Experiments with Information Maximizing Generative Adversarial Networks

Cian Eastwood

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2017

Abstract

Despite recent successes, state-of-the-art artificial intelligence models still struggle on certain tasks where humans excel, particularly when a conceptual understanding of the underlying structure in data is required in order to generalise to unseen data. Recent research suggests that this can only be achieved by learning disentangled representations of the underlying explanatory factors behind the data. Information-maximizing generative adversarial networks (InfoGANs) are a recent and promising approach to learn such representations. However, the degree disentanglement achieved by this model is only assessed qualitatively, making it difficult to truly evaluate the quality of the latent variables discovered. In this work, we conduct a number of qualitative and quantitative experiments in order to elucidate the quality of the latent variables discovered by InfoGAN. We also combine the mutual information cost of InfoGAN with a recent and more stable value function. Experiments show that this combination, along with an updated gradient penalty, leads to stable and reliable disentanglement.

Acknowledgements

Firstly, I would like to thank my supervisor, Prof. Chris Williams, for his valuable advice, guidance and insight. I would also like to thank Pol Moreno for generating the data and numerous helpful conversations. Finally, I would like to thank Andrew Brock and Akash Srivastava for helpful conversations.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Cian Eastwood*)

Contents

Chapter 1

Introduction

In this introductory chapter, we present the main objectives of this dissertation and the motivation behind them. We also outline the contributions of this dissertation and its overall structure.

1.1 Motivation and objective

Artificial intelligence (AI) has enjoyed many remarkable successes in recent years, including reaching human-level performance on some object recognition benchmarks [1, 2, 3] and superhuman-level performance on complex strategic games like Go [4]. However, state-of-the-art AI models still struggle on certain tasks where humans excel [5]. A prime example is zero-shot inference [6], where models must use their knowledge about individual factors of variation in the data to reason about new data with unseen factor combinations. Another example is transfer learning [7], where representations learned for one task are reused for several other tasks. While Lake et al. [5] suggest incorporating domain-specific knowledge into models such as the basic laws of physics, the quest for AI demands more powerful algorithms which learn with generic priors. An intelligent agent must fundamentally understand the world around us, and Bengio et al. [8] argue that this can only be accomplished if it learns to disentangle the underlying explanatory factors hidden in the observed data. Higgins et al. [6] support this argument, demonstrating that models may obtain a basic conceptual understanding of the visual world by learning disentangled representations of the factors behind low-level sensory input. A disentangled representation is one that separates the factors

of variation, explicitly representing the important attributes of the data. For instance, given an image dataset of human faces, a disentangled representation may consist of separate dimensions (or features) for the face width, face height, hairstyle, eye colour, facial expression, etc. Such representations would allow useful information to be easily extracted and hence would likely be useful for downstream discriminative tasks, transfer learning and zero-shot inference.

Unsupervised learning aims to extract value from unlabelled data that is available in large quantities. Utilizing this wealth of unlabelled data to learn disentangled representations is critical to developing intelligent algorithms that learn and think like humans [8, 6, 5]. Ultimately, we would like to learn features that are invariant to irrelevant changes in the data. However, this unsupervised problem is ill-posed. The relevant downstream tasks are generally unknown at training time and hence it is difficult to deduce a priori which set of variations will ultimately be relevant. Thus, the most robust method is to disentangle as many factors of variation as possible, discarding as little information as possible [8, 9].

Deep generative models are a powerful class of probabilistic models, providing tools for the unsupervised learning of complex probability distributions [10]. Driven by the idea that some form of understanding is required in order to be able to synthesize the observed examples, deep generative modelling has become one of the leading approaches to unsupervised representation learning. It is hoped that meaningful disentangled representations will be learned automatically by a sensible generative model. However, a generative model that perfectly reproduces the data distribution can have arbitrarily bad representations[11]. More specifically, the individual dimensions may not correspond to semantic features of the data if the generator uses the latent representations in a highly-entangled way.

The generative adversarial network (GAN) [12] is currently one of the most prominent generative models, with several recent successes of note [13, 14, 15]. Information Maximizing Generative Adversarial Network (InfoGAN) is a recent extension to the GAN that encourages the generative model to learn more interpretable and meaningful representations in an completely unsupervised manner [11]. This extension is remarkably effective, with the model appearing to learn highly semantic disentangled representations on several image datasets. Unlike previous unsupervised approaches to learning disentangled representations [16, 17, 18, 9], InfoGAN requires no prior knowledge about the underlying factors of variation in the data and scales to compli-

cated datasets. However, the degree disentanglement is only assessed qualitatively in [11], visually assessing the generated image for different types of semantic variation after varying a single latent variable. As a result, it is difficult to truly evaluate the quality of the latent variables discovered by InfoGAN. This leads to the main objective of this work—to experiment with InfoGAN on image data with known latent causes in order to quantitatively evaluate its ability to disentangle the factors of variation in data.

1.2 Contributions

Our contributions are as follows:

1. We combine the mutual information objective of InfoGAN with a recent and more stable GAN value function, the Wasserstein GAN [19, 20].
2. We show that this novel combination leads to stable and reliable disentanglement, removing InfoGAN's sensitivity to random initialization and requirement to re-tune network hyperparameters for each individual factor of variation in the data.
3. We propose new metrics to quantitatively compare the degree of disentanglement achieved by different models. These metrics extend previous attempts ([6]) and generalize to high-dimensional factors of variation.
4. We quantify the degree of disentanglement achieved by InfoGAN, providing empirical evidence that InfoGAN understands the factorial structure of the data, enabling it to successfully perform zero-shot inference.

1.3 Dissertation structure

Chapter one serves as an introduction to the dissertation topic, motivating our work and explaining our main contributions. Chapter 2 reviews GANs and their recent extensions, including InfoGAN. Chapter 3 details related material on disentangled representations, highlighting the factors that distinguish it from our work. Chapter 4 gives a brief overview of the resources, tools and data used throughout this project. Chapter 5 details how the data was generated and how the two GAN extensions were combined in a novel way to achieve reliable disentanglement. Chapter 6 describes our new metrics

for quantifying the degree of disentanglement in models before evaluating the quality of the latent variables discovered by InfoGAN using these new metrics. Finally, chapter 7 presents a conclusion of the dissertation and suggestions for future work.

Chapter 2

Background

Chapter 1 motivated our research, stated the objectives of this dissertation, outlined our main contributions and provided an overview of the dissertation structure. In this chapter, we review the relevant background material in order to facilitate a deeper understanding of the topics introduced in later chapters. Note that a familiarity with basic concepts in machine learning is assumed throughout this dissertation, particularly neural networks.

2.1 Convolutional Neural Networks

Traditional neural networks consist of 'fully-connected' layers in which every input feature is connected to every hidden unit. Intuitively, treating all of the input features equally does not seem like the best approach for images given their inherent spatial structure. Pixels that are close together are likely to be related, perhaps even part of the same object. This motivates the use of a neural network architecture that can exploit this spatial structure, namely convolutional neural networks (CNNs) [21]. CNNs slide a kernel or feature map across the input to detect a particular abstract pattern or feature, where the slide 'distance' is known as the stride. Each feature map uses a single set of weights to map every $F \times F$ region in the input to a single hidden unit in the convolutional (conv) layer, where F is the desired field size. As a result, each feature map learns to detect the same feature at any location in the input, leading to a natural invariance to object translation. Furthermore, using a single set of weights reduces the number of trainable parameters in the model, thus improving generaliza-

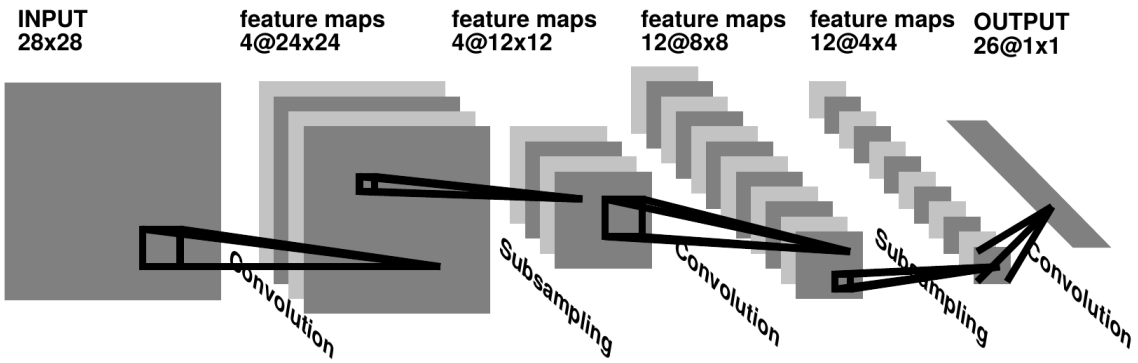
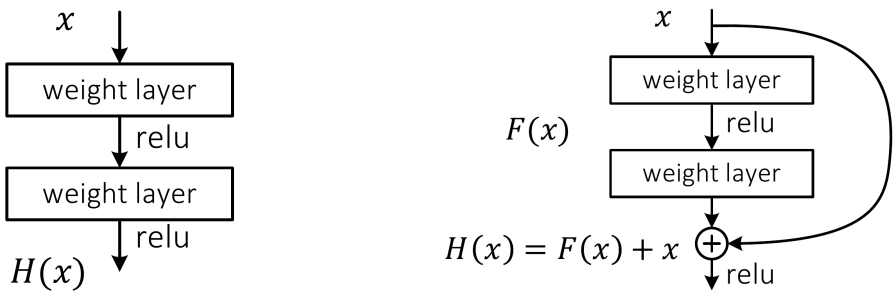


Figure 2.1: Convolutional neural network for image processing. Adopted from [21]. Figure shows alternating convolutional and subsampling layers, with each $F \times F$ region in the input 'image' being mapped to a single hidden unit in each of the feature maps, where F is the desired field size.

tion performance. As it is desirable to detect more than one feature, numerous feature maps are used in each conv layer (see figure 2.1). Finally, pooling layers are usually employed after conv layers in order to summarise the information in a given region for each feature map. This reduces the volume of the input to the layers that follow and hence reduces the required computation. However, it is worth noting that recent implementations generally replace deterministic pooling layers with increased stride convolutions. This allows the networks to learn appropriate spatial downsampling and is particularly important for the GANs discussed in section 2.3, with the generator and discriminator learning their own 'reversible' spatial upsampling and downsampling respectively [13]. CNNs are widely used in the field of image recognition, with most state-of-the-art networks, such as the popular VGG Net [22], using some variant of CNNs.

