $\boldsymbol{\theta}$ for brevity. The process of finding the parameter value which maximizes the probability of the data $\mathcal{D}$ is known as *maximum likelihood*. More formally we seek:

$$\hat{\boldsymbol{\theta}} \;=\; \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathcal{D}) \;=\; \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{n} p(\mathbf{x}_i). \tag{2.3}$$

The term *learning* is used to describe any process which does this. Note that in practice we maximize the log-probability instead, to avoid floating point underflow when $p(\mathbf{x})$ is extremely small, and also because it tends to simplify both analytical and numerical approaches. The logarithm is monotonically increasing so both problems are equivalent.

The problem of deducing the properties of the posterior distribution from data is known as *inference*. The direct approach is to learn the parameters $\boldsymbol{\theta}$ via maximum likelihood, and then derive the posterior using Bayes' rule. The feasibility of this approach of course depends on the specifics of the generative model. In the next section, we further develop this model.

2.4 Modeling Framework

How do we choose the likelihood and prior distributions in our generative model? The standard approach is to use the *exponential family*, which is a parametric family that makes inference and learning analytically tractable in many cases. It can be justified by evoking the principle of maximum entropy [Jaynes, 1957] which states that one should choose the distribution which makes the fewest assumptions beyond one's prior beliefs. The exponential family is the solution to a formalization of this problem [Wainwright et al., 2008], and it includes the Gaussian, Bernoulli, and categorical distributions.

We motivate our modeling approach with an example. Consider the factor analysis model where the likelihood distribution is a Gaussian $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\Lambda\mathbf{z}, \Psi)$ with parameters $\Lambda \in \mathbb{R}^{m\times k}$ and $\Psi \in \mathbb{R}^{m\times m}$. Here we can equivalently think of the random variable $\mathbf{x}$ as a function of a deterministic variable $\mathbf{z}$ and an additional stochastic noise variable $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ (where $\mathbf{I}$ denotes the identity matrix):

$$\mathbf{x} \;=\; \Lambda\mathbf{z} + \Psi^{-\frac{1}{2}}\boldsymbol{\epsilon}. \tag{2.4}$$

Importantly there is a linear relationship between $\mathbf{x}$ and $\mathbf{z}$. If the prior probability is also Gaussian, e.g. $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$, then the posterior $p(\mathbf{z}|\mathbf{x})$ and marginal $p(\mathbf{x})$ are Gaussian [Bishop, 2006]. An analytical solution is not available for the problem of maximizing $p(\mathbf{x})$ with respect to $\boldsymbol{\theta} = \{\Lambda, \Psi\}$, but iterative procedures such as the EM algorithm [Dempster et al., 1977] are applicable and efficient. Factor analysis is part

of a larger class known as linear Gaussian models [Roweis and Ghahramani, 1999], which includes many prevailing statistical methods for multidimensional data, e.g. hidden Markov models, Kalman filters.

The fundamental commonality across these models is the linear relationship between the observed and latent variables, exemplified in equation 2.4. This can be quite limiting especially in the case of rich and high-dimensional data such as natural images. An ideal setup should allow for the modeling of arbitrarily complex interactions between $\mathbf{x}$ and $\mathbf{z}$. Suppose we define the likelihood to be $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}), \Sigma_{\boldsymbol{\theta}}(\mathbf{z}))$ where $\boldsymbol{\mu}$ and Σ are functions of $\mathbf{z}$, parameterized by $\boldsymbol{\theta}$. In this case, the relationship between $\mathbf{x}$ and $\mathbf{z}$ is:

$$\mathbf{x} \;=\; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}) + \Sigma_{\boldsymbol{\theta}}(\mathbf{z})^{-\frac{1}{2}}\boldsymbol{\epsilon} \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \tag{2.5}$$

Here we have the freedom to construct highly nonlinear functional forms for $\boldsymbol{\mu}$ and Σ, giving us an incredibly flexible and powerful model. Moreover, we do not need to restrict ourselves to the Gaussian and may instead choose any distribution of the form:

$$p(\mathbf{x}|f_{\boldsymbol{\theta}}(\mathbf{z})) \tag{2.6}$$

where f is a nonlinear function of $\mathbf{z}$. This is our model of choice going forward.

In chapter 3, we describe an efficient and general-purpose method for inference and learning which can be applied to this model.

3. Approximate Methods

Consider the general case of parameterizing the likelihood and prior probabilities as $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{z})$. The problem of maximum likelihood is intractable because we do not assume a closed-form expression for the marginal $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$. A natural way to proceed is to maximize an approximation of $p(\mathbf{x})$ with respect to the model parameters $\boldsymbol{\theta}$. The EM algorithm [Dempster et al., 1977] is typically used in this setting but requires that the posterior be computed analytically. We make no such assumptions. Another approach is to use Monte Carlo or sampling-based methods, e.g. Neal [1993]. For example, we may be able to estimate the integral by taking the empirical average of the conditional $p(\mathbf{x}|\mathbf{z})$, evaluated at multiple values of $\mathbf{z}$ sampled from the prior $p(\mathbf{z})$. Such an estimator is unbiased, meaning it converges to the true marginal $p(\mathbf{x})$ as the number of samples goes to infinity. However the sampling procedure can be computationally expensive and a large number of samples may be required for a sufficiently accurate estimate. This makes it intractable for large datasets. Other methods in this camp such as Monte Carlo EM [Christian and Casella, 1999] also have this scalability problem. We instead turn to *variational* methods [Jordan et al., 1999] which estimate the marginal in a way that makes the maximum likelihood problem tractable and scalable.

3.1 Variational Lower Bound

The idea is to formulate a lower bound on $\log p(\mathbf{x})$ which can be efficiently maximized using gradient-based methods (section 3.2). The hope is that a sufficiently tight bound on the log-marginal will emerge. Although the optimization process will converge, there is no clear way to predict how close the bound is to the true maximum likelihood solution [Beal, 2003]. In this way, variational inference trades accuracy for

computational efficiency when compared with Monte Carlo methods, but it is unclear what the relative accuracy is between the two approaches [Blei et al., 2016]. In practice, variational methods perform well, especially with large datasets [Blei et al., 2003, Kucukelbir et al., 2015].

To construct the lower bound, we utilize a general result known as Jensen's inequality [Jensen, 1906] of which one outcome is that $\log \mathbb{E}[f(\mathbf{z})] \geq \mathbb{E}[\log f(\mathbf{z})]$ for a function f of a random variable $\mathbf{z}$. We also introduce the *variational distribution* $q(\mathbf{z}|\mathbf{x})$. A lower bound on $\log p(\mathbf{x})$ is derived in the following way:

$$\log p(\mathbf{x}) = \log \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z} \tag{3.1}$$

$$= \log \int \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} q(\mathbf{z}|\mathbf{x}) d\mathbf{z} \tag{3.2}$$

$$= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right] \tag{3.3}$$

$$\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right]. \tag{3.4}$$

The first three equalities phrase the log-probability as an expectation with respect to the variational distribution, and the fourth line is a direct application of Jensen's inequality. The goal is to tighten this bound on $\log p(\mathbf{x})$ by maximizing it with respect to both $\boldsymbol{\theta}$ and the density function q. Importantly we no longer have to contend with the integral in $p(\mathbf{x})$. The general problem of optimizing with respect to a function is the domain of the calculus of variations, and this is why we refer to this approach as a variational method. In the next section, we demonstrate how maximizing this lower bound is also a form of posterior inference.

3.1.1 KL-Divergence

While the lower bound in 3.4 may seem somewhat arbitrary, it can be motivated using the following reformulation:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}) p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \right] = \log p(\mathbf{x}) + \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \right] \tag{3.5}$$

$$\text{where} \quad \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \right] =: -KL[q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x})]. \tag{3.6}$$

Here $KL(q||p)$ is known as the KL-divergence [Kullback and Leibler, 1951], and has its roots in information theory. It is a measure of the proximity between distributions q and p. Jensen's inequality can be used to show that it is nonnegative:

$$KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})] = -\int q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} \tag{3.7}$$

$$\geq -\log \int q(\mathbf{z}|\mathbf{x}) \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} \tag{3.8}$$

$$= -\log \int p(\mathbf{z}|\mathbf{x}) d\mathbf{z} = 0. \tag{3.9}$$

This nonnegativity in combination with the lower bound reformulation in 3.5 implies that a maximum with respect to q is attained when $KL(q||p) = 0$. Moreover, equation 3.7 tells us that the KL-divergence is zero precisely when $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$. Therefore, the problem of maximizing the lower bound is equivalent to minimizing the KL-divergence, and in minimizing $KL(q||p)$ we are making a better and better approximation $q(\mathbf{z}|\mathbf{x})$ to the posterior $p(\mathbf{z}|\mathbf{x})$. Optimizing the lower bound is a means of learning *and* inference.

Note that $KL(q||p) \neq KL(p||q)$, and variational methods exist which make use of the latter, e.g. Minka [2001]. We prefer the former because it involves taking expectations with respect to q as opposed to the intractable posterior p.

3.1.2 Amortized Inference

The lower bound can be easily extended to the dataset $\mathcal{D}$:

$$\log p(\mathcal{D}) = \sum_{i=1}^{n} \log p(\mathbf{x}_i) \geq \sum_{i=1}^{n} \mathbb{E}_{q_i(\mathbf{z}_i|\mathbf{x}_i)} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{z}_i)}{q_i(\mathbf{z}_i|\mathbf{x}_i)} \right].$$

In this case, we maximize the lower bound in turn with respect to $\boldsymbol{\theta}$ and each of the q_i. However one may argue that a single variational approximation is preferred to separate posterior estimates for each of the $\mathbf{z}_i$. This can be done by parameterizing the variational distribution as $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, where $\boldsymbol{\phi}$ are the *variational parameters*:

$$\log p(\mathcal{D}) \geq \sum_{i=1}^{n} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{z}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} \right] =: \mathcal{L}[\boldsymbol{\theta}, \boldsymbol{\phi}]. \tag{3.10}$$

The lower bound $\mathcal{L}$ is known as the *evidence lower bound* (ELBO) or the (negative) *free energy*, and is the one we use in this thesis. All that is learned about the posterior is encoded into a single set of parameters ϕ and we can easily infer the latent distribution for new observations. This is known as *amortized inference* and improves the computational efficiency of the learning process [Gershman and Goodman, 2014, Rezende et al., 2014a]. The parametric family $q_\phi(\mathbf{z}|\mathbf{x})$ needs to be sufficiently flexible to obtain an accurate estimate of the true posterior. We use the same strategy as we did for the generative model, i.e. we consider distributions of the form $q(\mathbf{z}|f_\phi(\mathbf{x}))$ where f is a nonlinear function of $\mathbf{x}$, parameterized by ϕ.

We have turned the integration problem into one of optimization. To make this scalable we use a stochastic numerical optimization method which we outline in the next section. We require that the lower bound is differentiable with respect to the model and variational parameters, which in turn means that the distributions $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ must also be differentiable in these parameters.

3.2 Numerical Optimization

In this section, we describe the optimization method used in this thesis. The variational lower bound of equation 3.10 is a scalar-valued function of the form

$$F(\mathbf{w}; \mathcal{D}) \;=\; \frac{1}{n} \sum_{i=1}^{n} F(\mathbf{w}; \mathbf{x}_i) \tag{3.11}$$

for a dataset $\mathcal{D}$ of n observations, and parameters $\mathbf{w} \in \mathbb{R}^p$. Our goal is to maximize F with respect to $\mathbf{w}$. We approach this problem using gradient-based methods where the idea is to incrementally update the optimization variable $\mathbf{w}$ so as to slowly move the *objective function* F towards its maximum. If F is differentiable at $\mathbf{w}$, then a basic result of calculus is that the gradient vector $\nabla_{\mathbf{w}} F(\mathbf{w}; \mathcal{D})$ points in the direction of steepest ascent. This is at the core of our strategy and we require that F is differentiable almost everywhere. If there is an upper bound on the curvature of the function (technically known as Lipschitz continuity), then a sufficiently small step in the gradient direction will increase F. And so, we iteratively update the optimization variable:

$$\mathbf{w}^{(j+1)} \;=\; \mathbf{w}^{(j)} + \alpha \, \nabla_{\mathbf{w}} F(\mathbf{w}^{(j)}; \mathcal{D}), \tag{3.12}$$

where $\alpha > 0$ is the *step size* or *learning rate*. As long as α is sufficiently small, we will experience an increase in F with each step, arriving at a maximum when $\nabla F = \mathbf{0}$. This process is known as *gradient ascent*. The objective function is almost always a complicated function with many peaks and valleys, and the method will for all practical purposes be searching for a *local* maximum, i.e. a value $\mathbf{w}^*$ that is optimal within a local neighbourhood. Moreover the point where $\nabla F = \mathbf{0}$ may not always be a maximum and may instead be a saddle point where there is an upward slope in one dimension and downward slope in another.

3.2.1 Stochastic Gradients

Suppose we instead update $\mathbf{w}$ using a single datapoint $\mathbf{x}$ sampled from $\mathcal{D}$:

$$\mathbf{w}^{(j+1)} \;=\; \mathbf{w}^{(j)} + \alpha \, \nabla_{\mathbf{w}} F(\mathbf{w}^{(j)}; \mathbf{x}). \tag{3.13}$$

An update of this form does not depend on the size of the dataset and is much more scalable. We are in effect using the gradient computed from a single datapoint as an estimate for the true gradient, and this will produce updates with a large variance. This means that the updates may not always increase F and convergence is much more difficult. Nevertheless if we decrease the step size α sufficiently with each iteration, convergence is guaranteed. The advantage of the large variance is that there is always a chance of escaping neighbourhoods and moving to new regions with better local maxima.