2.2 Residual Networks

Theoretically, increasing the number of layers in a neural network should not result in a higher training error as the models are nested. However, He et al. [3] provide empirical evidence that this is not the case beyond a certain depth, where conventional deeper networks begin to demonstrate serious signs of underfitting due to optimization difficulties. As a result, these deeper networks are significantly outperformed by their shallower counterparts. While clever weight initialization techniques [23, 24] and in-



(a) Plain network. **(b) Residual network.**

Figure 2.2: Residual network reformulation. Figures adopted from [3], with (a) depicting conventional networks and (b) depicting the proposed residual networks.

termediate normalization layers [25] have largely addressed the problem of vanishing gradients, increasing depth still increases the size of the solution space and hence the difficulty of the optimization objective. Note that if the added layers can be formulated as identity functions, a deeper model should have at most the same training error as its shallower counterpart. As this is not the case, He et al. [3] infer that optimizers may have trouble approximating identity functions with several nonlinear layers.

To address this issue, He et al. [3] propose Residual Networks (ResNets). Contrary to conventional networks which attempt to learn an underlying function with a nonlinear function $H(\boldsymbol{x})$, residual networks use a nonlinear function $F(\boldsymbol{x}) = H(\boldsymbol{x}) - \boldsymbol{x}$ (see figure 2.2). As shown in figure 2.2b, this is achieved by adding the input $\boldsymbol{x}$ to the function output $F(\boldsymbol{x})$ before applying the final ReLU nonlinearity. With this reformulation, if the underlying function is the identity function, the optimizer can simply push the weights to zero. Furthermore, if the underlying function is closer to an identity function than to a zero function, it should be easier for the optimizer to find the weights with reference to an identity function than to learn the function 'as a new one'. Although it is unlikely that the identity function will be optimal is real cases, He et al. [3] show that the responses of learned residual functions are usually small, suggesting that identity functions provide more sensible preconditioning than zero functions.

Compared to conventional networks, ResNets converge faster and can gain accuracy improvements from significantly increased depth. He et al. [3] achieve state-of-the-art performance on ImageNet using a 152-layer ResNet. This network is 8 times deeper than the previous state-of-the-art, VGG nets [22], despite having lower time complexity[1]. More recently, promising results have been achieved with 1001-layer residual

[1]See [26] for more details on the time complexity of convolutions.

networks [27].

2.3 Generative Adversarial Networks

GANs [12] are a powerful framework for training deep generative models through an adversarial process in which two models are trained simultaneously; a generative model G - which generates samples by transforming a multivariate noise variable $\boldsymbol{z} \sim P_{noise}$ into sample $G(\boldsymbol{z})$, and a discriminative model D - which tries to distinguish between samples from the training data $\boldsymbol{x} \sim P_{data}$ and those generated by G. $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ came from the training data rather than G. The model is trained by simultaneously adjusting the parameters of G to maximize the probability of D making a mistake and the parameters of D to maximize the probability of correctly distinguishing samples from the training data (true distribution) and G. At the end (theoretically), the generator reproduces the true data distribution and the discriminator is unable to distinguish between the samples. This training procedure corresponds to a two-player minimax game with the following value function:

$$\min_G \max_D V_{GAN}(D, G) = \mathbb{E}_{\boldsymbol{x} \sim P_{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim P_{noise}}[\log(1 - D(G(\boldsymbol{z})))]. \quad (2.1)$$

A helpful analogy is given in [12] where G is viewed as a team of counterfeiters producing fake currency and D as the police, with competition pushing both teams to enhance their mechanisms until the counterfeits are indistinguishable from the genuine currency. By defining G and D as neural networks, the whole system can be trained with backpropogation. This has both computational (no Markov chains, intractable partition functions or inference during training) and flexibility (any differentiable function is allowed) advantages. GANs also generate much sharper images than competing generative models such as the variational autoencoder (VAE) [28] due to their learning process and lack of a heuristic loss function. However, GANs can be difficult to train due to unstable training dynamics and mode collapsing (where the generator only learns to generate samples from a few modes of the training data distribution) [29, 30]. As a result, a delicate balance is required in the training of the discriminator and the generator. Some helpful techniques were introduced by DC-GAN [13], although any deviation from the suggested architecture and corresponding hyperparameters can destroy this equilibrium, resulting in nonsensical outputs. Recently, there has been a

tremendous amount of interest in improving GANs. We will describe the most relevant to our work in the sections that follow.

2.4 Wasserstein GANs

Recent research on GANs has focused on methods to stabilize training and prevent mode collapse [31, 30, 19, 20, 32, 33]. In particular, Arjovsky et al. [30] present a detailed analysis of the GAN value function and its convergence properties. They also propose an alternative value function based on the Wasserstein distance, called Wasserstein GAN (WGAN) [19], and demonstrate that its better theoretical properties lead to improved training stability and increased resistance to mode collapse. WGAN requires that the discriminator (or 'critic') D is in the set of 1-Lipschitz functions $\mathcal{D}$, which is achieved through weight clipping. The new value function is

$$\min_G \max_{D \in \mathcal{D}} V_{WGAN}(D, G) = \mathbb{E}_{\boldsymbol{z} \sim P_{noise}}[D(G(\boldsymbol{z}))] - \mathbb{E}_{\boldsymbol{x} \sim P_{data}}[D(\boldsymbol{x})]. \tag{2.2}$$

Although this alternate value function improves training stability, the authors acknowledge that weight clipping is 'a clearly terrible way to enforce a Lipschitz constraint', encouraging researchers to improve on their method. Gulrajani et al. [20] subsequently obliged, showing that this weight clipping can lead to pathological behaviour before proposing an alternative, commonly called WGAN-GP, which does not suffer from these issues. More specifically, they note that 'a differentiable function is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere' and thus are able to enforce the Lipschitz constraint on the discriminator by constraining the gradient norm of its output with respect to its input using a gradient penalty (GP). Thus, the updated value function is

$$\min_G \max_D V_{WGAN_{GP}}(D, G) = V_{WGAN}(D, G) + \lambda \, \mathbb{E}_{\hat{\boldsymbol{x}} \sim P(\hat{\boldsymbol{x}})}[(\left\| \nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}) \right\|_2 - 1)^2], \tag{2.3}$$

where the sampling distribution $P(\hat{\boldsymbol{x}})$ is implicitly defined by the following equation:

$$\hat{\boldsymbol{x}} = \epsilon \boldsymbol{x} + (1 - \epsilon) G(\boldsymbol{z}), \; \epsilon \sim U[0, 1]. \tag{2.4}$$

Gulrajani et al. [20] provide empirical evidence that this alternative constraint significantly improves training speed and architecture robustness. This robustness allows the authors to improve sample quality by exploring a wider range of architectures, including a 101-layer Residual Network ([3]) that produces samples competitive with

the state-of-the-art on the LSUN bedrooms dataset. Bellemare et al. [34] provide theoretical evidence of bias in the sample gradient estimates of the Wasserstein distance and propose an alternative probability metric, the Cramér distance. However, although they conjecture that these issues remain for the WGAN-GP with a fixed number of samples, their proposed solution, the Cramér GAN, only achieves performance improvements over the WGAN-GP when both networks are trained using a single critic update per generator update ($n_{critic} = 1$). They observe no performance improvements when $n_{critic} = 5$, as suggested in the original work [20].

2.5 Information Maximizing GANs

GANs place no restriction on the manner in which the generator may use the input noise vector $\boldsymbol{z}$. As a result, individual dimensions of $\boldsymbol{z}$ may not correspond to semantic features of the data if the generator uses $\boldsymbol{z}$ in a highly-entangled way. InfoGAN [11] splits the noise vector into two parts; $\boldsymbol{z}$, which targets 'incompressible noise', and $\boldsymbol{c}$, 'latent codes' which target salient semantic features of the data. The manner in which the generator may use the latent codes is then constrained by adding a regularisation term $I(\boldsymbol{c}; G(\boldsymbol{z}, \boldsymbol{c}))$ to the GAN objective, representing the mutual information between the latent codes $\boldsymbol{c}$ and generated images $G(\boldsymbol{z}, \boldsymbol{c})$. Intuitively, we want this mutual information to be high so that the information in the latent codes reveals as much as possible about the image, rather than being lost in the generation process. As this term is difficult to directly maximize without access to the posterior $P(\boldsymbol{c}|\boldsymbol{x})$, it is approximated by the following variational lower bound $L_I(G, Q)$:

$$L_I(G, Q) = \mathbb{E}_{\boldsymbol{c}\sim P(\boldsymbol{c}), \boldsymbol{x}\sim G(\boldsymbol{z},\boldsymbol{c})}[\log Q(\boldsymbol{c}|\boldsymbol{x})] + H(\boldsymbol{c}), \tag{2.5}$$

where the auxiliary distribution $Q(\boldsymbol{c}|\boldsymbol{x})$ approximates $P(\boldsymbol{c}|\boldsymbol{x})$ and $H(\boldsymbol{c})$ denotes the entropy of the latent codes $\boldsymbol{c}$. The auxiliary distribution $Q(\boldsymbol{c}|\boldsymbol{x})$ is parametrized as a neural network, outputting the parameters of the posterior distribution over latent codes. For discrete codes, $Q(\boldsymbol{c}|\boldsymbol{x})$ is represented by a softmax non-linearity. For continuous codes, $Q(\boldsymbol{c}|\boldsymbol{x})$ is treated as a factorized Gaussian. As Q shares all layers except the final fully-connected layer with D, maximizing the added mutual information term essentially comes at no extra computational cost. The extended framework is illustrated in figure 2.3 and defined by the following value function:

$$\min_{G,Q} \max_{D} V_{InfoGAN}(D, G, Q) = V_{GAN}(D, G) - \lambda L_I(G, Q), \tag{2.6}$$

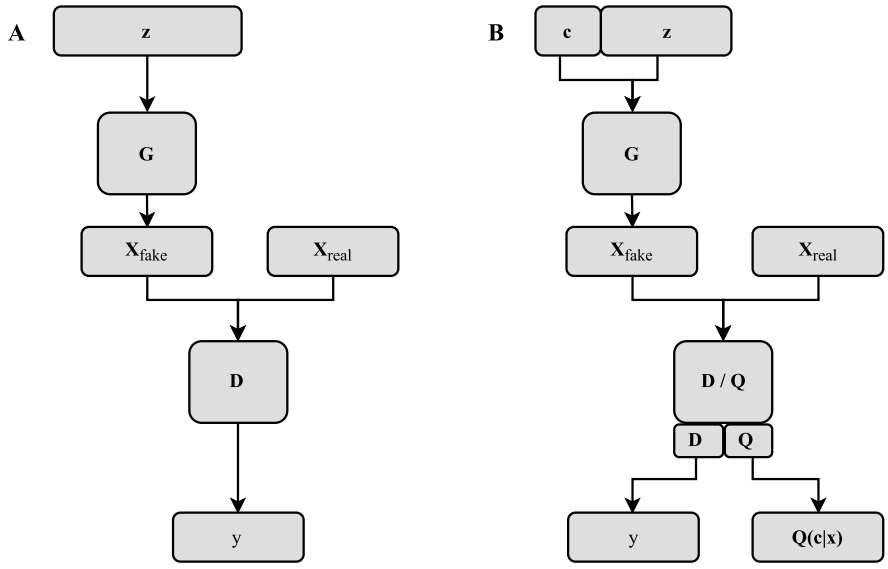


Figure 2.3: **A**: GAN framework. **B**: InfoGAN framework. $y = D(\boldsymbol{x})$ represents the probability that x came from the training data and not G.

where λ, the mutual information coefficient, is an additional hyperparameter.

Experiments on several image datasets suggest that InfoGAN discovers meaningful and highly semantic latent representations. On MNIST, a discrete code appears to capture the digit type while two continuous codes appear to capture the rotation and width. On a dataset of 3D faces, the latent codes appear to successfully capture azimuth, elevation, lighting and width. Chen et al. [11] evaluate the disentanglement of the latent representations by varying one code at a time, visually assessing the different types of semantic variation in the resulting image. A single type of semantic variation suggested that InfoGAN had successfully learned to disentangle the factors of variation in the dataset. While these qualitative results are impressive, the critical importance of disentangled representations to achieving true AI motivates further investigation into the quality of the latent variables discovered by InfoGAN.

Chapter 3

Related Work

Chapter 1 introduced the idea of disentangled representations and illustrated the crucial role that they play in achieving AI. Chapter 2 provided a review of the background material which is most relevant to our work, preparing the reader for subsequent chapters. This chapter explains related research on disentangled representations, highlighting the factors that distinguish it from our work.

3.1 Unsupervised disentangling

Most previous 'unsupervised' attempts to separate the factors of variation have required prior knowledge about the nature/number of underlying factors of variation in the data [16, 17, 18]. Others have simply not scaled well [9, 35]. However, the recent approach of Higgins et al. [6] does not suffer from these issues and hence is the most closely-related work. As Chen et al. [11] demonstrate with InfoGAN, Higgins et al. [6] demonstrate that deep unsupervised generative models are capable of learning disentangled representations if the correct learning constraints are imposed. Their proposed learning constraint is inspired by those that have been suggested to act in the human brain, encouraging the model to reduce redundancy and note statistical independencies in the data in order to learn disentangled representations. However, they do so using the VAE [28] framework. While popular, the VAE has some significant disadvantages compared to the GAN. In particular, the difficulty of specifying a good heuristic loss over complex image distributions generally leads to blurry images.

In addition, several recent works have successfully separated a class label from other

factors of variation with no supervision by combining adversarial methods and autoencoders [36, 37]. However, they rely on the autoencoder to capture the data distribution. This is a significant disadvantage for the reasons outlined in the previous paragraph.

3.2 Evaluating disentangled representations

Higgins et al. [6] also propose a method to quantify the degree of disentanglement in learnt representations. They suggest training a linear classifier to predict which generating factor caused the change between two images, where the images are exactly the same except for a change in a single generative factor. The low capacity of the linear classifier ensures that high classification accuracy can only be accomplished when the generative factors have already been disentangled in the latent space. The classifier essentially learns a mapping $f(\boldsymbol{c}_{change}) : \mathbb{R}^D \rightarrow \mathbb{R}^K$, where D is the dimensionality of the latent codes, K is the total dimensionality of all the factors of variation in the dataset and $\boldsymbol{c}_{change}$ is the 'change in latent space corresponding to a change in a single generative factor in pixel space'. Despite its success for simple 1D factors of variation, this metric may struggle to generalize to high-dimensional factors of variation where absolute difference along each individual dimension is not the most appropriate way to capture the factor's total change. Hence, we aim to explore alternative quantification metrics, including those that generalize to high-dimensional factors of variation.

Using the VAE framework, Higgins et al. [6] also demonstrate how learning disentangled representations can enable models to: a) perform zero-shot inference (see figure 3.1a); b) obtain a 'basic conceptual understanding of the visual world, such as "objectness"'. A VAE that learned disentangled representations achieved relatively high classification accuracy on the test data containing unseen combinations of factors. By contrast, a VAE that learned entangled representations (trained without the learning constraint) performed significantly worse on the test data. In order to visualize what the model understands about novel objects and hence demonstrate an understanding of basic visual concepts, the authors train one VAE on the original dataset of different 2D objects (heart, oval and square) and another on a new dataset of 2D objects (mushroom, rectangle and triangle), which were generated using the same factors of variation. A linear regressor H then joins the encoder trained on the original dataset (Enc_{orig}) with the decoder trained on the new dataset (Dec_{new}) by aligning the latent spaces $\boldsymbol{z}_{orig}$ and $\boldsymbol{z}_{new}$ (see figure 3.1b). The reconstructions $\hat{\boldsymbol{x}}_{new} = Dec_{new}(G(Enc_{orig}(\boldsymbol{x}_{new}))$ gener-

(a) Zero-shot inference. **(b) Learning basic visual concepts.**

Figure 3.1: Figures and captions adopted from [6]. **(a)** 'Models are unable to generalise to data outside of the convex hull of the training distribution (light blue line) unless they learn about the data generative factors and recombine them in novel ways'. **(b)** 'Model architecture used to visualise whether VAEs trained on the original dataset of 2D objects can reason about new object identities'.

ated by a VAE that had learned disentangled representations were far better than those generated by a VAE that had learned entangled representations, suggesting that the former had learned basic visual concepts that allowed it to reason about the properties of unseen objects. As disentangled representations should enable zero-shot inference and the emergence of basic visual concepts, tests for these properties serve as intuitive methods of evaluating the quality of latent representations. This idea will be further explored in chapters 3 and 5.

Chapter 4

Resources, Tools and Data

In chapter 2 we presented background material on the deep neural networks that were used in this project. Section 4.1 outlines both the software and hardware resources that were required to train such networks. Section 4.2 then presents the synthetic data that later facilitates a quantitative analysis of the latent variables discovered by InfoGAN.

4.1 Resources and tools

4.1.1 Software

This project was implemented using Python 2.7.5 and Tensorflow 1.0.0. Scientific computing libraries numpy and scipy were used throughout. The GANs in chapter 5 were based on code published by the authors of InfoGAN [38] and the improved WGAN [39]. Scikit-learn was used for most of the regression models in Chapter 6, along with Jupyter notebooks and Matplotlib for convenient prototyping and data visualization.

4.1.2 Hardware

Deep CNNs and ResNets are quite computationally expensive to train. It is simply not feasible to carry out adaptive experiments with such networks on CPUs. To enable GPU-accelerated functionality with Tensorflow, we used Nvidia's CUDA API and cuDNN deep learning library. This allowed us to train on several Nvidia Titan X

GPUs, which were available on university servers. Even with 4 such GPUs running in parallel, some ResNets took over 24 hours to train.

4.2 Data

One of the main objectives of this dissertation is to quantify the degree of disentanglement achieved by InfoGAN. In order to do so, we need the ground-truth values for each factor of variation. Using computer-graphics generated images of an object class (teapots) gives us direct access to these underlying scene parameters. The scene generator from [40] is used to generate the data, randomly instantiating a teapot object in one of 80 indoor scenes. The camera is at a fixed distance and centered on the object. Originally, 100000 images with dimensions $48 \times 48 \times 3$ were generated with varying scene backgrounds (see samples in figure 4.1) . However, as detailed in section 5.1.2, the dataset was later simplified by removing the backgrounds. The images are generated using the following generative factors, where each factor is independently sampled from its respective uniform distribution:

- **Pose (2D)**
  - $Azimuth \sim U[0, 2\pi]$
  - $Elevation \sim U[0, \pi/2]$
- **Appearance (3D)**
  - $Red \sim U[0, 1]$
  - $Green \sim U[0, 1]$
  - $Blue \sim U[0, 1]$

Note that the scene generator from [40] used two additional generative factors, shape and lighting, which were held constant for our experiments. These high-dimensional generative factors could be varied in future work to create richer data.

Figure 4.1: Original dataset samples.

Chapter 5

Learning Disentangled Representations

In chapter 1, we discussed the importance of learning disentangled representations from unlabelled data. Specifically, we highlighted its central role in building intelligent systems that understand the world around us. In chapter 2, we presented InfoGAN, a deep unsupervised generative model that appears to disentangle the underlying factors of variation in data to learn meaningful and highly semantic representations. However, the disentanglement was only evaluated qualitatively (by visual inspection) in [11]. In order to quantitatively assess the quality of latent variables discovered by InfoGAN, we must first train the InfoGAN on the synthetic dataset described in chapter 4. This will be the focus of chapter 5. In section 5.1, we describe the unsuccessful search for a GAN architecture (and corresponding hyperparameters) that allows the generator to learn the initial data distribution. Next, we illustrate the subsequent simplifications that were made to the dataset before exploring alternate GAN value functions in search of stability. In section 5.2, we present the stable GAN value function that was able to learn the data distribution and produce sharp images of teapots. Finally, in section 5.3 we detail the process of combining this new value function with that of InfoGAN in order to learn disentangled representations in a stable manner.

5.1 Initial InfoGAN training

5.1.1 Seeking Nash equilibria

As discussed in section 2.5, training InfoGANs is equivalent to optimizing the minimax game in equation 2.6, where D, G and Q are all neural networks. As discussed in [29], this requires finding a Nash equilibrium of a two-player 'non-cooperative' game in which both players want to minimize their own cost function. Salimans et al.[29] also note that gradient descent algorithms often fail to converge for such games as they are designed to seek a low value for the cost function rather than a Nash equilibrium.

To train an InfoGAN on our dataset, we needed to find an architecture and hyperparameter 'pair' that allowed the minimax game to converge. However, finding Nash equilibria is a very difficult problem [29]. Starting with the architectures given in [11], we tried every sensible architecture and hyperparameter combination using the 'guidelines' of DC-GAN [13]. We also tried removing the mutual information regularization term completely (setting $\lambda = 0$ in eq. 2.6). However, the instability stemmed purely from the GAN value function in eq. 2.1. As a result, the balance between the generator and the discriminator was always lost after a few iterations, with the losses diverging once the generator was no longer able to 'fool' the discriminator. As a result, the generator was unable learn the data distribution. This frustrating experience reiterated the fickle nature of the GAN training procedure and prompted a simplification of the problem. The best samples produced with this 'vanilla' GAN were very close to random noise and are depicted in Appendix A.2, while relevant architecture details are given in Appendix A.1.