Instead of sampling a single datapoint from $\mathcal{D}$, we can sample $b < n$ datapoints forming a *minibatch* $\mathcal{B}$. This results in an update of the form:

$$\mathbf{w}^{(j+1)} \;=\; \mathbf{w}^{(j)} + \alpha \, \frac{1}{b} \sum_{i \in \mathcal{B}} \nabla_{\mathbf{w}} F(\mathbf{w}^{(j)}; \mathbf{x}_i). \tag{3.14}$$

The increase from a single sample to b samples improves the stability of updates and yet makes it possible to escape poor local maxima. This form of update is known as *stochastic* gradient ascent and is the one we make use of.

3.2.2 Momentum

Gradient updates may sometimes wildly oscillate and even diverge if the step size is too large. On the other hand, they may take a long time to converge if the step size

is too small. We can accelerate convergence and mitigate oscillations by introducing *momentum* into the updates. We denote the gradient in equation 3.14 as $\nabla F_{\mathcal{B}}(\mathbf{w}) := \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla_{\mathbf{w}} F(\mathbf{w}; \mathbf{x}_i)$ for brevity. The momentum update is:

$$\boldsymbol{\nu}^{(j+1)} = \eta\boldsymbol{\nu}^{(j)} + \alpha\, \nabla F_{\mathcal{B}}(\mathbf{w}^{(j)}) \tag{3.15}$$

$$\mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} + \boldsymbol{\nu}^{(j+1)}. \tag{3.16}$$

Here we have an exponential moving average of gradient updates which reduces movement in directions of gradient inconsistency across updates, and increases movement in directions for which the gradients agree. The scalar η is the friction constant and is usually set to 0.9. An improvement to this update proposed by Nesterov [2005] is:

$$\boldsymbol{\nu}^{(j+1)} = \eta\boldsymbol{\nu}^{(j)} + \alpha\, \nabla F_{\mathcal{B}}(\mathbf{w}^{(j)} + \eta\boldsymbol{\nu}^{(j)}) \tag{3.17}$$

$$\mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} + \boldsymbol{\nu}^{(j+1)}. \tag{3.18}$$

The gradient is computed not at the current parameter position but at an estimated future position. This allows the algorithm to anticipate when the momentum is too large and reduce it.

3.2.3 Adaptive Learning Rates

For increased learning flexibility, we incorporate a unique step size for each element of the parameter vector $\mathbf{w}$. At a high level, the idea is to track an exponential moving average of squared gradient computations from previous updates. The learning rate is set in inverse proportion to the square root of these averages. The effect of this is that frequently updated elements are updated with a smaller learning rate, and vice versa for less frequently updated elements of $\mathbf{w}$. We use an algorithm called *RMSProp* [Tieleman and Hinton, 2012] to do this which has improved upon previous adaptive methods [Duchi et al., 2011]. We use RMSProp in combination with Nesterov's update in equations 3.17 and 3.18.

3.3 Computing the Gradients

In order to optimize the lower bound $\mathcal{L}$, we need to compute the gradients with respect to both the model parameters $\boldsymbol{\theta}$ and variational parameters $\boldsymbol{\phi}$. Thus, we require

that the distributions $p(\mathbf{x}, \mathbf{z})$ and $q(\mathbf{z}|\mathbf{x})$ are differentiable for both sets of parameters. Unfortunately the gradients cannot be computed directly because the expectations in 3.10 cannot be evaluated exactly. Instead we resort to a Monte Carlo approximation, which for a single datapoint $\mathbf{x}$ is:

$$\mathcal{L}[\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}] \;=\; \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right] \;\approx\; \frac{1}{S}\sum_{j=1}^{S} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}^{(j)})}{q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)}|\mathbf{x})}. \tag{3.19}$$

Here each $\mathbf{z}^{(j)}$ is a sample from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ for a total of S samples. Although we use a Monte Carlo approach to compute the gradients, we do it in a computationally efficient manner since we are performing these operations in minibatches. Moreover the stochastic nature of our optimization procedure means that a small sample size is sufficient; we set $S = 1$ for all our experiments. It is straightforward to compute the gradients with respect to $\boldsymbol{\theta}$ for the estimate in 3.19, but we run into difficulties for $\boldsymbol{\phi}$. The issue is that the expectation is over the variational distribution which is itself parameterized by $\boldsymbol{\phi}$.

3.3.1 Reparameterization Trick

The problem we are confronted with can be generalized to the problem of differentiating $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f(\mathbf{z})]$ with respect to $\boldsymbol{\phi}$, for some function f. It can be resolved using a simple but powerful method known as the *reparameterization trick* [Kingma and Welling, 2013], where the strategy is to divorce the stochastic nature of $\mathbf{z}$ from the parameters $\boldsymbol{\phi}$. The latent variable $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is defined using a transformation $\mathbf{z} = g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon})$ where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. This allows for the following Monte Carlo estimate:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f(\mathbf{z})] \;=\; \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon})] \;\approx\; \frac{1}{S}\sum_{j=1}^{S} f(g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon}^{(j)})), \tag{3.20}$$

where each $\boldsymbol{\epsilon}^{(j)}$ is a sample from $p(\boldsymbol{\epsilon})$. As long as g is differentiable, we can take derivatives with respect to $\boldsymbol{\phi}$ using this estimate. For example, if $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$, then we can use the *location-scale* transform, $\mathbf{z} = g(\boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that $\odot$ denotes element-wise multiplication. Here $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are functions parameterized by $\boldsymbol{\phi}$, and we have in effect isolated the stochasticity in $\mathbf{z}$ from the parameters $\boldsymbol{\phi}$. Many other approaches also exist for constructing such a transformation [Kingma

and Welling, 2013]. And so, the Monte Carlo estimate in 3.19 is replaced with:

$$\mathcal{L}[\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}] \;=\; \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right] \;\approx\; \frac{1}{S}\sum_{j=1}^{S} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}^{(j)})}{q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)}|\mathbf{x})} \tag{3.21}$$

$$\text{where} \quad \mathbf{z}^{(j)} = g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon}^{(j)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(j)} \sim p(\boldsymbol{\epsilon}). \tag{3.22}$$

Note that this estimate can only be used for continuous $\mathbf{z}$ because of the differentiability condition. The idea of reparameterization has also been proposed by Challis and Barber [2012a] and Rezende et al. [2014a].

Worth mentioning is an alternative approach for solving this problem known as the *score function estimator*:

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f(\mathbf{z})] \;=\; \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f(\mathbf{z}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z})] \;\approx\; \frac{1}{S}\sum_{j=1}^{S} f(\mathbf{z}^{(j)}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)}), \tag{3.23}$$

where $\mathbf{z}^{(j)} \sim q_{\boldsymbol{\phi}}(\mathbf{z})$. We derive this result in section A.2. It works for both continuous and discrete $\mathbf{z}$, but the corresponding gradient estimator exhibits a much higher variance than the reparameterization approach [Kingma and Welling, 2013]. There are methods which attempt to reduce this variance [Mnih and Gregor, 2014, Paisley et al., 2012], and a general framework is also available for seamlessly computing deterministic *and* stochastic derivatives, making it possible to use both the log-derivative and reparameterization tricks in the same model [Schulman et al., 2015]. Since we are modeling a continuous latent space, we solely rely on the reparameterization trick.

Summary

In this chapter, we demonstrated how inference and learning is possible for the model of 2.6. We derived and parameterized a lower bound on $\log p(\mathbf{x})$ such that learning and inference are both achieved by maximizing this bound. The gradient-based techniques for solving this problem were outlined and finally, we described the reparameterization trick which allows us to compute gradients of the lower bound.

4. Multimodal Learning

In this chapter, we outline our approach to multimodal learning. A generative model for a single input is described, and this thinking is extended to paired variables. We then outline a variation of the generative model as well as a modification of the lower bound objective function. The chapter concludes by describing the evaluation methods for these models.

4.1 A Model for Images

In previous chapters, we described a general-purpose generative model for which large-scale inference and learning is possible using variational methods. Here we introduce a specification of this model for images, which we represent as multidimensional arrays. Black and white images are rectangular grids of binary pixels. Coloured images are three-dimensional arrays where the third dimension indicates the colour channel: red, green, or blue. Each element of a coloured image is an integer in $\{0, 1, \ldots, 255\}$, also referred to as 8-bit colour depth. We use the term *pixel* for a single element of such multidimensional arrays. In some cases, it may be useful to think of an image in vectorized form.

Consider the generative process of figure 2.1 for an independent and identically distributed dataset of images $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Variable $\mathbf{z} \in \mathbb{R}^k$ is drawn from prior $p_{\boldsymbol{\theta}}(\mathbf{z})$, and subsequently $\mathbf{x}$ is generated from the conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. We interpret the latent $\mathbf{z}$ to be the explanatory factors of variation underlying the observed data, and it is natural to think of these factors as mutually independent. Therefore, we choose a prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ which can be factorized. The use of a Gaussian simplifies the lower bound as we demonstrate later, but any other differentiable distribution is permissible.

In a similar spirit, we assume that the elements of the image $\mathbf{x}$ are independent of each other when conditioned on the latent representation $\mathbf{z}$. A natural model for black and white images is the multivariate Bernoulli distribution. For a (vectorized) image $\mathbf{x} \in \{0,1\}^m$, this is:

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{m} \beta_i^{x_i}(1-\beta_i)^{(1-x_i)}, \tag{4.1}$$

where each $\beta_i \in [0,1]$ is the probability that pixel $x_i = 1$. Here $\boldsymbol{\beta}$ is the output of a differentiable function $f_{\boldsymbol{\theta}}(\mathbf{z})$ parameterized by $\boldsymbol{\theta}$. A natural model for a coloured image $\mathbf{x} \in \{0,\ldots,255\}^m$ is the multivariate categorical distribution:

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{m}\prod_{j=0}^{255} \alpha_{ij}^{\mathbb{I}[x_i=j]} \quad \text{where} \quad \mathbb{I}[x_i=j] = \begin{cases} 1 & \text{if} \quad x_i = j \\ 0 & \text{otherwise} \end{cases}. \tag{4.2}$$

Each α_{ij} is the probability that $x_i = j$, and $\sum_{j=0}^{255} \alpha_{ij} = 1$ for $i = 1,\ldots,m$. This is an extension of the Bernoulli distribution to the case where each pixel has 256 possible values. We parameterize this distribution in a similar way, i.e. the $\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_m$ are outputs of function $f_{\boldsymbol{\theta}}(\mathbf{z})$.

4.2 Variational Approximation, Inference, and Learning

We can make the function $f_{\boldsymbol{\theta}}(\mathbf{z})$ highly nonlinear which in turn makes our model very flexible (see chapter 5). Even though the conditional $p(\mathbf{x}|\mathbf{z})$ is a simple, fully-factorized distribution, the marginal probability of the image $p(\mathbf{x})$ is a much more complex distribution, more than able to capture the richness of natural images. By the same token, the posterior $p(\mathbf{z}|\mathbf{x})$ is intractable and we introduce a variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$. Here $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are outputs of a differentiable function $f_{\boldsymbol{\phi}}(\mathbf{x})$ with parameters $\boldsymbol{\phi}$. The diagonal covariance for q is chosen as a simplification, but it is possible to replace this with the full covariance matrix (e.g. Rezende et al. [2014b]).

The inference and learning problems are solved by maximizing the variational lower bound (3.10) with respect to parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. This lower bound is refor-

mulated (for a single datapoint $\mathbf{x}$) to better suit our purposes:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right] \tag{4.3}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - KL[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]. \tag{4.4}$$

This formulation involves the KL-divergence between the variational distribution and the prior probability. The reparameterization trick (section 3.3) can be used to compute gradients for the first term in 4.4. The corresponding Monte Carlo estimate will use the following location-scale transform:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.5}$$

As we outlined in section 3.3, the use of this transform separates the stochasticity in $\mathbf{z}$ from the variational parameters $\boldsymbol{\phi}$, which in turn results in a differentiable estimate.

The KL term in 4.4 can be resolved analytically:

$$-KL[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] = \frac{1}{2}\sum_{j=1}^{k}\Big(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\Big). \tag{4.6}$$

This can be justified using properties of Gaussians [Kingma and Welling, 2013]. Thus, the lower bound estimate for a single datapoint $\mathbf{x}$ becomes:

$$\mathcal{L}_{\mathbf{x}} := \frac{1}{S}\sum_{s=1}^{S}\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(s)}) + \frac{1}{2}\sum_{j=1}^{k}\Big(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\Big), \tag{4.7}$$

where each sample $\mathbf{z}^{(s)}$ is defined as in 4.5. We find that setting the sample size $S = 1$ works well for our experiments. We now have a complete specification of model, learning, and inference for a single image modality.

4.3 Multiple Modalities

We can extend our model to a set of paired images $\mathcal{D}_{\mathbf{x},\mathbf{y}} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ which we think of as two simultaneously observed information sources. Again we would like to learn a shared representation $\mathbf{z} \in \mathbb{R}^k$ across the two image modalities.

The latent prior is $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ as before, but now both $\mathbf{x}$ and $\mathbf{y}$ are generated from the conditional distribution, $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}|\mathbf{z})$. Since $\mathbf{z}$ is meant to be a shared representation of the two inputs, we assume that knowing $\mathbf{z}$ makes $\mathbf{x}$ and $\mathbf{y}$ independent

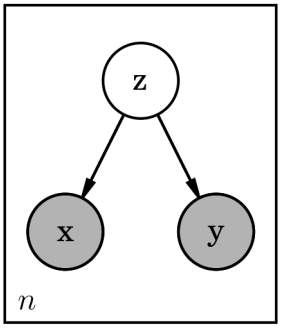


Figure 4.1: Generative model with two inputs.

of each other. That is to say $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}|\mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{z})$. We model each conditional as either a multivariate Bernoulli (4.1) or a multivariate categorical distribution (4.2), and we parameterize each conditional in the same way as section 4.1. The generative process is depicted in figure 4.1.