5.1.2 Simplifying the problem

As discussed in the previous section, the difficulty of finding a GAN architecture that allowed the generator to 'keep up' with the discriminator prompted several simplifications in the hope of generating reasonable samples. A simpler distribution should be easier to learn. Thus, we aimed to simplify the problem for the generator by simplifying the data distribution, removing the background scene. The dimensions were also changed to $64 \times 64 \times 3$, in line with popular architectures [13]. Samples from this new dataset are shown in figure 5.1a.

(a) Groundtruth samples. **(b) WGAN-GP samples.**

Figure 5.1: Samples from the new dataset. (a) 'Real' samples from the new dataset. (b) 'Fake' samples generated by WGAN-GP.

Despite a prolonged learning period, the losses once again diverged. The best samples from the generator were produced when training was prematurely halted at the first sign of divergence. However, the generator was unable to learn the data distribution in such a small number of iterations (typically around 5). The best samples generated with this simplified dataset are given in Appendix A.3, while relevant architecture details are given in Appendix A.1.

5.2 Stable GANs

Unable to find an architecture and hyperparameter 'pair' that allowed the GAN to learn the distribution of our data, we turned to recent literature on improving the stability of GANs [31, 32, 30, 19, 20, 33]. As described in section 2.4, this led us to the improved WGAN (WGAN-GP) [20] which replaces the weight clipping of WGAN [19] with a gradient penalty to improve stability. This in turn leads to improved architecture robustness, allowing the authors to generate high-quality samples through the use of deep ResNets. By adapting the open-source implementation of WGAN-GP [39] to our dataset, we were able to generate very sharp samples without any hyperparameter tuning. In fact, it is not obvious which samples are 'real' and which are 'fake' when comparing samples from the dataset to those generated by WGAN-GP (see figure 5.1).

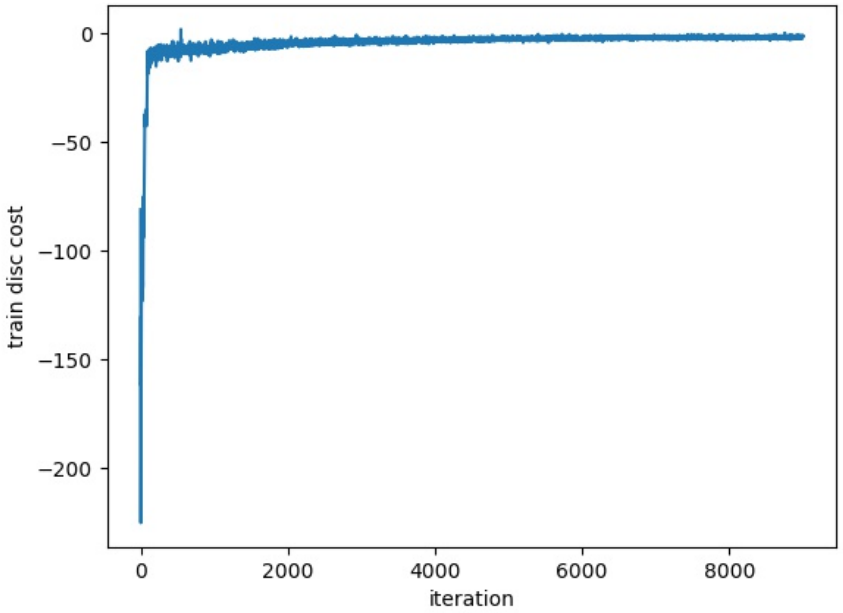


Figure 5.2: WGAN discriminator cost over iterations. Train disc cost indicates discriminator cost on the training set.

As shown in figure 5.2, the stability of WGAN-GP allows the minimiax game to converge. Furthermore, unlike GANs value function, WGANs unbounded value function correlates with sample quality. Thus, as the discriminator loss peaks, so too does the sample quality. We used the exact experimental setup in [20]. Details are provided in their open-source implementation [39].

5.3 Stable InfoGANs

WGAN-GP enables stable training and the use of a wide variety of architectures, including deep ResNets which can generate extremely sharp samples. However, just like the original GAN, WGAN-GP places no constraint on how the generator may use the input noise vector z. As a result, individual dimensions of z may not correspond to semantic features of the data if the generator uses z in a highly-entangled way. Section 5.3.1 outlines the transition from WGAN-GP to InfoGAN-GP. Section 5.3.2 details the subsequent hyperparameter searches which were carried out. Finally, section 5.3.3 details further stability improvements.

5.3.1 WGAN-GP to InfoWGAN-GP

In order to learn disentangled representations with WGAN-GP, we needed to apply the same modifications that took GANs to InfoGANs in [11]. As discussed in section 2.5, these modifications allow a mutual information regularization term to be added to the value function, encouraging the network to learn more interpretable and meaningful representations.

The first step was to split the noise vector into 'incompressible' noise $\boldsymbol{z}$ and latent codes $\boldsymbol{c}$, subsequently ensuring that the latent codes were sampled from the correct discrete/continuous prior distributions. The concatenated noise vector is then used to generate samples $G(\boldsymbol{z}, \boldsymbol{c})$. The next step was to implement the auxiliary network $Q(\boldsymbol{c}|\boldsymbol{x})$ which approximates $P(\boldsymbol{c}|\boldsymbol{x})$. Like InfoGAN, we ensure that the discriminator D shares all convolutional layers with Q, with both networks having their own fully-connected output layer. The final step was to add the mutual information term to the WGAN-GP value function. As detailed in section 2.5, this term is approximated by the variational lower bound given in eq. 2.5. Algorithm 1 illustrates the practical steps required to approximate this lower bound for continuous latent codes with Monte Carlo simulation. [Worked only with continuous latent codes]. Note that a similar procedure can be applied for discrete codes, with $Q(\boldsymbol{x})$ instead returning the parameters of a categorical distribution which can be passed to a CATEGORICAL function analogous to the GAUSSIAN function. Adding this variational lower bound of the mutual information to the WGAN-GP value function, we arrive at the new value function:

$$\min_{G,Q} \max_{D} V_{InfoWGAN_{GP}}(D, G, Q) = V_{WGAN_{GP}}(D, G) - \lambda_{mi} L_I(G, Q). \tag{5.1}$$

Initial experiments indicated that the unbounded loss of the WGAN required a larger value for mutual information coefficient λ_{mi} (λ_{gp} now refers to the gradient penalty coefficient) in order to ensure that the mutual information cost was on the same scale as the WGAN objectives. Setting $\lambda_{mi} = 10$ was sufficient, with the network appearing to successfully disentangle the factors of variation in the data with all other hyperparameters set to their respective default values (see Appendix B.1 for further details on the experimental setup). The disentanglement was visually assessed in a similar manner to [11], individually varying each latent code from -1 to 1 and inspecting the resulting images for different types of semantic variation. These images are given in figure 5.3. Note that the factors do not appear to be completely disentangled, with c_2 and c_4 both

capturing a combination of azimuth and the red colour channel. The effect of λ_{mi} and other hyperparameters is further investigated in the next section.

Algorithm 1 Approximating the variational lower bound on mutual information (eq. 2.5) for continuous latent codes with Monte Carlo simulation. Based on the procedure in [11].

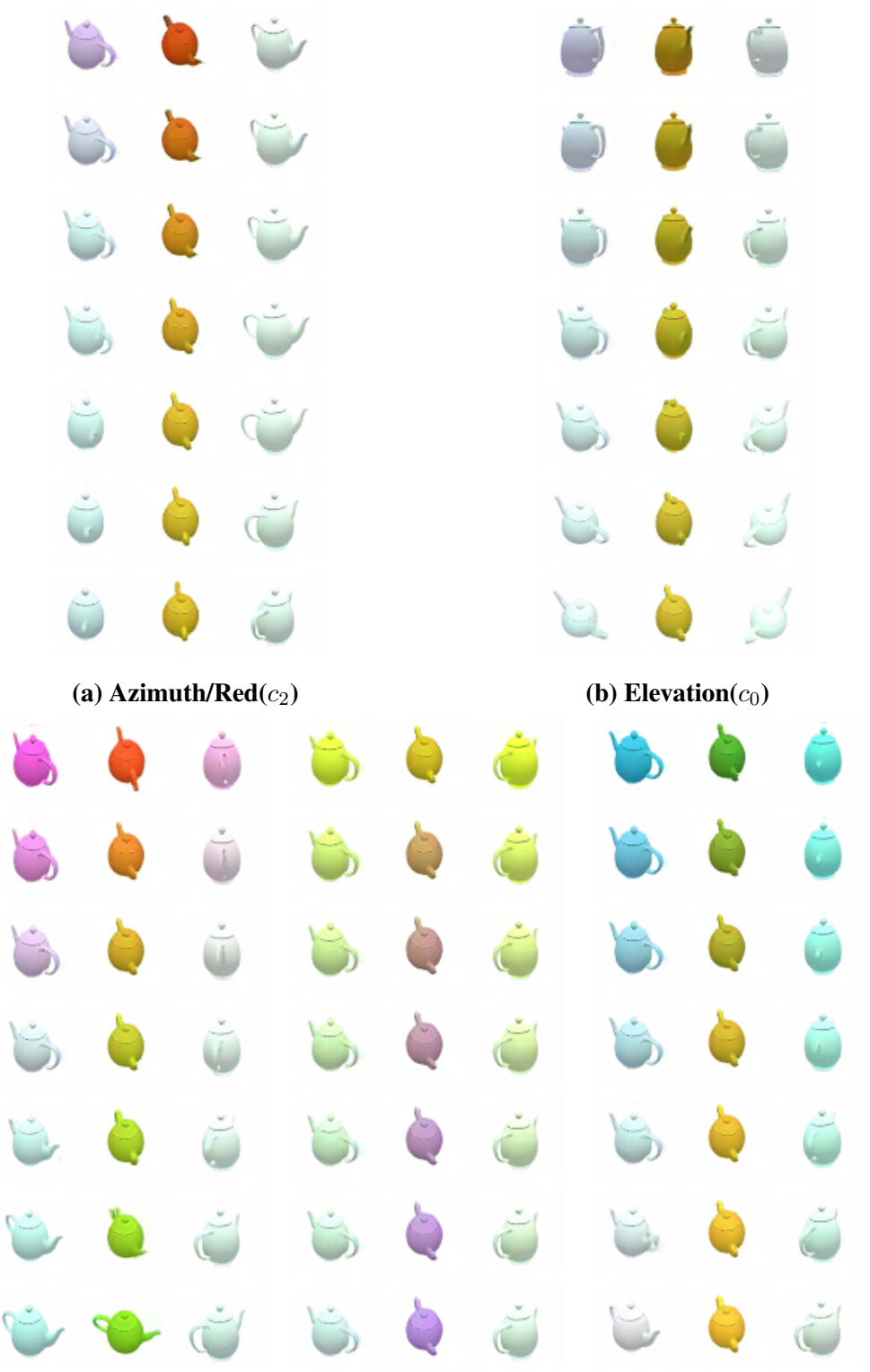
Require:

Batch of N continuous latent codes $\boldsymbol{c}$, batch of N images $\boldsymbol{x}$, neural network Q which parametrizes the auxiliary distribution $Q(\boldsymbol{c}|\boldsymbol{x})$ and function GAUSSIAN($\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\sigma}$) which returns $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \sum)$, where $\sum_{ij} = \boldsymbol{\delta}_{ij}\boldsymbol{\sigma}_i$, $\mathcal{N}$ denotes the PDF of a multivariate Gaussian. Note that $\boldsymbol{\delta}_{ij}$ is the Kronecker delta which is zero unless $i = j$.

procedure APPROXIMATE_MI($\boldsymbol{c}, \boldsymbol{x}, Q$)
  $\boldsymbol{\mu}, \boldsymbol{\sigma} \leftarrow Q(\boldsymbol{x})$
  $\boldsymbol{q\_c\_given\_x} \leftarrow$ GAUSSIAN($\boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\sigma}$)
  $\boldsymbol{prior\_c} \leftarrow$ GAUSSIAN($\boldsymbol{c}, \mathbf{0}, \mathbf{1}$)
  $cross\_entropy \leftarrow \frac{1}{N}\sum_{i=1}^{N}(-\log(\boldsymbol{q\_c\_given\_x}_i))$
  $entropy \leftarrow \frac{1}{N}\sum_{i=1}^{N}(-\log(\boldsymbol{prior\_c}_i))$
  $mi \leftarrow -cross\_entropy + entropy$ ▷ $\approx \mathbb{E}_{\boldsymbol{c}\sim P(\boldsymbol{c}),\boldsymbol{x}\sim G(\boldsymbol{z},\boldsymbol{c})}[\log Q(\boldsymbol{c}|\boldsymbol{x})] + H(\boldsymbol{c})$
  return mi
end procedure

5.3.2 Optimization and exploration

- **Mutual information coefficient $\boldsymbol{\lambda_{mi}}$.** To discover the effect of λ_{mi} on disentanglement, we compared the value of mutual information lower bound (eq. 2.5) over iterations, the sample quality and the visual disentanglement for different settings of λ_{mi}. Intuitively, an underestimate would result in tangled representations while an overestimate would prevent the networks from learning the data distribution. Experimental results supported these intuitions, with $\lambda_{mi} = 1$ resulting in lower mutual information than $\lambda_{mi} = 10$ (see figure 5.4) and $\lambda_{mi} = 25$ resulting in blurry samples (see Appendix B.4). Further hyperparameter searches were carried out in an attempt to find a value for λ_{mi} that completely disentangled the factors of variation. However, we found that the (visual) degree of disentanglement was more dependent on the random initialization than the value of λ_{mi}, as discussed in the final point below.



(a) Azimuth/Red(c_2) **(b) Elevation(c_0)**

(c) Red/Azimuth(c_4) **(d) Green(c_1)** **(e) Blue/Azimuth(c_3)**

Figure 5.3: Manipulating the latent codes. Each column in the subfigures illustrates how the learnt code affects different random samples. Learned continuous latent codes are varied from -1 (bottom) to 1 (top) to show the effect on generated images. The model appears to have successfully disentangled (b) elevation and (d) green. However, azimuth and red are clearly tangled in (a) and (c). Although less obvious, azimuth and blue are also tangled in (e). The colour captured by a given latent code is determined by examining the colour channel which needs to be changed to produce each teapot in a column.

- **Architecture of the output layer(s) for $Q(\boldsymbol{c}|\boldsymbol{x})$.** D and Q share all convolution layers, but some experiments in [11] used multiple (nonlinear) FC output layers for Q. However, we found that additional FC layers in the output layer of Q did not improve the networks ability to disentangle the factors of variation. Details of the architectures experimented with are provided in Appendix B.2.

- **Unfixing the standard deviation of continuous latent codes.** For continuous latent codes, $Q(\boldsymbol{c}|\boldsymbol{x})$ outputs the parameters of a Gaussian distribution, i.e. the mean and standard deviation. Although it is not explicit in [11], their published code [38] indicates that the standard deviations of continuous latent codes were fixed to 1. We unfixed these standard deviations to determine its effect on the models ability to disentangle the generative factors. Experiments indicated that this allowed the network to 'cheat', driving the standard deviations up to extremely large values to achieve high mutual information. However, the constantly increasing mutual information prevented the model from learning the data distribution. Perhaps first training the network with a fixed standard deviation and later freeing the standard deviation would be a more appropriate approach. We leave this investigation to future work.

- **Random initializations.** Chen et al. [11] only presented the representations which most resembled prior supervised results after training the InfoGAN five times with different random initializations. This perhaps indicates a sensitivity to random initialization when using the original GAN value function. We found that a similar sensitivity exists with the WGAN-GP value function, with different random initializations achieving varying levels of disentanglement (samples for multiple runs are provided in Appendix B.1). Furthermore, we found that the degree of disentanglement was more dependent on the random initialization than specific hyperparameter values. However, choosing one of many random runs by visually inspecting the degree of disentanglement requires human supervision. This prompted an investigation into ways of reducing this severe sensitivity in the next section, where we aimed to take another step towards black-box disentanglement.

5.3.3 Further improving stability

- **Correlation penalty.** As depicted in Appendix B.1, most random runs did not result in completely disentangled latent codes, with some generative factors captured by a combination of latent codes. Intuitively, if several latent codes capture the same

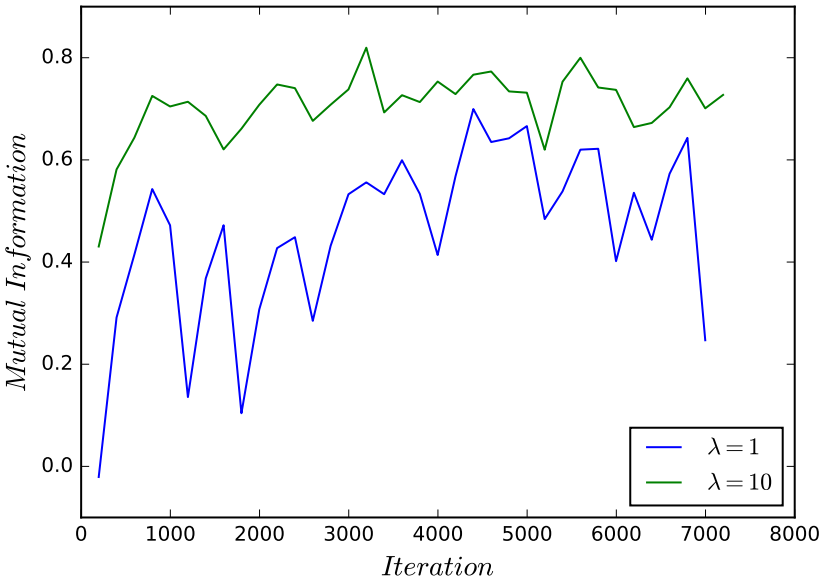


Figure 5.4: Effect of mutual information coefficient λ_{mi}.

generative factors they may be correlated. Furthermore, maximizing the mutual information does not explicitly penalize correlated representations, although such representations would ultimately be able to capture less information about the image. Hence, we aimed to encourage the model to learn more disentangled latent codes by discouraging correlated representations with a correlation penalty. The penalty was calculated using the standardized covariance matrix, summing the absolute values of the off-diagonal elements in a similar manner to the Pearson correlation coefficient [41]. Although this penalty reduced the correlation between the latent codes (see figure 5.5), the degree of disentanglement achieved by the model was still primarily dependent on the random initialization.

- **Updated gradient penalty.** The WGAN-GP improved the stability of the WGAN by replacing the weight clipping with a gradient penalty. This penalty ensured that the discriminator had 'gradients with norm at most 1 everywhere', thus enforcing the necessary Lipschitz constraint on the discriminator. We hypothesize that subsequently adding a mutual information cost to the discriminator adversely affects this required constraint, allowing the discriminator to lie outside of the set of 1-Lipschitz functions. Furthermore, we hypothesize that this is the source of the observed instability and unreliable disentanglement which culminates in a sensitivity to random initialization. Thus, to place the discriminator back in the space of 1-Lipschitz functions, we add the per-image mutual information $L_{\hat{I}}(Q)$ to the per-image discrimina-

tor cost $D(\hat{\boldsymbol{x}})$ *before* calculating the gradient norm with respect to its input. $L_{\hat{I}}(Q)$ is defined as:

$$L_{\hat{I}}(Q) = \log Q(\boldsymbol{c}|\hat{\boldsymbol{x}}) - \log P(\boldsymbol{c}), \tag{5.2}$$

where $\hat{\boldsymbol{x}} \sim P(\hat{\boldsymbol{x}})$ and $\boldsymbol{c} \sim P(\boldsymbol{c})$ as before, with the sampling distribution $P(\hat{\boldsymbol{x}})$ implicitly defined by equation 2.4. Thus, the updated gradient penalty is

$$GP_{updated} = \mathbb{E}_{\hat{\boldsymbol{x}}\sim P(\hat{\boldsymbol{x}})}[(\left\|\nabla_{\hat{\boldsymbol{x}}}[D(\hat{\boldsymbol{x}}) - \lambda_{mi} L_{\hat{I}}(Q)]\right\|_2 - 1)^2]. \tag{5.3}$$

This updated gradient penalty ensures that the gradients take the added mutual information 'into account', correctly enforcing the required Lipschitz constraint on the discriminator. Putting all of these equations together, we can define the new value function:

$$\min_{G,Q} \max_{D} V_{InfoWGAN_{GP}}(D, G, Q) = V_{WGAN}(D, G) - \lambda_{mi} L_I(G, Q) + \lambda_{gp} GP_{updated}. \tag{5.4}$$

Algorithm 2 describes the the new training procedure. The updated penalty can be viewed as transitioning from Info(WGAN-GP) to (InfoWGAN)-GP. This simple update is remarkably effective, allowing the model to achieve stable and reliable disentanglement. Several random runs produced comparable results, with each run appearing to successfully disentangle all of the factors of variation in the data. The latent representations for one such run are given in figure 5.6, where a clear improvement can be seen over the degree of disentanglement achieved with the old penalty (see figure =5.3). In addition, figure 5.5 shows that the correlation between the latent representations is significantly reduced with this updated penalty. Together, these results support the hypothesis that the previously-illustrated instability stemmed from an inaccurate gradient penalty. This improvement in stability will be quantitatively evaluated in Chapter 6.