A lower bound can be derived using the same principles as in section 3.1. For a single pair $(\mathbf{x}, \mathbf{y})$, this is:

$$\log p(\mathbf{x}, \mathbf{y}) \;\geq\; \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{z})] - KL[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{y})||p(\mathbf{z})]. \tag{4.8}$$

This is justified in section A.1. The variational distribution is again Gaussian, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are outputs of a differentiable function $f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{y})$ parameterized by $\boldsymbol{\phi}$.

In the same manner as before, we use the reparameterization trick for the first term in 4.8 and we analytically derived the KL term. This leads to the following estimate:

$$\mathcal{L}_{\mathbf{x},\mathbf{y}} \;:=\; \sum_{s=1}^{S} \Big(\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(s)}) + \log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{z}^{(s)}) \Big) \;+\; \frac{1}{2} \sum_{j=1}^{k} \Big(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2 \Big), \tag{4.9}$$

$$\text{where} \quad \mathbf{z}^{(s)} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}^{(s)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(s)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.10}$$

Again we use a sample size of $S = 1$ in our experiments.

4.4 Semi-Supervised Learning

We can naturally extend the multimodal model to the semi-supervised setting where we only have a few paired examples $\mathcal{D}_{\mathbf{x},\mathbf{y}}$, and much larger sets of unpaired data $\mathcal{D}_{\mathbf{x}} =$

$\{\mathbf{x}_1, \ldots, \mathbf{x}_{n_x}\}$ and $\mathcal{D}_\mathbf{y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_{n_y}\}$. In this scenario, we simultaneously optimize the lower bound on $\log p(\mathbf{x})$, $\log p(\mathbf{y})$, and $\log p(\mathbf{x}, \mathbf{y})$. We compute gradients using the following estimate:

$$\mathcal{L} := \frac{1}{b_{x,y}} \sum_{i \in \mathcal{B}_{\mathbf{x},\mathbf{y}}} \mathcal{L}^{(i)}_{\mathbf{x},\mathbf{y}} + \frac{1}{b_x} \sum_{i \in \mathcal{B}_\mathbf{x}} \mathcal{L}^{(i)}_\mathbf{x} + \frac{1}{b_y} \sum_{i \in \mathcal{B}_\mathbf{y}} \mathcal{L}^{(i)}_\mathbf{y}, \tag{4.11}$$

which is simply a combination of the previously defined lower bounds. Here $\mathcal{L}_\mathbf{y}$ is derived for $\mathbf{y}$ in the same way as $\mathcal{L}_\mathbf{x}$ was for $\mathbf{x}$ in 4.7. The sets $\mathcal{B}_{\mathbf{x},\mathbf{y}}$, $\mathcal{B}_\mathbf{x}$, and $\mathcal{B}_\mathbf{y}$ index minibatches from the datasets $\mathcal{D}_{\mathbf{x},\mathbf{y}}$, $\mathcal{D}_\mathbf{x}$, and $\mathcal{D}_\mathbf{y}$ respectively. The values $b_{x,y}$, b_x, and b_y are the corresponding minibatch sizes. Having three independent minibatch sizes allows this approach to leverage large amounts of unpaired data when only a few paired examples are available.

We now have three variational distributions $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$, $q_\phi(\mathbf{z}|\mathbf{x})$, and $q_\phi(\mathbf{z}|\mathbf{y})$ corresponding to each lower bound in 4.11. In learning the model and variational distributions, we are now able to translate between the modalities. From an observed $\mathbf{x}$, we can sample a possible latent representation $\mathbf{z}$ using $q(\mathbf{z}|\mathbf{x})$, and this latent representation can be used to generate a plausible $\mathbf{y}$ from the model distribution $p(\mathbf{y}|\mathbf{z})$. Similarly, we can reconstruct a plausible $\mathbf{x}$ from an initially observed $\mathbf{y}$.

Finally, note that the probability $p(\mathbf{x}, \mathbf{y})$ will tend to be orders of magnitude smaller than the marginals $p(\mathbf{x})$ or $p(\mathbf{y})$, because probability mass is distributed over a much greater region. This disparity also manifests itself in the objective function of equation 4.11, where the bounds $\mathcal{L}_\mathbf{x}$ and $\mathcal{L}_\mathbf{y}$ will be of a different magnitude than the shared bound $\mathcal{L}_{\mathbf{x},\mathbf{y}}$. This will negatively impact the process of learning a shared representation of the two modalities. To counteract this imbalance, we introduce an additional parameter λ into the objective to encourage optimization of the joint bound $\mathcal{L}_{\mathbf{x},\mathbf{y}}$:

$$\tilde{\mathcal{L}} := \frac{\lambda}{b_{x,y}} \sum_{i \in \mathcal{B}_{\mathbf{x},\mathbf{y}}} \mathcal{L}^{(i)}_{\mathbf{x},\mathbf{y}} + \frac{1}{b_x} \sum_{i \in \mathcal{B}_\mathbf{x}} \mathcal{L}^{(i)}_\mathbf{x} + \frac{1}{b_y} \sum_{i \in \mathcal{B}_\mathbf{y}} \mathcal{L}^{(i)}_\mathbf{y}. \tag{4.12}$$

We find that λ in the set of values $\{0.3, 0.5\}$ results in a significant improvement in the optimization process when compared to the bound in 4.11.

4.5 Scaling to Multiple Modalities

Notice that in the bound of 4.12, we require 3 variational distributions. Unfortunately, the number of variational distributions we must keep track of grows exponentially

as we increase the number of modalities. We can resolve this by conditioning the multimodal variational approximation $q(\mathbf{z}|\mathbf{x}, \mathbf{y})$ on just one of the two modalities. For example, we may assume that $q(\mathbf{z}|\mathbf{x}, \mathbf{y}) = q(\mathbf{z}|\mathbf{x})$. This results in the following lower bound on the likelihood:

$$\log p(\mathbf{x}, \mathbf{y}) \;\geq\; \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{y}|\mathbf{z})] + \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right]. \tag{4.13}$$

We derive this in section A.3. Similarly, if we instead assume that $q(\mathbf{z}|\mathbf{x}, \mathbf{y}) = q(\mathbf{z}|\mathbf{y})$, then we get the following lower bound:

$$\log p(\mathbf{x}, \mathbf{y}) \;\geq\; \mathbb{E}_{q(\mathbf{z}|\mathbf{y})}[\log p(\mathbf{x}|\mathbf{z})] + \mathbb{E}_{q(\mathbf{z}|\mathbf{y})}\left[\log \frac{p(\mathbf{y}, \mathbf{z})}{q(\mathbf{z}|\mathbf{y})}\right]. \tag{4.14}$$

We can derive an estimate similar to the one in equation 4.9 using either of these lower bounds. The advantage here is that we do not require a new variational distribution for every combination of modalities we are modeling, making this a scalable alternative.

Notice that the first term in 4.13 is designed to improve translation performance from $\mathbf{x}$ to $\mathbf{y}$ when optimized. Similarly, the first term in 4.14 is designed to optimize translation from $\mathbf{y}$ to $\mathbf{x}$. Therefore, we can construct an objective function which potentially encourages bidirectional translation by taking the average of both these bounds. In our experiments, we use an estimate of this average bound, which can easily be derived in the same way as 4.9. Note however that because of our simplifying assumptions, these bounds will never do better than the original lower bound in equation 4.8 in terms of maximizing the likelihood.

4.6 An Autoregressive Image Model

An alternative to the fully-factorized image model of section 4.1 is an *autoregressive* model, which has been shown to be a powerful generative model [Larochelle and Murray, 2011, van den Oord et al., 2016a]. For an image $\mathbf{x}$, it is defined as:

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \;=\; \prod_{i=1}^{m} p_i(x_i|x_{<i}, \mathbf{z}), \tag{4.15}$$

where $x_{<i}$ represents the elements of $\mathbf{x}$ which occur before x_i in some prespecified ordering. For our purposes, this ordering will be a row-by-row and pixel-by-pixel

ordering of the elements of $\mathbf{x}$. Here each $p_i(x_i|x_{<i}, \mathbf{z})$ is either a univariate Bernoulli or categorical distribution for black-and-white or colour images respectively.

The autoregressive model can be powerful enough to explain all the structure in $\mathbf{x}$ without relying on the latent variable $\mathbf{z}$ [Bowman et al., 2016, Fabius et al., 2014]. Consider the lower bound formulation in the single modality case: $\mathbb{E}[\log p(\mathbf{x}|\mathbf{z})] - KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]$. If there is no inherent dependency between the observations and the latents, then the KL-divergence will be zero because the posterior (and approximate posterior q) is equal to the prior. To encourage the model to use the latent representation, we introduce a constant ψ into the training objective:

$$\mathbb{E}[\log p(\mathbf{x}|\mathbf{z})] - \sum_{j=1}^{k} \max\Big(\psi,\ KL[q(z_j|\mathbf{x})||p(z_j)]\Big), \tag{4.16}$$

where we can factorize $q(\mathbf{z}|\mathbf{x})$ because it is a Gaussian with diagonal covariance (from our assumption). The constant ψ prevents the iterative optimization scheme from reducing the KL-divergence to zero too early in the learning, and this in turn encourages the usage of the latent space in the generative process. We can of course also extend this to the multimodal case. The approach was proposed by Kingma et al. [2016] and is known as the *freebits penalty*. We find the authors' suggested settings of $\psi \in \{0.125, 0.25\}$ to work well in our experiments.

4.7 Evaluation

Learning in these models proceeds by maximizing a lower bound on $\log p(\mathbf{x}, \mathbf{y})$. In order to ensure that this generalizes to new observations, the data is split into a *training* and *test* set. The optimization procedure takes place solely using the training data, and the lower bound is evaluated on both the training and test sets. In general, the training and test bounds will keep increasing during the iterative optimization process, but it is likely that after a certain point, the test bound will stop increasing and even decrease. It is after this point that the model is losing its generalizability to unseen data, and this is known as *overfitting*. In order to prevent overfitting, the training and test bounds are tracked throughout the iterative optimization scheme, and the process is stopped when the test bound begins to diverge from the training bound. This is known as *early stopping*. Note that in order to compare various model configurations, we make a third split of our data called the *validation* set. The lower bound on this validation

set is used to select the best model configurations. For these chosen models, we report the test lower bound in our results.

Our models are compared using the log-likelihood lower bound, and learned representations are sometimes evaluated by their performance on downstream tasks. We also evaluate the models visually by generating synthetic samples. The quality of these samples and the log-likelihood lower bound are not correlated in terms of their performance [Theis et al., 2015], and we use both approaches for a more complete evaluation of our models.

Summary

In this chapter, we introduced several generative models for image modalities and derived the variational lower bound for inference and learning. We also introduced a variant of the lower bound which scales easily to a large number of modalities.

5. Neural Networks

Much of the discussion in previous chapters alluded to the use of highly nonlinear and flexible functions to parameterize a model or variational distribution. In this chapter, we elaborate on the nature of these functional forms using the language of neural networks.

5.1 Deep Neural Networks

For our purposes, a deep neural network is a mapping $f : \mathcal{U} \to \mathcal{V}$ built using compositions of linear transforms and nonlinear functions. Here $\mathcal{U}$ and $\mathcal{V}$ are subsets of the real vector space, with a varying number of dimensions. Given that these networks are built using compositions of simpler functions, they provide an inherently modular framework for representing relationships of immense complexity. The *depth* of the network is usually measured by the number of nonlinear functions, and with each additional nonlinearity, the network is able to better capture higher-order and more abstract statistical properties of the input. This makes deep neural networks ideal for modeling complex, structured data such as images, audio, or video. A few examples of this are object recognition [Krizhevsky et al., 2012], acoustic modeling [Mohamed et al., 2012], and reinforcement learning [Mnih et al., 2013].

5.2 Backpropagation

Neural networks are parameterized in order to make learning and inference possible. When embedded inside the lower bound of equation 4.9, a network's parameters are optimized so as to maximize this bound. In this process, the mapping f becomes a better and better approximation to the true relationship between an image $\mathbf{x}$ and its

latent representation $\mathbf{z}$ in the model $p(\mathbf{x}|f_{\boldsymbol{\theta}}(\mathbf{z}))$, for example. A key requirement is that the network is differentiable almost everywhere with respect to its parameters, as this enables the use of the scalable optimization methods we outlined in section 3.2.

Derivatives are computed using the *backpropagation* algorithm [Werbos, 1994], which is an application of the chain rule of calculus. The crucial insight here is that these network derivatives are computed in an order which avoids redundant computation, allowing for a dramatic improvement in efficiency. Backpropagation is a subset of automatic differentiation [Baydin et al., 2015, Griewank et al., 1989], a more general framework for evaluating derivatives which even allows control-flow layers in a neural network, e.g. conditional statements and for-loops.

5.3 Layers

We begin by describing the neural network layers used in this thesis. We denote the input and output of each layer as $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^k$ respectively.

5.3.1 Linear Layer

The linear layer is simply an affine transform between input and output:

$$\mathbf{v} = \mathbf{W}\mathbf{u} + \mathbf{b}.$$

Here $\mathbf{W} \in \mathbb{R}^{k \times m}$ and $\mathbf{b} \in \mathbb{R}^k$ are the learnable parameters. This layer is *fully-connected* because the k elements or *units* of output $\mathbf{v}$ each depend on every single unit of input $\mathbf{u}$ via the matrix $\mathbf{W}$.