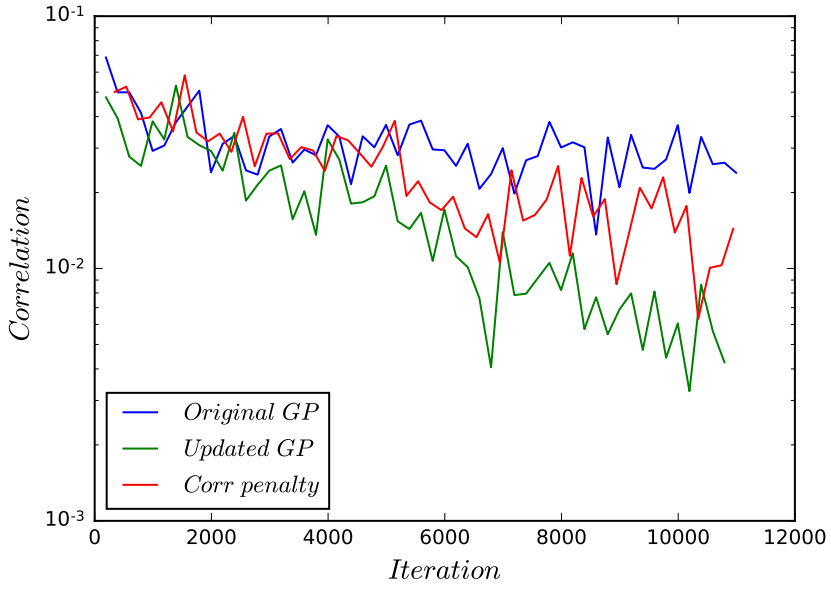
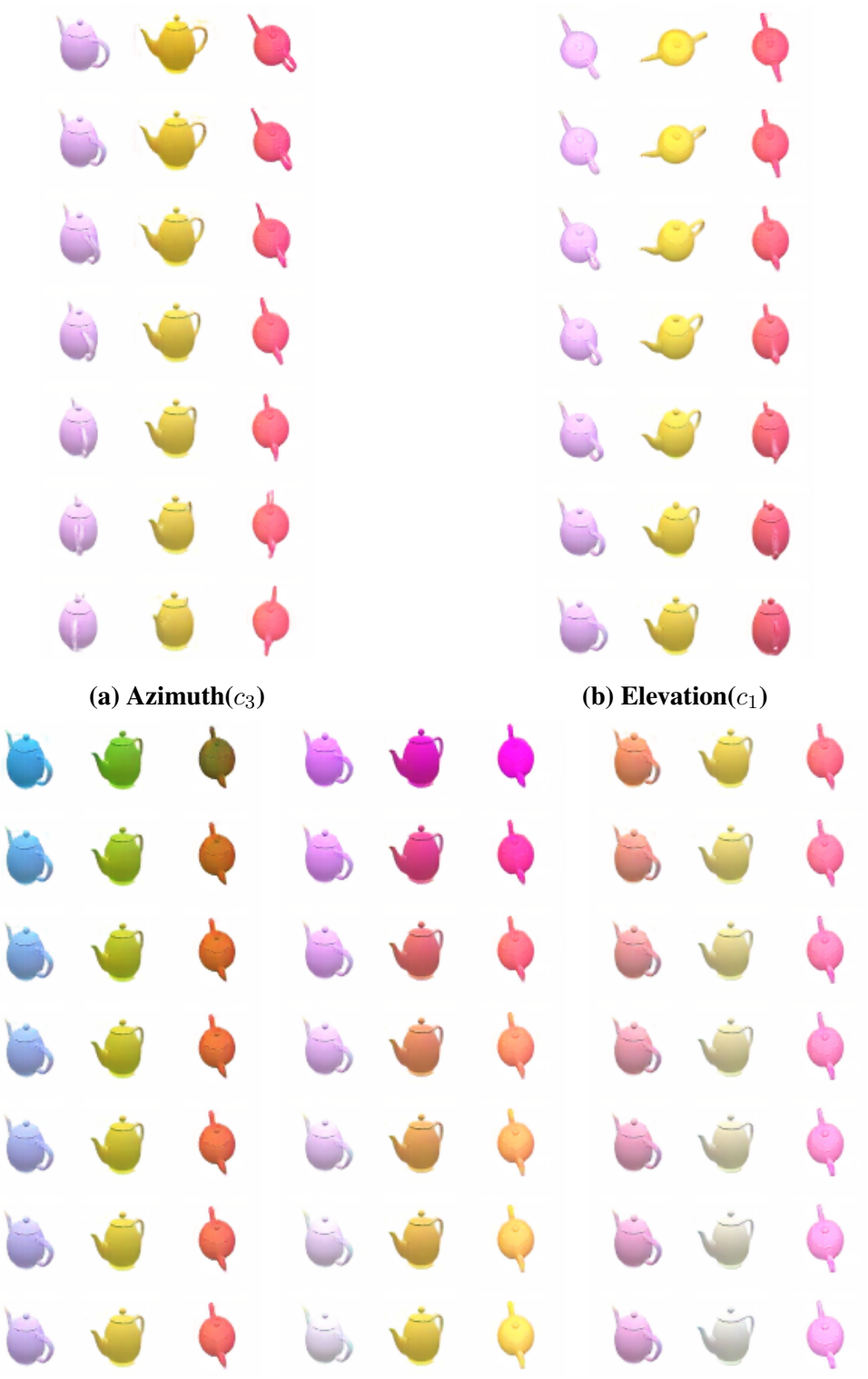


Figure 5.5: Batch correlation between latent codes over iterations. Correlation value is equal to the sum the off-diagonal elements of the standardized covariance matrix.



(a) Azimuth(c_3) **(b) Elevation(c_1)**

(c) Red(c_2) **(d) Green(c_0)** **(e) Blue(c_4)**

Figure 5.6: Manipulating the latent codes. Each column in the subfigures illustrates how the learnt code affects different random samples. Learned continuous latent codes are varied from -1 (bottom) to 1 (top) to show the effect on generated images. The model appears to have successfully disentangled all the factors of variation in the data and learned to smoothly interpolate between them. The colour captured by a given latent code is determined by examining the colour channel which needs to be changed to produce each teapot in a column.

Algorithm 2 InfoWGAN with updated gradient penalty. Based on [20] and [11]. Default values of $\lambda_{gp} = 10$, $\lambda_{mi} = 10$, $n_{critic} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$, PER_IMAGE_APPROXIMATE_MI is analogous to APPROXIMATE_MI in algorithm 1 without the summations in steps 5 and 6, thus returning a vector $[m_i, \ldots, m_N]$, where m_i denotes the approximate mutual information between latent codes $\boldsymbol{c}_i$ and image $\boldsymbol{x}_i$.

Require:

The gradient penalty coefficient λ_{gp}, the mutual information coefficient λ_{mi}, the number of critic iterations per generator iteration n_{critic}, the batch size N, Adam hyperparameters α, β_1, β_2, functions APPROXIMATE_MI and PER_IMAGE_APPROXIMATE_MI.

Require:

Initial critic parameters $\boldsymbol{w}_0$, initial generator parameters $\boldsymbol{\theta}_0$, initial auxiliary network parameters $\boldsymbol{q}_0$.

1: **while** $\boldsymbol{\theta}$ has not converged **do**
2: **for** $t = 1, \ldots, n_{critic}$ **do**
3: **for** $i = 1, \ldots, N$ **do**
4: Sample real data $\boldsymbol{x} \sim P_{data}$, noise variables $\boldsymbol{z} \sim p(\boldsymbol{z})$ and $\boldsymbol{c} \sim p(\boldsymbol{c})$, a random number $\epsilon \sim U[0, 1]$.
5: $\tilde{\boldsymbol{x}} \leftarrow G_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{c})$
6: $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
7: $mi_{\tilde{x}} \leftarrow$ APPROXIMATE_MI$(\boldsymbol{c}, \tilde{\boldsymbol{x}}, Q_{\boldsymbol{q}})$
8: $\boldsymbol{mi}_{\hat{x}} \leftarrow$ PER_IMAGE_APPROXIMATE_MI$(\boldsymbol{c}, \hat{\boldsymbol{x}}, Q_{\boldsymbol{q}})$
9: $gp \leftarrow (\left\| \nabla_{\hat{\boldsymbol{x}}}[D_{\boldsymbol{w}}(\hat{\boldsymbol{x}}) - \lambda_{mi}(\boldsymbol{mi}_{\hat{x}})] \right\|_2 - 1)^2$
10: $L^{(i)} \leftarrow D_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}) - D_{\boldsymbol{w}}(\boldsymbol{x}) - \lambda_{mi} mi_{\tilde{x}} + \lambda_{gp} gp$
11: **end for**
12: $\boldsymbol{w} \leftarrow Adam(\nabla_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} L^{(i)}, \boldsymbol{w}, \alpha, \beta_1, \beta_2)$
13: $\boldsymbol{q} \leftarrow Adam(\nabla_{\boldsymbol{q}} \frac{1}{N} \sum_{i=1}^{N} L^{(i)}, \boldsymbol{q}, \alpha, \beta_1, \beta_2)$
14: **end for**
15: Sample batch of noise variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{N} \sim p(\boldsymbol{z})$ and $\{\boldsymbol{c}^{(i)}\}_{i=1}^{N} \sim p(\boldsymbol{c})$.
16: $\tilde{\boldsymbol{x}} \leftarrow G_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{c})$
17: $mi_{\tilde{x}} \leftarrow$ APPROXIMATE_MI$(\boldsymbol{c}, \tilde{\boldsymbol{x}}, Q_{\boldsymbol{q}})$
18: $\boldsymbol{\theta} \leftarrow Adam(\nabla_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} -(D_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})) + \lambda_{mi} mi_{\tilde{x}}, \theta, \alpha, \beta_1, \beta_2)$
19: **end while**

Chapter 6

Evaluating the Quality of Learnt Representations

In chapter 2, we presented InfoGAN, a deep unsupervised generative model that appears to disentangle the underlying factors of variation in data to learn meaningful and highly semantic representations. Unfortunately, the degree of disentanglement achieved by this promising model was only evaluated qualitatively (by visual inspection) in [11].

In chapter 3, we reviewed some related works that seek to learn disentangled representations in an unsupervised manner, along with some approaches to quantifying the degree of disentangled achieved by different models.

In chapter 4, we detailed the software and hardware that would later facilitate the training and quantitative analysis of InfoGAN. Chapter 4 also presented the computer-graphics generated dataset, highlighting the five independent generative factors for which we have access to the underlying ground-truth values.

In chapter 5, we illustrated the instability of the 'vanilla' GAN learning objective, before simplifying the dataset and employing more stable GANs based on the Wasserstein distance. In particular, we detailed the process of adding the mutual information cost of InfoGAN to the stable value function of the improved WGAN (WGAN-GP). With the help of an updated gradient penalty, this new combined generative model appeared to achieve stable and reliable disentanglement. However, this only brought us to the same level of qualitative analysis carried out in [11].

To gain a conceptual understanding of our world, models must first learn to understand the factorial structure of low-level sensory input without supervision. As argued by several notable works [8, 9, 6], this can only be accomplished if the model learns to disentangle the underlying explanatory factors hidden in the observed data. As discussed in chapter 1, this motivates a deeper quantitative analysis of InfoGAN and ultimately leads to the main objective of this dissertation—to truly elucidate the quality of the latent variables discovered by InfoGAN. This will be the focus of chapter 6.

In section 6.1, we propose a new metric to quantify the degree of disentanglement achieved by different models, using it to quantify the degree of disentanglement achieved by InfoGAN on our teapot dataset. Section 6.2 further evaluates the quality of the latent variables discovered by InfoGAN, quantifying the models ability to reason about new data with unseen factor combinations by recombining previously-learnt factors.

6.1 Quantifying the degree of disentanglement

In this section, we propose a new metric to quantify the degree of disentanglement in learnt latent representations and employ it to quantify the degree of disentanglement achieved by InfoGAN. To verify the propriety of our proposed metric, we also present several alternative metrics, including those based on higher-capacity models and those which have built-in methods to quantify the degree of *separation* within the learnt latent representations. A detailed description and comparison of these metrics in provided in sections 6.1.1- 6.1.5, before analysing their results and respective strengths in section 6.1.6.

6.1.1 Metric

As discussed in chapter 1, a disentangled representation is one that separates the factors of variation, explicitly representing the important attributes of the data. We can decompose this definition into two characteristics of disentangled representations:

- *Separation*: ability to separate the factors of variation, with learnt latent variables capturing at most one statistically independent generative factor.
- *Explicit representation*: ability to explicitly represent the important attributes of the data, allowing information to be easily extracted.

Armed with these two defining characteristics, we devise a metric to quantitatively approximate the degree of disentanglement within learnt latent representations. Our metric uses K linear regressors to predict the ground-truth value of K generative factors given the latent representation. In essence, each regressor learns the mapping $f(\boldsymbol{c}) : \mathbb{R}^D \rightarrow \mathbb{R}^1$, where $\boldsymbol{c}$ is the latent representation and D is its dimensionality. The model's low expressive power ensures that low prediction error can only be achieved if the generative factors have already been disentangled in latent space. More specifically, low prediction error can only be achieved if the information is easily extractable, thus quantifying the models ability to capture and explicitly represent the important attributes of the data (*explicit representation*). To ensure our metric also evaluates a models ability to *separate* the factors of variation, we apply an $l1$ regularization penalty. If $D \leqslant K$, as in our case, this regularization encourages a sparse one-to-one mapping between the learnt factors and the generative factors. If $D > K$, a sparse many-to-one mapping is encouraged, where multiple learnt factors may contribute to a given prediction. Either way, the magnitude of the resulting regression weights should rank the latent variables in order of relative importance to the prediction. Note, this is assuming that the inputs and targets have been normalized to have zero mean and unit variance (see section 6.1.4). The degree of separation can then be qualitatively evaluated by visualizing the magnitude of the weights with a Hinton diagram (see section 6.1.6). To quantitatively evaluate the degree of separation, we devise a new 'separation score' based on the principle that each latent factor should be important for predicting at most one generative factor if the factors have been successfully disentangled. The average separation score (ASS) across all D latent variables is defined as:

$$\frac{1}{D}\sum_{i=0}^{D}\left(\frac{\max(|\boldsymbol{W}_i|) - \min(|\boldsymbol{W}_i|)}{\sum_{j=0}^{K}|\boldsymbol{W}_{ij}|}\right), \tag{6.1}$$

where $|\boldsymbol{W}_{ij}|$ denotes the magnitude of the weight used to scale latent factor i when predicting generative factor j and $\boldsymbol{W}_i$ denotes the regression inputs, i.e. the latent variables for a given prediction. If the latent factor is important for predicting a single generative factor, the score will be 1. If the latent factor is equally important for predicting all generative factors, the score will be 0. If multiple latent variables capture overlapping or similar information about a given generative factor, the score may be similar to the correlation coefficient. The ASS could also be used in a different manner in order to reveal the 'purity' of each prediction, measuring the number of latent variables which capture a generative factor rather than the number of generative factors captured by a latent variable. Although this alternative interpretation may seem more

natural when dealing with an equal number of latent variables and generative factors, purity is not desirable in learnt representations where a generative factor is captured by more than one latent variable. With Higgins at al. [6] demonstrating the tenancy of unsupervised generative models to learn such representations, we believe that it is more useful to quantify separation than purity.

To summarise our metric:

- The prediction error of the linear regressor quantifies the amount of easily-extractable information captured by the latent representation.
- The magnitude of the weights allow the degree of separation to be quantified using the separation score.
- Together, the prediction error and weight magnitudes allow our metric to quantify the degree of disentanglement achieved by a given model.

It should be noted that Higgins et al. also use linear regression in [6]. However, in contrast to their 'factor change classification' approach, our metric:

- **Is simple and intuitive.** The is no subtraction of rather arbitrary 'start' and 'end' latent representations. There is no need for multiple random runs or to discard the worst results (bottom 50% discarded in [6]).
- **Requires no additional data.** There is no need to generate new 'factor change' data. Only the regression inputs (learnt latent representations) and targets (ground-truth factor values) are needed.
- **Naturally generalizes to high-dimensional factors of variation.** Taking the absolute change along each dimension may not be the best way to capture the total change for a high-dimensional factor of variation. For example, if a factor is sampled from a 10D Gaussian, spherical distance from the origin may be a more appropriate measure of total change. We don't subtract any values, so don't have to worry about such issues.
- **Quantifies the amount of information captured.** Disentangled representations are likely to be useful for downstream tasks if they disentangle as many factors of variation as possible, discarding as little information as possible [9]. Good regression performance is not possible without capturing a significant amount of information about the generative factors, with the predicition error quantifying this amount. In

contrast, much less information about the generative factors is required to identify which generative factor has been altered. Furthermore, classification accuracy is more difficult to interpret as a measure of information captured by the latent representation.

- **Quantifies the degree of *separation*.** The magnitude of the regression weights rank the latent variables in order of relative importance to the prediction. Each latent factor should be important for predicting at most one generative factor. Thus, the degree of separation can be quantified using equation 6.1.

6.1.2 Retrieving the latent codes

In order to use our new metric and quantify the degree of disentanglement achieved by InfoGAN, we first needed to retrieve the *most likely* latent representations for each image $\boldsymbol{x}$ in our dataset. As a reminder, Q parametrizes the auxiliary distribution $Q(\boldsymbol{c}|\boldsymbol{x})$, with $Q(\boldsymbol{x})$ returning the parameters of the posterior distribution over latent codes (the mean and standard deviation of a Gaussian for our continuous latent codes). This mean represents the most likely latent representation for a given image under the learned auxiliary distribution $Q(\boldsymbol{c}|\boldsymbol{x})$. Thus, to retrieve the most likely latent codes for all images in the dataset, we propagated them through the trained network Q, discarding the standard deviation.

6.1.3 Baselines

As with any new metric, baselines are need to put the performance of a given model into context or perspective. We chose to use the popular matrix decompositions outlined below as simple but intuitive baselines. Both methods seek a set of feature vectors (i.e. a basis) which can be linearly-combined to reach any point in the dataset. Intuitively, these pixel-level decompositions should not be able to reasonably estimate abstract generative factors such as the pose of the teapot. However, as illustrated in section 6.1.6, they turn out to be surprisingly competitive baselines, explaining a large amount of the variance in our dataset. This is likely due to the centring and lack of background scene which allow pixel-wise variations to reveal a great deal about these abstract factors. Note, we chose to use the same number of features vectors (or components) as latent codes (5). Both methods were implemented using sci-kit learn.

- **Principal Components Analysis (PCA).** PCA seeks a basis that best explains the variability in the dataset. To do so, it decomposes the dataset into a set of linearly uncorrelated features (principal components), ordered by the amount of variance that they explain. PCA is often used as a dimensionality reduction method, taking the top N features which explain most of the variance in the data. Hence, it serves as an intuitive 'disentanglement baseline', seeking to capture 5 linearly uncorrelated features which best explain the variance of data generated by 5 independent factors. Given the size of the data ($100000 \times 64 \times 64 \times 3$), the sci-kit learn method for PCA uses the efficient randomized method of Halko et al. [42] to compute the the singular value decomposition (SVD).

- **Independent Components Analysis (ICA).** ICA seeks a basis in which each feature vector is an independent component of the dataset. Hence, ICA is often used to separate a mixed signal into independent source components. This serves as an intuitive baseline given that the images were generated by 5 independent components. The sci-kit learn implementation used, *FastICA*, is based on [43].

6.1.4 Preparing the data

For regression inputs, we have the five latent codes (LC), the five principal components (PC) and the five independent components (IC). Each set of inputs is paired with a set of targets, i.e. the ground-truth values of the generative factors. These ground-truths (GT) were loaded from a file which was created during the generation process. To calculate the absolute azimuth of the teapot, we subtracted the camera azimuth from the object azimuth.

The regression inputs were all normalized by subtracting the mean and dividing by the standard deviation. This is a standard pre-processing technique which improves the efficiency of learning in most models by ensuring that all features are in the same range and on the same scale. This also allows default model (hyper)parameters to be used, but more importantly for our metric, it ensures that the magnitude of the learned weights are comparable.

The regression targets were also normalized in the same manner, except for the azimuth. The azimuth is not normalized as it is later interpreted as an angle to deal with wraparound, that is, the fact that there is just a 1 degree difference between 359 degrees

and 0 degrees. Finally, the data was divided into training (60000), validation (20000) and test sets (20000).