5.3.2 Convolution

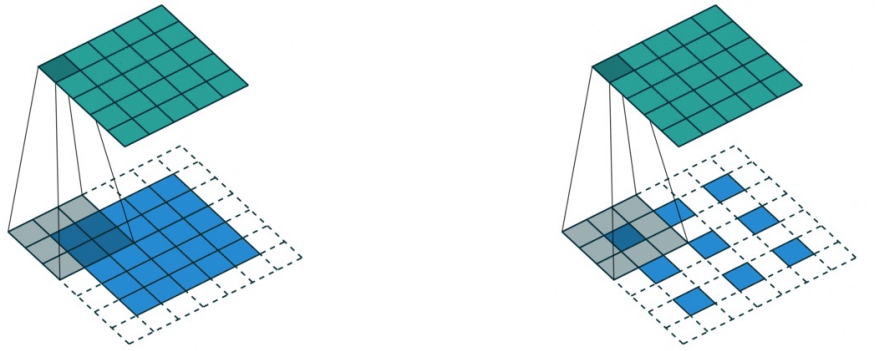
Convolution layers have been fundamental to modeling images with neural networks [Fukushima, 1980, LeCun, 1998] and they hinge on two important priors. The first is that objects in natural scenes tend to exist in a *local* space, i.e. they are not segmented into discontinuous regions across the image. The second prior is *translation invariance* which means that the appearance of an object is independent of its location in the image. Generally speaking, an object is recognizable regardless of whether it is located in the top-left corner or the center of an image.

Like the linear layer, the convolution layer is an affine transform, but it operates on multidimensional arrays. Here we restrict our discussion to the two-dimensional case. The convolution mechanism is much better explained in terms of connectivity. Recall that a linear layer is fully-connected because every input unit is connected to every output unit. In a convolution layer, each output unit is connected to a subset of input units known as the *receptive field*. For an $h \times w$ input image $\mathbf{u}$, each output unit is connected to a $k \times k$ patch of the input. This type of connectivity enacts the locality prior for natural scenes. The convolution operator works by moving the receptive field in increments of 1 across the rows and columns of the image $\mathbf{u}$, each time mapping to a corresponding output unit. Therefore, the output $\mathbf{v}$ will share the two-dimensional structure of the input. Figure 5.1a visualizes the convolution operator.

Formally, the i-th output unit is defined as $v_i = b_i + \sum_{j \in R} W_j u_j$ where R indexes a $k \times k$ patch of input units. The parameter vector $\mathbf{W} \in \mathbb{R}^{k^2}$ is known as the convolution *kernel*, and we generally think of it as a two-dimensional parameter in $\mathbb{R}^{k \times k}$. Note that this kernel is *shared* across all output units. This enacts the translation invariance property because the same parameter weights are applied regardless of the global position of the $k \times k$ patch. In the case of an $h \times w \times c$ input image, where the third-dimension is for colour, the kernel becomes $\mathbf{W} \in \mathbb{R}^{k \times k \times c}$ and now operates on $k \times k \times c$ patches of the image, although the receptive field only moves along the spatial dimensions.

Note that any receptive field larger than 1×1 will result in the output $\mathbf{v}$ having a smaller dimensionality than the input $\mathbf{u}$. To counteract this, the input image $\mathbf{u}$ is usually *padded* with enough zeros on each side to ensure the dimensionality remains the same. This is the approach we take with our networks. Moreover, in most applications, using a single kernel is restrictive, and we generally utilize n kernels $\{\mathbf{W}_1, \ldots, \mathbf{W}_n\}$ which correspond to n independent transformations from $\mathbf{u}$ to $\mathbf{v}$. The output $\mathbf{v}$ now has dimensionality $h \times w \times n$. The output from a single kernel is referred to as a *feature map*, and in this case, $\mathbf{v}$ consists of n feature maps.

Since the dimensionality k of the receptive field is usually much smaller than the input image, storage of the parameters $\mathbf{W}$ is much cheaper. On the other hand, the convolution layer is computationally more expensive than the linear layer.



(a) Convolution **(b)** Deconvolution with stride $\frac{1}{2}$

Figure 5.1: Visualization of convolution and deconvolution operations with a 3×3 receptive field, taken from `https://github.com/vdumoulin/conv_arithmetic`. Input **u** is blue; output **v** is green; white squares indicate zero-padding.

5.3.3 Strided Convolution

In some cases, we may want an output with reduced dimensionality, e.g. when transforming an observed image to a lower-dimensional latent representation. We can do this using strided convolutions. Here the receptive field moves in increments or *strides* of 2 across the rows and columns of the input, resulting in an output which is half its size in both spatial dimensions.

5.3.4 Deconvolution

Sometimes, we may want to reverse the effect and get an output with increased dimensionality. This is done using a deconvolution layer [Zeiler et al., 2010], also known as a transposed convolution. The idea is to zero-pad the input as shown in figure 5.1b, and this has the effect of upsampling the image. We can think of this operator as having a receptive field with a *fractional* stride. For example, a deconvolution with a fractional stride of $\frac{1}{2}$ will double the size of the input image (across spatial dimensions).

5.3.5 Activations

All layers defined up to this point are variations of linear transforms. Networks consisting solely of such transforms can be mathematically condensed into a single linear transform, and this is simply not sufficient to describe most real-world phenomena.

Activation functions are a crucial component because they introduce nonlinearity into the network, which enables much more flexible modeling. Many activation functions have been proposed in the literature and we briefly describe the ones used in this work. Note that these activations preserve dimensionality between input and output.

The *sigmoid function* is defined as:

$$\mathbf{v} = \frac{1}{1+\exp(-\mathbf{u})}.$$

It squashes the input $\mathbf{u}$ to be between 0 and 1, and is commonly used when the output $\mathbf{v}$ is to be interpreted as a multivariate Bernoulli probability vector. The *softmax function* is defined as:

$$\mathbf{v} = \frac{\exp(\mathbf{u})}{\mathrm{sum}\big(\exp(\mathbf{u})\big)},$$

where the sum operator denotes a summation of the elements of $\exp(\mathbf{u})$, and the division operator is element-wise. This function squashes the input between 0 and 1 with the added constraint that the sum of elements of $\mathbf{v}$ is 1. Here the output can be interpreted as a categorical probability vector. Both the sigmoid and softmax are typically used as final layers of a neural network.

However these activations are much more difficult to use at intermediary steps because they tend to create regions in the output space where computed gradients are close to zero, meaning that optimization (and learning) is very difficult. This is known as the vanishing gradient problem [Bengio et al., 1994]. The *rectified linear unit* (ReLU) is an activation less prone to this issue [Hochreiter, 1998, Maas et al., 2013]. It is defined as

$$\mathbf{v} = \max(0, \mathbf{u}),$$

where the max operator is element-wise. ReLUs have also demonstrated good empirical performance [Glorot et al., 2011]. *Exponential linear units* (ELU) extend ReLUs and are defined as:

$$v_j = \begin{cases} u_j & \text{if} \quad u_j > 0 \\ \exp(u_j) - 1 & \text{if} \quad u_j \leq 0 \end{cases}$$

for corresponding input and output units. ELUs have been shown to make learning more efficient in deep networks [Clevert et al., 2015], and are heavily used in the intermediary layers of all our networks.

5.4 Residual Blocks

The iterative optimization process described in section 3.2 will eventually saturate as it approaches a local maximum of an objective function. But when a very deep network is embedded in this objective, the iterative process saturates and then quickly degrades, meaning we no longer have a maximum [He et al., 2016, Srivastava et al.]. Intuitively we would expect a deep network's performance to be better than or on par with a shallower network, but this is not the case in practice.

Residual blocks are a solution to this degradation problem proposed by He et al. [2016], where the neural network incorporates blocks of layers with shortcut connections between them. Let g be a linear transform and let σ be an activation function. Then, a network may include the following residual block with input $\mathbf{u}$ and output $\mathbf{v}$:

$$\mathbf{v} = g(\sigma(g(\mathbf{u}))) + \mathbf{u}. \tag{5.1}$$

While the input $\mathbf{u}$ is transformed through internal layers, it also directly connects to the output $\mathbf{v}$. To gain an intuition for why this helps, suppose that the true relationship between a network's input and output is simply the identity. Such a mapping can be difficult to learn for a deep network composed of many nonlinearities. However if the network is built using a series of residual blocks with shortcut connections, learning the identity mapping becomes trivial because the layer parameters can be set to zero, thereby allowing the input to propagate through undisturbed. This intuition has been extensively tested by He et al. [2016] and they find that residual blocks alleviate the degradation problem while allowing for much deeper networks.

The residual block in figure 5.2 is used in our networks. When decreasing output dimensionality, the second convolution layer is replaced with a stride-2 convolution. In this case, the shortcut connection is also placed through a stride-2 convolution to keep the dimensionality consistent. We also use deconvolutional residual blocks where the convolutions in figure 5.2 are replaced with deconvolutions, and we use the same (fractional) striding strategy for upsampling. Finally, note that we keep the number of feature maps of each convolution/deconvolution layer consistent through the residual block.

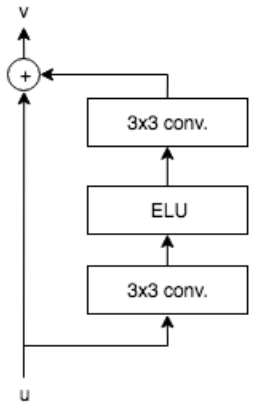


Figure 5.2: Residual block with 3×3 convolutions and an exponential linear unit.

5.5 Architectures

Here we outline the neural network architectures used to parameterize the model and variational distributions of sections 4.1, 4.2, and 4.3.

5.5.1 Inference Networks

The variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is parameterized by a neural network known as the *inference network*, because it maps the observed variable to the latent space. This also includes the additional distribution $q_\phi(\mathbf{z}|\mathbf{y})$ in the multimodal case. We primarily use two kinds of inference networks: fully-connected networks and convolutional networks.

Table 5.1 shows the fully-connected inference architecture. There are two intermediary linear layers and two ELU nonlinearities. Recall that the variational distribution is a Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\sigma}^2\mathbf{I}$. Therefore, the second nonlinearity is separated to form the two outputs $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. The additional third output $\mathbf{h}$ is required for sharing parameters but we defer its explanation to a later section. We introduce another nonlinearity known as the *softplus* which is a smooth version of the ReLU, and this is to ensure that the $\boldsymbol{\sigma}$ output remains positive. The number of outputs of all intermediary linear layers are kept the same, and this is referred to as the number of *hidden* units. The final linear layers have a number of output units equal to the desired dimensionality of the latent space.

The convolutional inference network of table 5.2 instead consists of multiple residual blocks (figure 5.2). Here each residual block is followed by a nonlinearity, and this is repeated multiple times. Some of these residual blocks incorporate striding to reduce dimensionality. We use 3×3 receptive fields in all convolutional layers.

input ($\mathbf{x}$ or $\mathbf{y}$)		
linear layer ELU		
linear layer ELU		
linear $\boldsymbol{\mu}$	linear + softplus $\boldsymbol{\sigma}$	$\mathbf{h}$

Table 5.1: Fully-connected inference network.

input ($\mathbf{x}$ or $\mathbf{y}$)		
residual block + ELU $(\times n)$		
linear layer ELU		
linear $\boldsymbol{\mu}$	linear + softplus $\boldsymbol{\sigma}$	$\mathbf{h}$

Table 5.2: Convolutional inference network.

Parameter Sharing

For the multimodal problem of section 4.3, we are primarily concerned with semi-supervised learning where we have a small number of paired examples and a large number of unpaired examples. An independent third inference network for the multimodal variational distribution $q(\mathbf{z}|\mathbf{x}, \mathbf{y})$ would be difficult to learn with only a few paired examples. Therefore, we instead take the outputs $\mathbf{h}$ of the inference networks parameterizing $q(\mathbf{z}|\mathbf{x})$ and $q(\mathbf{z}|\mathbf{y})$ and merge them to form the multimodal inference network (table 5.3). We find this strategy works well for us and it reduces the total number of network parameters.

$\mathbf{h}_\mathbf{x}$ linear	$\mathbf{h}_\mathbf{y}$ linear
$\oplus$	
ELU	
linear $\boldsymbol{\mu}$	linear + softplus $\boldsymbol{\sigma}$

Table 5.3: Multimodal inference network.

5.5.2 Generation Networks

The two model distributions $p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{y}|\mathbf{z})$ are each parameterized by neural networks known as *generation networks*. The fully-connected architecture is depicted in table 5.4. The output activation is a sigmoid for black-and-white images, and a softmax function for colour images. The output $\mathbf{p}$ here represents the probability parameters for either the multivariate Bernoulli or the multivariate categorical distribution, again corresponding to black-and-white or colour images respectively.

Table 5.5 depicts the deconvolutional generation network. Here multiple residual blocks are used which contain deconvolutional layers with 3×3 receptive fields. Fractional striding is incorporated in some of these blocks to increase the dimensionality.

$\mathbf{z}$
linear layer + ELU linear layer + ELU
linear layer sigmoid or 256-way softmax
$\mathbf{p}$

Table 5.4: Fully-connected generation network.

$\mathbf{z}$
linear layer
ELU + residual block $(\times n)$
sigmoid or 256-way softmax
$\mathbf{p}$

Table 5.5: Deconvolutional generation network.

5.6 Autoregressive Network

Here we outline the generation network used to parameterize the autoregressive model of section 4.6. Recall that the model is of the form $p(\mathbf{x}|\mathbf{z}) = \prod_i p(x_i|x_{<i}, \mathbf{z})$, where the pixel ordering is set to be row-by-row and pixel-by-pixel. In this case, we use both the observation $\mathbf{x}$ and latent $\mathbf{z}$ as inputs to the network. However, we cannot use standard convolutions because they introduce connections between pixels x_i which contradict our model. Instead we incorporate a *masked convolution* whereby elements of the convolution kernel are set to zero for connections which do not exist in our model. Figure 5.3 shows our network architecture. This network is one variation of the PixelCNN model [van den Oord et al., 2016a,b]. The first part of the network is a convolutional network as before, and this is attached to a second part with input $\mathbf{x}$ which incorporates masked convolutions to enforce the dependencies in the autoregressive model.

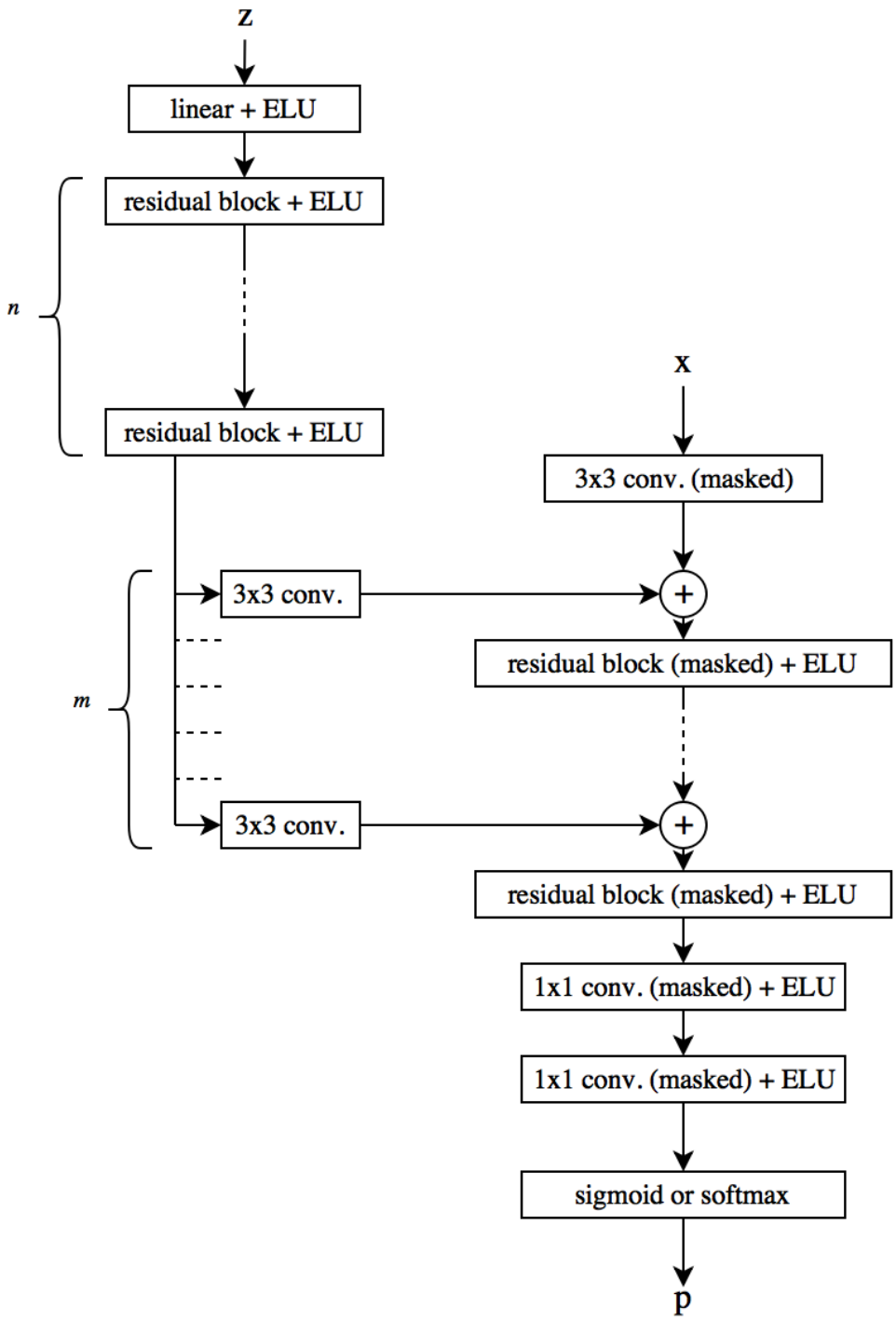


Figure 5.3: Autoregressive generation network architecture.

5.7 Variational Autoencoder

We begin by describing the *autoencoder* neural network which is a mapping from an input to itself. We can think of it in two steps: $\mathbf{x} \rightarrow \mathbf{h} \rightarrow \tilde{\mathbf{x}}$. Importantly, the intermediary layer $\mathbf{h}$ is of lower-dimension than the input $\mathbf{x}$ which means the network cannot simply learn the identity function. With the right objective function, the bottleneck layer $\mathbf{h}$ can force the network to learn a lower-dimensional representation of the input.

Similarly, when our generative model and variational distribution are parameterized by neural networks, the combination of generation and inference networks is known as a *variational autoencoder* (VAE) in the literature [Kingma and Welling, 2013]. For the single modality case, we have a mapping from the input $\mathbf{x}$ to the Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, which are then used to sample $\mathbf{z}$ from $q(\mathbf{z}|\mathbf{x})$. The generation network maps this sampled $\mathbf{z}$ to the Bernoulli or categorical probabilities $\mathbf{p}$, which can be used to sample from the model distribution $p(\mathbf{x}|\mathbf{z})$. Here we have a similar pattern as the autoencoder where there is an intermediary mapping to a lower-dimensional latent space.

The difference comes from the stochasticity inherent in the variational autoencoder mapping. The VAE is able to capture uncertainty, which makes it more robust to errors and noise in the data. In the multimodal case, if the goal is to reconstruct a modality $\mathbf{y}$ from an ambiguous input $\mathbf{x}$, the VAE is able to express its uncertainty about this ambiguity. There will be high variability in the sampled reconstructions. In contrast, the deterministic autoencoder is reduced to produce the same output $\mathbf{y}$ every time.

An additional advantage of the VAE approach is the generative model which is learned. Synthetic samples resembling the data generating process can be produced with this model. This is not possible with the autoencoder.

5.8 Weight Normalization

At a high level, a deep network is simply a combination of linear transforms and nonlinearities of the form:

$$v \;=\; \sigma(\mathbf{w}^T\mathbf{u} + b) \tag{5.2}$$

where v is a single output unit, $\mathbf{u}$ is the input, $\mathbf{w}, b$ are parameters, and σ is an activation function. Suppose we redefine $\mathbf{w}$ in terms of two new parameters g and $\mathbf{r}$:

$$\mathbf{w} = \frac{g}{||\mathbf{r}||}\mathbf{r}. \tag{5.3}$$

This disentangles the magnitude and direction of the original parameter vector $\mathbf{w}$. It can be shown that by training the network with respect to g and $\mathbf{r}$ instead, we accelerate the convergence of the stochastic optimization procedure [Salimans and Kingma, 2016]. This is known as *weight normalization* and we incorporate it into all our networks. It has competitive performance when compared with another highly successful method known as batch normalization [Ioffe and Szegedy, 2015], but at a lower computational cost.

5.9 Parameter Initialization

Of fundamental importance when optimizing deep networks is the initialization of parameters [Sutskever et al., 2013]. Consider the layer in equation 5.2. We ideally want to zero-initialize the parameters because we initially have no preference for positive or negative values. But this strategy fails because it makes all gradient computations equal to zero, and learning is not possible. A better approach is to initialize parameters from a Gaussian distribution centered at zero with a small variance. This enables learning while centering the parameters around zero as desired.

There is still a possibility that the initialized parameters make the linearity in equation 5.2 extremely large in magnitude. This may saturate the activated output $\mathbf{y}$, pushing it to regions which have gradients very close to zero, and again this will prevent learning. One solution is to randomly initialize the parameters using a variance that is inversely proportional to the number of input units $\mathbf{x}$ [Glorot and Bengio, 2010]. A more effective approach is by way of batch normalization which scales the outputs of each layer in the network. Unfortunately it is computationally expensive and introduces dependencies between minibatch samples during training. We instead turn to the strategy of Salimans and Kingma [2016] which resolves both these problems.

The idea is to initialize the parameters $\mathbf{r}$ and g such that the scale of input $\mathbf{x}$ to output $\mathbf{y}$ is fixed for all layers of the network. This is done by first randomly initializing $\mathbf{r}$ using a zero-centered Gaussian distribution. We then feed a single minibatch of data

through the network, initializing the parameters g and b such that each layer output y is centered at zero with unit variance. This has the effect of deterring layer outputs from regions of saturation and results in a more resilient optimization process.

6. Related Work

In this chapter we review related and competing methods for multimodal learning, density estimation, as well as synthesizing and translating between modalities.

6.1 Variational Autoencoders

The term *variational autoencoder* (VAE) was proposed by Kingma and Welling [2013] to describe the combination of neural networks and probabilistic modeling which we use in this thesis. The use of reparameterization to make inference and learning tractable has been proposed by Challis and Barber [2012b], Kingma and Welling [2013], and Rezende et al. [2014a]. The use of model and variational distributions in the VAE is a special case of the Helmholtz machine introduced by Dayan et al. [1995], although the approach to learning is different.

Many variants of the VAE have been recently proposed including an approach which extracts representations of data which are invariant to certain sensitive factors, e.g. race, gender [Louizos et al., 2015]. The DRAW model [Gregor et al., 2015] is another VAE example which uses sequential models for image generation and is inspired by the attention mechanisms of the human eye. Rezende et al. [2016] use a variant of DRAW for one-shot generalization tasks. As a final example, the variational auto-encoder framework has also demonstrated state of the art performance in semi-supervised classification [Kingma et al., 2014].

In our models, we use a Gaussian variational approximation with a diagonal covariance matrix. Much effort has been put into enriching this distribution in order to better capture the true posterior. A conceptually straightforward solution is to introduce a hierarchy of stochastic latent variables, where the generative process begins at the topmost latent and culminates at the observed variable. The net effect of this

approach is a much more expressive variational approximation. Inference in such models is typically problematic, but Sonderby et al. [2016] have proposed a method which makes it possible to learn such a hierarchical model.

Another approach is to use a series of transformations of a latent variable $\mathbf{z}$ from a simple initial distribution [Rezende and Mohamed, 2015]. The transformed variable can be seen as coming from a much richer distribution, and the change of variables mapping is constructed so that the variational lower bound is easy to estimate. The inverse autoregressive flow [Kingma et al., 2016] is an example of this which has state of the art density estimation performance on image benchmarks. One final method is to introduce auxiliary variables into the inference process in a way that leaves the generative model unchanged but results in a more flexible variational distribution [Maaloe et al., 2016]. In principle, all of these methods can be used to extend our proposed model.

6.1.1 Structured Prediction

Sohn et al. [2015] introduce the *conditional variational autoencoder* (CVAE) which uses the efficient learning methods of the VAE framework for structured prediction on image data. Here, a latent representation of the input image is learned and used in combination with this input, to predict an output image. The probabilistic approach allows their method to produce multiple possible outputs for a given input. Shu et al. [2017] merge the CVAE with a joint model similar to figure 4.1 and use this for structured prediction on image data. Similarly, Pu et al. [2016] use the VAE framework to predict labels and captions from images. In contrast to these methods, our approach focuses on learning a shared representation which can be used to translate between modalities in both directions.

6.2 Multimodal Learning

The use of multiple modalities has shown to improve performance on a variety of tasks. For example, image data has been combined with textual captions and other data to improve the accuracy of classification algorithms on a variety of tasks [Guillaumin et al., 2010, Huiskes et al., 2010]. A different approach is the one of Frome et al. [2013] which maps textual data (words) into a latent space, and then trains a

neural network to learn the mapping between images and words in this space. This in effect creates a shared representation over the image and text modalities. It has resulted in impressive performance on zero-shot learning, where images from categories never-before-seen by the network are mapped onto the embedding space at test time. Norouzi et al. [2013] further extend and improve this approach. Unfortunately, these methods are not able to utilize unpaired modalities and do not work when some modalities are missing. The VAE framework used in this thesis is advantageous because it can construct representations even when certain input sources are missing, and is also suitable in a semi-supervised setting, which is desirable because paired data is usually expensive.

Ngiam et al. [2011] investigate the problem of learning a shared representation of audio and video signals. They compare several deep autoencoder configurations which are trained to simultaneously reconstruct each modality. They found that the shared representation improved performance on a variety of downstream classification-based tasks. The method is also compared against another approach which uses a latent variable model known as the restricted Boltzmann machine (RBM) [Smolensky, 1986] to extract separate representations of each modality, which are then merged into a shared representation using canonical correlation analysis (CCA) [Hardoon et al., 2004], a method which solely uses linear transformations. Surprisingly, they found that the simpler CCA approach resulted in better classification performance for certain tasks. Note also that the autoencoder parameters were initialized using a scheme involving RBMs [Hinton and Salakhutdinov, 2006].

Both autoencoder and variational autoencoder methods can be considered a form of multitask learning [Caruana, 1997] since the respective objective functions are designed to reconstruct multiple modalities. The core difference between the autoencoder-based approach and the approach taken in this thesis is that the shared representations learned are probabilistic in nature. Uncertainty about the observed data is taken into account, and this is reflected in the sampled reconstructions. If for example an observed image is ambiguous about the underlying concept being depicted, there will be a much larger diversity in statistical structure of the reconstructed images.

Finally, [Srivastava and Salakhutdinov, 2012] learn shared representations of image-text as well as audio-video data using a latent variable model known as the deep Boltzmann machine (DBM) [Salakhutdinov and Hinton, 2009]. They evaluate the quality

of the learned representation on information retrieval and classification tasks, and improve upon the performance of autoencoder-based methods. This DBM approach to multimodal learning has been applied in various settings, including facial recognition and image-text retrieval [Feng et al., 2013, Yi et al., 2015]. [Sohn et al., 2014] use a different objective function from the previous approaches to learn joint representations using a combination of RBMs and recurrent neural networks. They achieve state of the art performance on a downstream visual recognition task. The deep Boltzmann machine and similar models are also applicable to image generation [Salakhutdinov, 2009].

6.3 Implicit Models for Image Generation

The generative procedure can also be formulated without explicitly specifying a probability distribution $p(\mathbf{x})$; this is known as an *implicit* probabilistic model [Mohamed and Lakshminarayanan, 2016]. The generative adversarial network (GAN) [Goodfellow et al., 2014] is one example which tends to produce images with a high visual fidelity. The objective function being optimized in this case results in probability mass assigned to some plausible regions of the data space, while other regions may be ignored [Theis et al., 2015]. This is advantageous if the goal is to produce highly plausible examples of the data, but disadvantageous for other applications, e.g. compression, where a severe underestimation of the probability may be very expensive. Isola et al. [2016] use the adversarial framework for image-to-image translation. This has been extended to the unsupervised setting where paired images are unavailable [Liu et al., 2017, Zhu et al., 2017]. Unfortunately, there is currently no well-established method for comparing these models other than by visual inspection.

6.4 Fully-Observed Models for Image Generation

Generative models for images can be constructed without latent variables. The most straightforward approach is to assume a fully-factorized model $p(\mathbf{x}) = \prod_{i=1}^{m} p_i(x_i)$ where each x_i is a pixel of an image $\mathbf{x}$. Each $p_i(x_i)$ is usually modeled as a Bernoulli or categorical distribution, and learning is accomplished by directly maximizing the likelihood. One method to directly improve this simplistic approach is by imposing

an ordering of the elements x_i such that the model becomes

$$p(\mathbf{x}) = \prod_{i=1}^{m} p_i(x_i|\text{pa}(x_i)),$$

where $\text{pa}(x_i)$ represents the predecessors or parents of x_i. An earlier example of this is Neal [1992], and a more recent example is Larochelle and Murray [2011] which improved upon the performance of all previous methods on image benchmarks. Such models are known as autoregressive models (section 4.6) because they define an ordering of pixels in the image.

The PixelRNN and PixelCNN models of van den Oord et al. [2016a] are the current state of the art in this respect, where the pixels are ordered row-by-row. The PixelCNN model has been further extended by Salimans et al. [2017] who introduce several architectural changes for improved efficiency. These methods perform well at density estimation and image generation, and we can also use them for representation learning by incorporating them into latent variable models, as is done in this thesis. Chen et al. [2016] and Gulrajani et al. [2016] are two recent examples of variational autoencoders which use an autoregressive model (PixelCNN), and arrive at state of the art density estimation on image benchmarks.

7. Experiments

We begin by outlining the model configurations used in our experiments. Our models are compared with state of the art methods on a benchmark dataset in the single modality setting. We then move to the multimodal learning case, evaluating our model on a variety of datasets. We assess the model's ability to infer one modality from the other, and we also investigate the extent to which the shared representation captures both modalities.

7.1 Architectures Tested

In this section, we describe the neural network architectures used in our models. The original variational autoencoder introduced by Kingma and Welling [2013] consists of inference and generation networks solely built using fully-connected layers. We introduce weight normalization (section 5.8) into both networks to improve performance, and we refer to this model as the VAE. We also use weight normalization in all our models. The VAE uses 450 hidden units in each intermediary layer.

The primary architecture used in our experiments consists of a convolutional inference network and a deconvolutional generation network (tables 5.2 and 5.5), which we abbreviate as a CNN-VAE. We make use of 5 residual blocks in each of the inference and generation networks of which 3 are strided blocks (either for upsampling or downsampling). The blocks of each network alternate between striding and no striding. In certain experiments, we replace the convolutional generation network with an autoregressive network (figure 5.3), and we refer to this architecture as the AR-CNN-VAE. Here we use the 5 residual blocks as before with an extra 3 masked residual blocks in the generation network.

7.2 Benchmarking with a Single Modality

We trained these models on the MNIST dataset of handwritten digits [LeCun et al., 1998] using a latent dimensionality of 50, batch size of 256, and learning rate of 0.002. We used 16 feature maps in each convolution layer of the CNN-VAE and AR-CNN-VAE. The MNIST dataset is a standard benchmark for image modeling consisting of 50000 training examples and 10000 test examples. Table 7.1 compares the lower bound on $\log p(\mathbf{x})$ for these configurations, and also includes the performance of state of the art models for a single modality. Although not state of the art, the CNN-VAE is a competitive model, and we use it as the foundation for multimodal learning in the sections which follow. The AR-CNN-VAE is a much more powerful model which can be used if an increase in performance is required. The disadvantage with this model, and autoregressive models in general, is that sampling is computationally expensive because each pixel must be generated one at a time.

Model	$\log p(\mathbf{x})$
VAE	≥ -90
CNN-VAE	≥ -87
AR-CNN-VAE	≥ -82
VAE [Kingma and Welling, 2013]	≥ -98
Normalizing Flows [Rezende and Mohamed, 2015]	≥ -85
DRAW [Gregor et al., 2015]	≥ -81
IAF [Kingma et al., 2016]	≥ -80

Table 7.1: Log-likelihood lower bound on the MNIST test set. Models in the upper half are used in this thesis, and state of the art models are in the lower half.

Synthetic samples are generated by sampling from the prior $p(\mathbf{z})$ and then subsequently sampling from the learned likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Figure 7.1 shows synthetic MNIST samples generated from the VAE-CNN model. We also trained the model on the more challenging CIFAR-10 dataset [Krizhevsky and Hinton, 2009] which consists of 32×32 colour images of which we have 50000 training and 10000 test examples. Figure 7.2 shows synthetic samples of CIFAR images. Although these generated samples are blurry, they do seem to capture the underlying shape and composition of the test set images. For this experiment, we used 64 feature maps in each layer and a latent

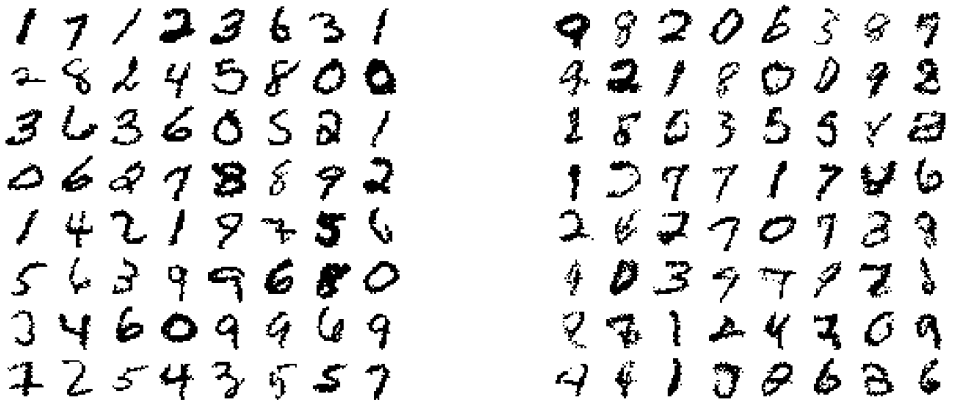


Figure 7.1: *Left*: Samples from MNIST test set. *Right*: Synthetic samples from CNN-VAE.

dimensionality of 200.

7.3 Multimodal Learning with MNIST

We divide the MNIST images into top and bottom halves, and treat each half as a separate modality. A shared representation is learned which can be used to reconstruct one modality from the other. We optimize the objective function of equation 4.12. The training set consists of 1000 complete MNIST images (i.e. paired examples), and the remaining images are split evenly into either top half or bottom half examples. We train a multimodal CNN-VAE on this dataset using minibatches of 64 paired examples and 256 unpaired examples, a learning rate of 0.001, and 16 feature maps in each layer. The model achieves a lower bound of -95 on $\log p(\mathbf{x}, \mathbf{y})$ while using only 1000 paired examples. This is the key strength in our approach; we only require a few paired examples.

The learned latent representation can be used to reconstruct one modality from the other. For example, the variational distribution $q(\mathbf{z}|\mathbf{x})$ can be used to sample $\mathbf{z}$ given an input example $\mathbf{x}$, and this latent can then be used to sample from both $p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{y}|\mathbf{z})$. We can also reconstruct using the combined distribution $q(\mathbf{z}|\mathbf{x}, \mathbf{y})$ if both modalities are observed. Figure 7.3 depicts reconstructed samples from the CNN-VAE model.

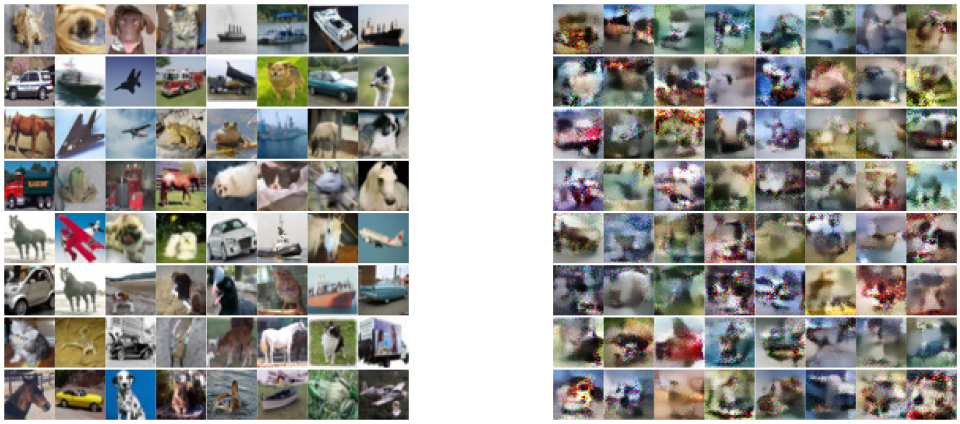
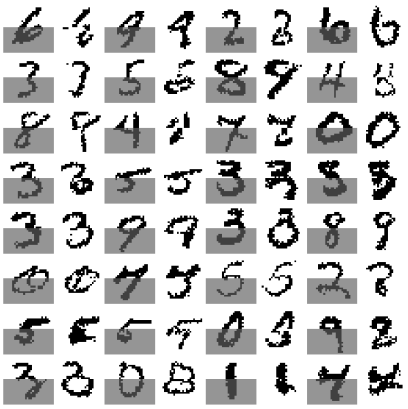


Figure 7.2: *Left*: Samples from CIFAR test set. *Right*: Synthetic samples from CNN-VAE.

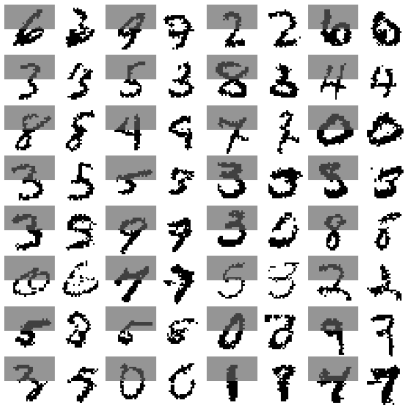
7.4 Colored MNIST

We increase the difficulty of multimodal learning by constructing a coloured version of the MNIST dataset using the method of Liu et al. [2017]. We randomly colour each MNIST digit one of three colours, and we then map these digits to a second set where each digit's edge map is extracted and coloured differently. There is a one-to-one mapping between the colours in each set. Figure 7.4 shows test set samples for the two image modalities, each of which is a different view of the underlying digit concept.

Again we train the CNN-VAE model in the semi-supervised setting using 1000 paired and 49000 unpaired examples. Again we use minibatches of 64 and 256 for paired and unpaired data respectively, and a learning rate of 0.001. Here we use 32 feature maps in all convolutional layers, and a latent dimensionality of 200. Figure 7.5 shows the sampled reconstructions from the trained CNN-VAE model. Figure 7.6 shows multiple reconstructions from the same input image. Here we notice variations in each generated sample because the probabilistic framework we use takes into account uncertainty in the observed modality. This is useful in situations where an observation is ambiguous, and the model will capture this by producing higher variation in the reconstructed samples.



(a) Reconstruction from top half.



(b) Reconstruction from bottom half.

(c) Reconstruction from complete image.

Figure 7.3: Reconstructions sampled from the CNN-VAE model for cases when one or both modalities are observed. The grayed area represents the portion of the test image that is not available to the model during the reconstruction process.

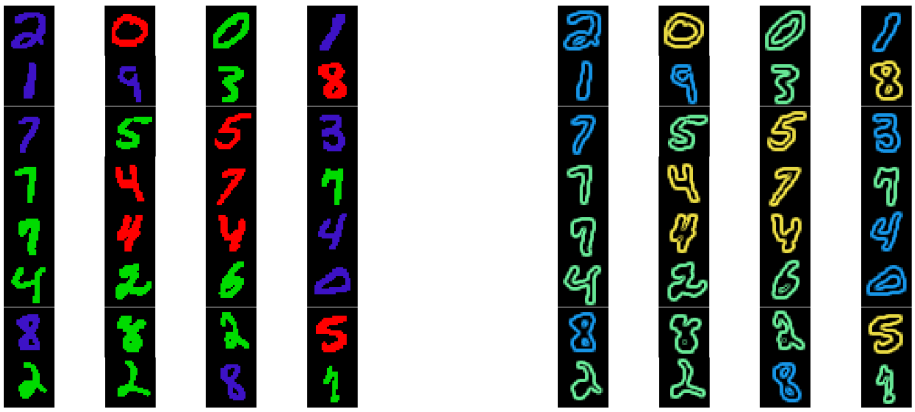


Figure 7.4: Test samples from the coloured MNIST dataset. On the left are the solid digits and on the right are the corresponding edge maps. Blue is mapped to blue, green to green, and red to yellow.

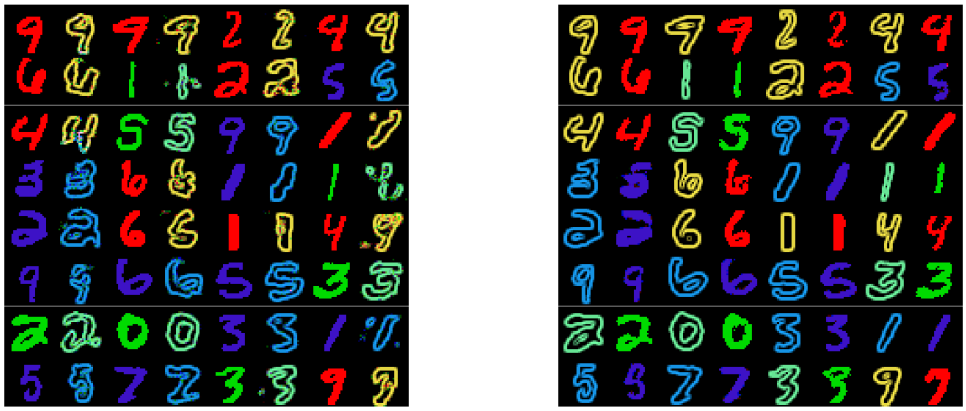


Figure 7.5: *Left:* Edgemap reconstructions from solid digits using the CNN-VAE model. *Right:* Solid digit reconstructions from edgemaps.

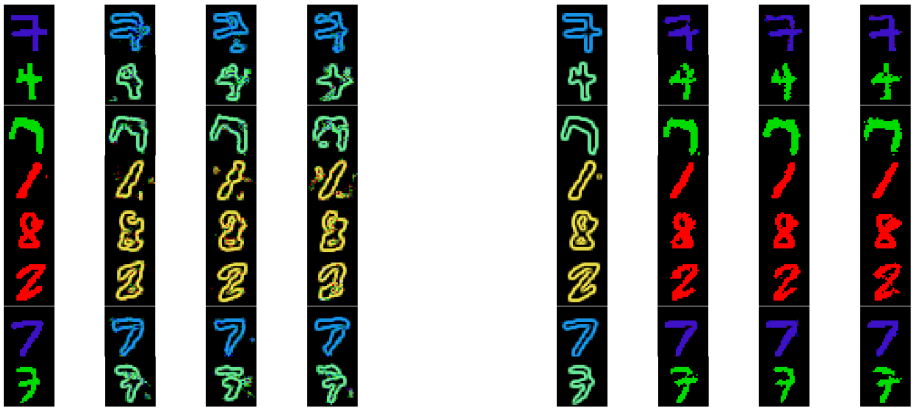


Figure 7.6: *Left:* Multiple edgemap reconstructions from solid digits. *Right:* Multiple solid digit reconstructions from edgemaps.

7.4.1 Alternative Objective

We also train the same CNN-VAE network using the alternative objective function of section 4.5. Figure 7.7 shows multiple reconstructions from the same initial image. Here we see that the reconstructed images are much more crisp, meaning that this alternative objective does in fact improve the model's ability to translate. There is also less overall variation between samples from the same image.

7.4.2 Evaluating the Shared Representation

In order to evaluate the shared representation's ability to capture correlations across the two modalities, we can assess its performance on a unique classification task [Ngiam et al., 2011]. For our purposes, classification is the problem of predicting MNIST digit labels (i.e. $0, \ldots, 9$) using raw images or extracted representations. This experiment consists of training the classifier using one modality and then testing its accuracy using the other modality. This is all done using representations extracted from the CNN-VAE. More formally, modality $\mathbf{x}$ from the training set is used to create latents $\mathbf{z}$ by sampling from $q(\mathbf{z}|\mathbf{x})$. The classification algorithm is then trained to predict the digit label using $\mathbf{z}$ as input. At test time, modality $\mathbf{y}$ from the test set is used to create latents this time by sampling from $q(\mathbf{z}|\mathbf{y})$. These latents are used as input in the

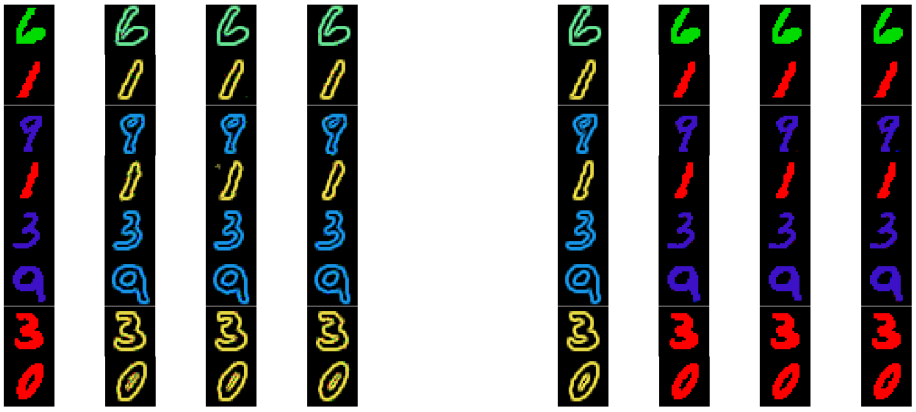


Figure 7.7: Reconstructions using the translation objective. *Left:* Multiple edgemap reconstructions from solid digits. *Right:* Multiple solid digit reconstructions from edgemaps.

trained classifier to predict the digit labels. If the shared representation has in fact captured correlational structure across the two modalities, we would expect the classifier accuracy to be significantly greater than 10%, which is the random guessing baseline. We also perform this experiment in the opposite direction by training on modality **y** and testing on **x**.

The classifier we use is called a multilayer perceptron (MLP) which is a neural network with a single hidden (fully-connected) layer and a softmax output which represents the probability of occurrence of each of the 10 possible classes. This entire experiment is also performed using the raw images directly as classifier input, in order to correct for any statistical structure which is superficially shared between the two modalities. Table 7.2 shows the classification results for this set of experiments. We see that there is a dramatic improvement in accuracy when moving from the raw images to the CNN-VAE representation. This is evidence that the shared representation learned by our model is invariant to each modality and is able to capture correlations between modalities which are highly nonlinear at the level of the raw inputs. We also include performance results when the classifier is trained and tested using the same modality. There is an improvement when using the CNN-VAE representation as expected.

Training Modality	Test Modality	Method	Accuracy
Solid Colors	Edgemaps	Baseline	10%
		Raw Image Inputs	45.82%
		CNN-VAE Representation	**92.21%**
Edgemaps	Solid Colors	Baseline	10%
		Raw Image Inputs	57.63%
		CNN-VAE Representation	**91.67%**
Solid Colors	Solid Colors	Raw Image Inputs	94.88%
		CNN-VAE Representation	96.88%
Edgemaps	Edgemaps	Raw Image Inputs	93.86%
		CNN-VAE Representation	96.77%

Table 7.2: Accuracy for downstream classification in various configurations.

7.4.3 Generalization

Here we test the ability of the CNN-VAE model to generalize to never-before-seen concepts. The model is trained in a semi-supervised setting as before, but this time only using the digits between 0 and 7. We then test the model's reconstruction ability on the digits 8 and 9; figure 7.8 depicts these results. Generalization to these unseen concepts is successful to varying degrees.

7.4.4 Autoregressive Model

The autoregressive generation network is a powerful extension to our convolutional variational autoencoder. Here we test the AR-CNN-VAE model on the coloured MNIST dataset, again in a semi-supervised setting. Figure 7.9 plots the log-likelihood lower bound of both CNN-VAE and AR-CNN-VAE. Each model was trained for about 2000 epochs, where one epoch is one pass over the complete paired dataset. As we see, there is a significant performance improvement when using the autoregressive generative process. Figure 7.10 depicts AR-CNN-VAE reconstructions on the coloured MNIST dataset. Note that the finer level details are much cleaner in these samples in comparison to original CNN-VAE samples of figure 7.5. The autoregressive model introduces dependencies between pixels in local regions of the image, which improves the quality of these images. The downside is that the sampling process is much more

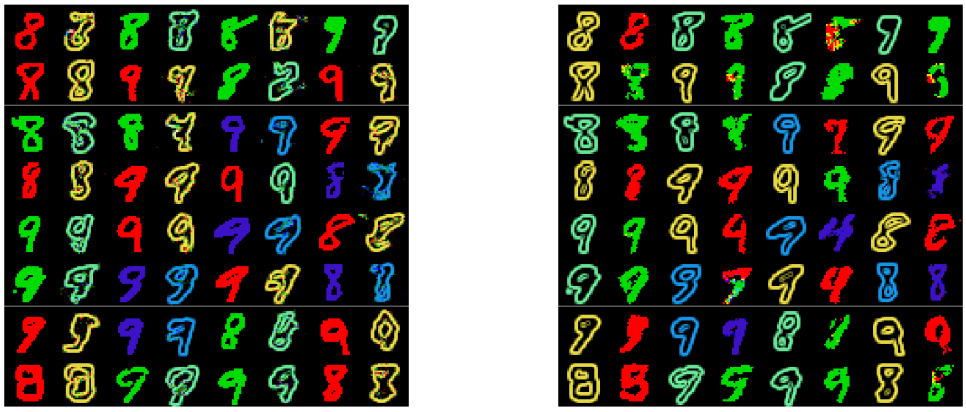


Figure 7.8: *Left:* Edgemap reconstructions from previously unseen concepts. *Right:* Solid digit reconstructions from previously unseen concepts.

involved. The pixels must be generated sequentially because they depend on the sampling of previous pixels.

7.5 Day and Night

We also evaluate our model on the Day-Night dataset [Zhou et al., 2016] which is a collection of photographs taken at different times of day at 17 urban locations (figure 7.11). There are a total of 1722 photographs which we sort into day-time and night-time images. Since the camera at each location had a fixed position, we are able to construct pairings of these images. The downside is that we have very few paired images and the effective sample size is in fact much smaller because these images come from just 17 locations. We augment this dataset to a total of 9434 paired training and 3003 paired test examples by forming many-to-many mappings between day and night images at each location. Nevertheless, the effective sample size remains much smaller, and we evaluate the performance of the CNN-VAE in this extreme setting of semi-supervised learning. Note that we use 11 locations in the training set and the other 6 locations in the test set.

Additionally, we use unpaired images of daytime and nighttime scenes from the AMOS dataset [Jacobs et al., 2007]. The images come from 29945 cameras across the

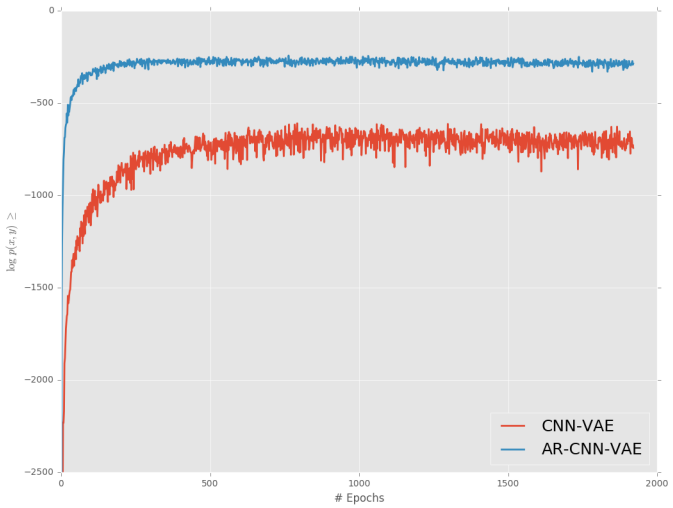


Figure 7.9: Log-likelihood lower bound for CNN-VAE and AR-CNN-VAE models on the coloured MNIST test.

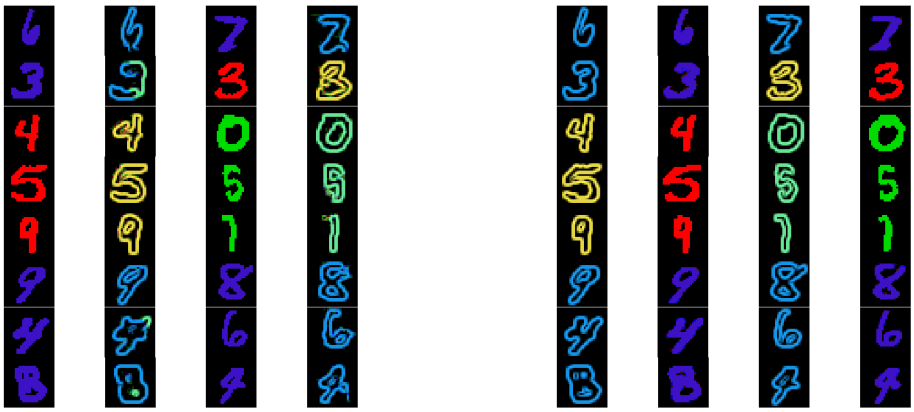


Figure 7.10: *Left:* Edgemap reconstructions from solid colors using the AR-CNN-VAE model. *Right:* Solid digit reconstructions from edgemaps.

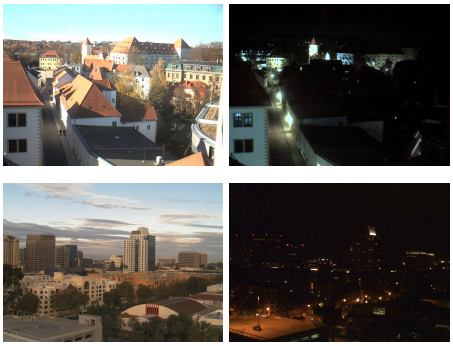
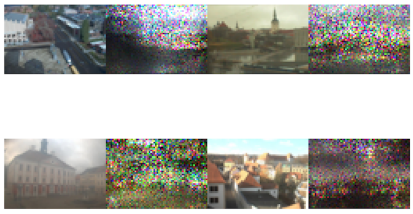


Figure 7.11: Paired examples from the day-night dataset (before downsampling).

United States and have a high variability in quality. Many examples were of low resolution and some were overlaid with pieces of text and other graphics. We curate 25423 daytime and 17865 nighttime images coming from 800 cameras. Our paired examples are purely of buildings and cityscapes, while the AMOS data comes from both urban and natural settings and is of variable resolution. We can think of this as a mild form of transfer learning, which refers to the setting where unlabelled and labelled data come from different distributions.

All images are downsampled (or upsampled) to a 44×64 resolution, and we train the CNN-VAE model on this data using minibatch sizes of 64 and 256 for the paired and unpaired data respectively. We found that a learning rate of 0.001 was stable. Figure 7.12 shows modality reconstructions from the trained data as well as samples generated from the model by first sampling from the prior $p(\mathbf{z})$ and subsequently sampling from each of $p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{y}|\mathbf{z})$. Unfortunately, the generated modalities are quite blurry. The problem of multimodal learning in this setting is very challenging, but there is room to improve by increasing the number of paired examples.



(a) Daytime to nighttime.

(b) Nighttime to daytime.

(c) Joint day-night samples from the generative model. Each consecutive pair of images is a single sample.

Figure 7.12: Modality reconstructions and samples from CNN-VAE model on day-night dataset.

8. Conclusion

In this work, we investigated the multimodal learning problem using two image modalities. We incorporated a probabilistic framework which allows for efficient inference and learning, and our approach has the advantage of working in a semi-supervised setting, where few paired examples are available. We proposed a variant of the variational objective function which improves the quality of image translations, and allows our approach to better scale to more than two modalities.

In our evaluation, we found that our models were successfully able to infer one modality from another, and the quality of translations were improved with an autoregressive generation network. We took the semi-supervised setting to its limit in the Day-Night dataset which resulted in lower quality image reconstructions. However, this can potentially be improved by increasing the number of paired examples or even with a larger set of unpaired examples. We also assessed the extent to which the shared representation captures the two modalities. In training a classifier using representations from one modality and testing it using representations from the other modality, we found a dramatic improvement from the expected baseline performance. This shows that our shared representation successfully merges the two information sources. Overall, our approach has demonstrated promising results for both generative modeling and representation learning in a multimodal setting using few paired examples.

8.1 Future Directions

Many future directions may be explored. The model could be evaluated in settings where there are more than two modalities, and a comparison can be made between

the joint lower bound and the alternative bound in section 4.5. Moreover, this approach may be extended to other modalities such as text and audio. Another interesting direction is in the transfer learning setting, where the unpaired and paired examples come from inherently different distributions. While we demonstrated a mild form of transfer learning, this could be pushed much further using datasets with very different distributions. The performance of our multimodal variational autoencoder may also be improved by considering different assumptions about the prior and variational approximation. For example, normalizing flows or other techniques which enrich the approximate posterior may be assessed. Finally, the ability to generalize to never-before-seen concepts in a multimodal setting can be much more extensively explored.

References

Atilim Gunes Baydin, Barak A. Pearlmutter, and Alexey Radul. Automatic differentiation in machine learning: a survey. *CoRR*, abs/1502.05767, 2015. 26

Matthew James Beal. *Variational algorithms for approximate Bayesian inference*. University of London London, 2003. 9

Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66, 1994. 29

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 6

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003. 10

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *CoRR*, abs/1601.00670, 2016. 10

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CoNLL*, 2016. 23

Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997. 41

Edward Challis and David Barber. Affine independent variational inference. In *NIPS*, 2012a. 16

Edward Challis and David Barber. Affine independent variational inference. In *Advances in Neural Information Processing Systems*, pages 2186–2194, 2012b. 39

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *CoRR*, abs/1611.02731, 2016. 43

P Robert Christian and George Casella. Monte carlo statistical methods, 1999. 9

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015. 29

Peter Dayan, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. The helmholtz machine. *Neural computation*, 7 5:889–904, 1995. 39

REFERENCES

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. 1977. 6, 9

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. 14

Otto Fabius, Joost R. van Amersfoort, and Diederik P. Kingma. Variational recurrent auto-encoders. *CoRR*, abs/1412.6581, 2014. 23

Fangxiang Feng, Ruifan Li, and Xiaojie Wang. Constructing hierarchical image-tags bimodal representations for word tags alternative choice. *CoRR*, abs/1307.1275, 2013. 42

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 40

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980. 26

Samuel Gershman and Noah D. Goodman. Amortized inference in probabilistic reasoning. In *CogSci*, 2014. 12

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 36

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011. 29

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 42

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, 2015. 39, 46

Andreas Griewank et al. On automatic differentiation. *Mathematical Programming: recent developments and applications*, 6(6):83–107, 1989. 26

Matthieu Guillaumin, Jakob J. Verbeek, and Cordelia Schmid. Multimodal semi-supervised learning for image classification. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 902–909, 2010. 40

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vázquez, and Aaron C. Courville. Pixelvae: A latent variable model for natural images. *CoRR*, abs/1611.05013, 2016. 43

Alan Hájek et al. Dutch book arguments. *The handbook of rational and social choice*, pages 173–196, 2008. 3

David R. Hardoon, Sándor Szedmák, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16 12:2639–64, 2004. 41

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 30

G E Hinton and R R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313 5786:504–7, 2006. 41

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 1998. 29

Mark J. Huiskes, Bart Thomee, and Michael S. Lew. New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative. In *Multimedia Information Retrieval*, 2010. 40

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 36

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. 42

Nathan Jacobs, Nathaniel Roman, and Robert Pless. Consistent temporal variations in many outdoor scenes. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, 2007. 54

Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957. 6

Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193, 1906. 10

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999. 9

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. 15, 16, 19, 35, 39, 45, 46

Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014. 39

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. 2016. 23, 40, 46

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 46

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60:84–90, 2012. 25

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference. *CoRR*, abs/1603.00788, 2015. 10

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951. 11

REFERENCES

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *AISTATS*, 2011. 22, 43

Yann LeCun. Gradient-based learning applied to document recognition. 1998. 26

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 46

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017. 42, 48

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S. Zemel. The variational fair autoencoder. *CoRR*, abs/1511.00830, 2015. 39

Lars Maaloe, Casper Kaae Sonderby, Soren Kaae Sonderby, and Ole Winther. Auxiliary deep generative models. In *ICML*, 2016. 40

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. 2013. 29

Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001. 11

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, 2014. 16

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. 25

Abdelrahman Mohamed, George E. Dahl, and Geoffrey E. Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20:14–22, 2012. 25

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *CoRR*, abs/1610.03483, 2016. 42

Radford M. Neal. Connectionist learning of belief networks. *Artif. Intell.*, 56:71–113, 1992. 43

Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993. 9

Yu. Nesterov. Introductory lectures on convex programming volume i: Basic course. 2005. 14

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *ICML*, 2011. 41, 51

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Gregory S. Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *CoRR*, abs/1312.5650, 2013. 41

John William Paisley, David M. Blei, and Michael I. Jordan. Variational bayesian inference with stochastic search. In *ICML*, 2012. 16

Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *NIPS*, 2016. 40

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015. 40, 46

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014a. 12, 16, 39

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014b. 18

Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. In *ICML*, 2016. 39

Sam T. Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11 2:305–45, 1999. 7

Ruslan Salakhutdinov. Learning deep generative models. 2009. 42

Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep boltzmann machines. In *AISTATS*, 2009. 41

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016. 36

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017. 43

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *NIPS*, 2015. 16

Rui Shu, Hung Hai Bui, and Mohammad Ghavamzadeh. Bottleneck conditional density estimation. In *ICML*, 2017. 40

Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE, 1986. 41

Kihyuk Sohn, Wenling Shang, and Honglak Lee. Improved multimodal deep learning with variation of information. In *NIPS*, 2014. 42

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015. 40

Casper Kaae Sonderby, Tapani Raiko, Lars Maaloe, Soren Kaae Sonderby, and Ole Winther. Ladder variational autoencoders. In *NIPS*, 2016. 40

Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Journal of Machine Learning Research*, 2012. 41

REFERENCES

Rupesh Kumar Srivastava, Klaus Greff, and year=2015 volume=abs/1505.00387 Jürgen Schmidhuber, journal=CoRR. Highway networks. 30

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013. 36

Lucas Theis, Aaron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *CoRR*, abs/1511.01844, 2015. 24, 42

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. 14

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016a. 22, 33, 43

Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016b. 33

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008. 6

Paul John Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, volume 1. John Wiley & Sons, 1994. 26

Dong Yi, Zhen Lei, and Stan Z. Li. Shared representation learning for heterogenous face recognition. In *FG*, 2015. 42

Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, 2010. 28

Hao Zhou, Torsten Sattler, and David W. Jacobs. Evaluating local features for day-night matching. In *ECCV Workshops*, 2016. 54

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. 42

A. Appendix

A.1 Variational Lower Bound for Paired Data

Here we derive the variational lower bound for two modalities $\mathbf{x}$ and $\mathbf{y}$:

$$\log p(\mathbf{x},\mathbf{y}) \;=\; \log \int \frac{p(\mathbf{x},\mathbf{y},\mathbf{z})}{q(\mathbf{z}|\mathbf{x},\mathbf{y})} q(\mathbf{z}|\mathbf{x},\mathbf{y}) d\mathbf{z} \;=\; \log\, \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\frac{p(\mathbf{x},\mathbf{y},\mathbf{z})}{q(\mathbf{z}|\mathbf{x},\mathbf{y})}\right] \tag{A.1}$$

$$\geq\; \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log \frac{p(\mathbf{x},\mathbf{y},\mathbf{z})}{q(\mathbf{z}|\mathbf{x},\mathbf{y})}\right] \tag{A.2}$$

$$=\; \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x},\mathbf{y}|\mathbf{z})] + \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log \frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x},\mathbf{y})}\right] \tag{A.3}$$

$$=\; \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{y}|\mathbf{z})] - KL[q(\mathbf{z}|\mathbf{x},\mathbf{y})||p(\mathbf{z})]. \tag{A.4}$$

A.2 Score Function Estimator

Here we derive the score function estimator of equation 3.23:

$$\nabla_{\phi}\mathbb{E}_{q_{\phi}(\mathbf{z})}[f(\mathbf{z})] \;=\; \int \nabla_{\phi} q_{\phi}(\mathbf{z}) f(\mathbf{z}) d\mathbf{z} \tag{A.5}$$

$$=\; \int q_{\phi}(\mathbf{z}) \frac{\nabla_{\phi} q_{\phi}(\mathbf{z})}{q_{\phi}(\mathbf{z})} f(\mathbf{z}) d\mathbf{z} \tag{A.6}$$

$$=\; \int q_{\phi}(\mathbf{z}) f(\mathbf{z}) \nabla_{\phi} \log q_{\phi}(\mathbf{z}) d\mathbf{z} \tag{A.7}$$

$$=\; \mathbb{E}_{q_{\phi}(\mathbf{z})}[f(\mathbf{z}) \nabla_{\phi} \log q_{\phi}(\mathbf{z})] \tag{A.8}$$

$$\approx\; \frac{1}{S}\sum_{j=1}^{S} f(\mathbf{z}^{(j)}) \nabla_{\phi} \log q_{\phi}(\mathbf{z}^{(j)}) \quad \text{where} \quad \mathbf{z}^{(j)} \sim q_{\phi}(\mathbf{z}). \tag{A.9}$$

We can interchange integration and differentiation in the first equality using Leibniz's rule. The third equality comes from noticing the derivative of the logarithm in A.6. Finally, a Monte Carlo estimate is possible for the formulation in A.8.

A.3 Alternative Variational Bound for Paired Data

Here we derive the alternative variational lower bound of section 4.5 for two modalities $\mathbf{x}$ and $\mathbf{y}$ using the assumption that $q(\mathbf{z}|\mathbf{x},\mathbf{y}) = q(\mathbf{z}|\mathbf{x})$.

$$\log p(\mathbf{x},\mathbf{y}) \;=\; \log \int \frac{p(\mathbf{x},\mathbf{y},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} q(\mathbf{z}|\mathbf{x}) d\mathbf{z} \;=\; \log\, \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\frac{p(\mathbf{x},\mathbf{y},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right] \tag{A.10}$$

$$\geq\; \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right] \tag{A.11}$$

$$=\; \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{y}|\mathbf{z})] + \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right]. \tag{A.12}$$