6.1.5 Fitting the regression models

With the data prepared, the next step was the fit the regression models. Along with the proposed linear regressor, we fit several high-capacity regressors and those which have built-in methods to quantify the relative importance of each feature. This later allows us to verify the appropriateness of our method by comparing the results of each regressor. To evaluate the models, we use the mean squared error (MSE):

$$L_{MSE} = \frac{1}{N}\sum_{i=0}^{N}(y^{(i)} - f(\boldsymbol{X}^{(i)}))^2, \tag{6.2}$$

where $y^{(i)}$ is the target for test example i, $\boldsymbol{X}^{(i)}$ is the vector of input data for test example i and f is the regression model that takes an input vector and outputs a prediction. We use the MSE as it is a standard measure for regression problems. As the targets (except azimuth) have been normalized to have unit variance, the MSE is naturally standardized relative to the constant regressor, which guesses the expected value of the targets. Hence, this measurement is actually the standardized mean squared error (SMSE). To standardize the azimuth results, we simply divide by the MSE of the constant regressor, i.e., the variance of the azimuth ground-truths. This standardization makes it easy to interpret the results. For example, if the SMSE of a model is 0.6, then the model has a MSE that is 60% of the constant model (40% lower).

For azimuth, the GT angle cannot be directly used as the regression target due to the previously-mentioned wraparound issue. Instead, we use $\sin(target_{angle})$ and $\cos(target_{angle})$ as regression targets and calculate the residuals as follows:

$$pred_{angle} = \arctan 2(pred_{\sin}, pred_{\cos}), \tag{6.3}$$

$$residuals_{angle} = (target_{angle} - pred_{angle} + \pi) \bmod (2\pi) - \pi, \tag{6.4}$$

where $pred_{\sin}$ is the predicted $\sin(target_{angle})$, $pred_{\cos}$ is the predicted $\cos(target_{angle})$ and $target_{angle}$ is the ground-truth azimuth angle in radians.

We will now describe all of the regression models which are employed, including our proposed metric and alternatives which serve as suitable comparisons.

- **Linear Regression (LR):** Our proposed metric uses linear regression with an $l1$ penalty. Note that this is sometimes called Lasso regression. For linear regression, $f(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{w}$ in equation 6.2, with $\boldsymbol{w}$ denoting the regression weights and $\boldsymbol{X}$ denoting the input matrix augmented a with column of ones (i.e. a constant feature) to represent the bias. Also note that the bias is only strictly necessary for the azimuth predictions, whose regression targets are not normalized to have zero mean. However, it is included when predicting all generative factors for uniformity, facilitating dynamic factor prediction. The objective function is thus:

$$\underset{\boldsymbol{w}}{min}\frac{1}{N}\sum_{i=0}^{N}(y^{(i)} - \boldsymbol{X}^{(i)}\boldsymbol{w}^{(i)}))^2 + \alpha|\boldsymbol{w}|\,, \tag{6.5}$$

  where α is the $l1$ regularization coefficient. This model was first implemented in Tensorflow with a single FC layer, adding $\alpha|\boldsymbol{w}|$ to the (squared-error) objective function before passing it to the optimizer. However, we later used the *Lasso* regressor module of sci-kit learn for two reasons. Firstly, its simplified syntax enabled dynamic hyperparameter searches. Secondly, its uniformity with all other sci-kit learn models allowed dynamic model selection. As with all regression models, the hyperparameters (α) are fitted to the validation set.

- **K Nearest Neighbours (KNN):** Given an input $\boldsymbol{x}$, KNN predicts the value of a target variable y by averaging the values of its closest k neighbours. KNN is not 'fitted' to the data, but rather needs access to all the training data at test time. Despite its simplicity, KNN has been successful in a large number of regression (and classification) problems. As KNN is a non-parametric model, it has been most successful in cases where the underlying function is highly non-linear. Thus, we wished to compare the SMSE of this model to that of linear regression in order to reveal how much information has been captured but not completed disentangled (explicitly-represented). We implement KNN using the *KNeighborsRegressor* module of sci-kit learn. We found that the default Euclidean distance was sufficient to determine the nearest k neighbours. We also found that the default uniform averaging was sufficient, allowing each of the k nearest neighbours to contribute uniformly to the prediction (rather than weighted by distance, for example). As is usually the case with KNN, we found that the performance was a monotonic function of k below $k \approx 200$. Hence, for computational reasons and simplicity, we fix $k = 50$ for all inputs rather than fit it to the validation set.

- **Decision Tree (DT):** DTs [44] predict the value of a target variable y by learning

simple decision rules (splits) inferred from the input $\boldsymbol{x}$. These splits can be compared to *if-then-else* decision rules, partitioning the input space to determine the terminal 'cell' in which the input lies. The prediction is then the average of the target variables in this terminal cell, making DTs a piece-wise constant model. The next best split is determined by how much it reduces the impurity of the cell it splits, where impurity is generally measured by the SMSE for regression. The number of times the model chooses to split on a particular input variable indicates its importance to the prediction. Hence, the importance of each feature can be calculated as the number of cases split on that variable over the total number of splits. This natural mechanism for determining feature importances is our main motivation for using DTs, allowing us to determine the degree of disentanglement achieved by InfoGAN as well as the suitability of our proposed metric to do just that. We implement DTs using the *DecisionTreeRegressor* module of sci-kit learn, fitting the max_depth parameter to the validation set. Note that max_depth controls the maximum number of splits allowed and hence the complexity of the fit.

- **Random Forest (RF):** DTs are prone to overfitting as slight variations in the data distribution can result in completely different trees. This problem is alleviated by using decision trees within an ensemble, i.e., a forest. RFs [45] are an ensemble method which average the predictions of several DTs in order to improve generalization. However, simply training many DTs on the same data would result in strongly correlated DTs (perhaps even several identical DTs). Instead, RFs select random samples (with replacement) and fit the DTS to these samples. This 'bootstrapping' method builds independent estimators by showing them different datasets. The average of these independent estimators is generally better than any single estimator as its variance is reduced. RFs can also determine the relative importance of each feature by averaging the features importances from each DT. We implement RFs using the *RandomForestRegressor* module of sci-kit learn, again fitting the max_depth parameter to the validation set. As the performance generally increases with the number of estimators n, we fix $n = 20$ for simplicity and computational efficiency.

- **Gaussian Process (GP):** GPs [46] are distributions over functions which can be used to analytically model complex functions, accurately representing the uncertainty. We use GPs as another flexible regression model, comparing the results to our linear model in order to determine how much information about the generative factors has been captured but not sufficiently disentangled. Viewing the GP as a

distribution over function values $\boldsymbol{f}$, we have:

$$\boldsymbol{f} \sim GP, \tag{6.6}$$

$$p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}; 0, K), \tag{6.7}$$

where $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the kernel and $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ are the inputs at index i and j respectively. For regression, we would like to predict the function values at our test points $\boldsymbol{x}_*$ given some noisy observations of the underlying function (i.e. the regression targets). To do so, we analytically calculate the posterior distribution over function values at our test points $\boldsymbol{f}_*$. As we are only interested in the most likely function values at these test points (rather than the uncertainty of the predictions), we simply take the mean of this posterior distribution $\bar{\boldsymbol{f}}_*$. To implement the GP regressor, we use sci-kit learn's *GaussianProcessRegressor* module, which is based on Algorithm 2.1 in [46]. The default kernel for this module is a squared-exponential (SE) kernel:

$$k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_s^2 exp(-\frac{1}{2l^2}\big\|(\boldsymbol{x}_i - \boldsymbol{x}_j)\big\|^2), \tag{6.8}$$

where l is the length-scale and σ_s^2 is the signal variance. To estimate the noise level of the data, we use this kernel as part of a sum-kernel defined as follows:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \boldsymbol{\delta}_{ij}\sigma_n^2, \tag{6.9}$$

where σ_n^2 is the noise variance, $\boldsymbol{\delta}_{ij}\sigma_n^2$ is a white noise kernel and $\boldsymbol{\delta}_{ij}$ is the Kronecker delta which is zero unless $i = j$. With this sum-kernel, fitting the parameter of the white noise kernel corresponds to estimating the noise level of the data. This allowed us to fit all three kernel parameters (l, σ_s^2, σ_n^2) with maximum likelihood estimation (MLE). An illustration of the log-marginal-likelihood (LML) landscape is provided in figure 6.1. As the complexity of a GP scales cubically with the number of data points, we trained the regressor on a random subset of the data containing 5000 samples.

- **Neural Networks (NN):** Neural networks are a powerful model that can theoretically approximate any function by stacking multiple non-linear layers. Hence, they serve as another good comparison to our metric, providing an estimate of how much information has been captured but not disentangled. All neural networks were implemented with Tensorflow. For uniformity, we fix the architecture (and hence the expressive power) of the network. We found that 3 hidden layers, each containing 50 units, had more than enough expressive power for our purposes. As this flexible

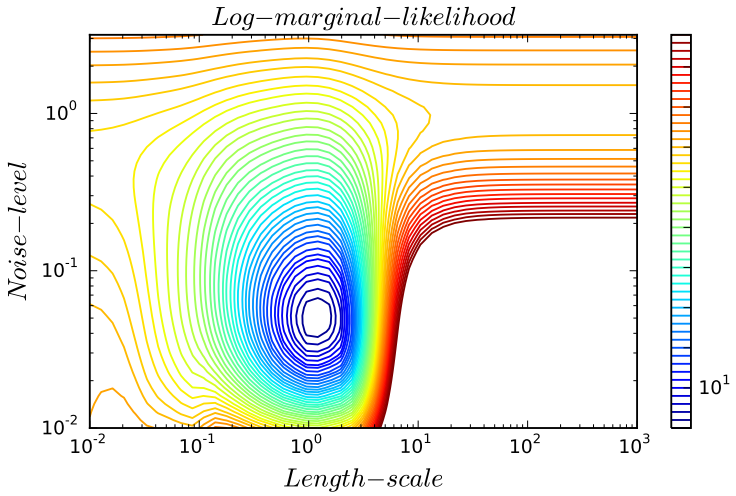


Figure 6.1: Illustration of local minima in the LML landscape. Based on Figure 5.5 in [46]. Depicted is the relationship between the LML, length-scale l and noise-level σ_n^2 (for fixed signal variance $\sigma_s^2 = 1$). There exists two local minima. The top right (shallow) minimum corresponds to a model with a large length-scale and high noise-level, where all variations in the data are explained as noise. The bottom left minimum corresponds to a model with a shorter length-scale and smaller noise-level, explaining most of the variation with a functional relationship.

model was intended to serve as an upper bound on regression performance, the focus was on optimal performance rather than sparsity. Hence, we used an $l2$ penalty, fitting the coefficient β to the validation set.

6.1.6 Results

Table 6.1 presents the test set SMSE for all regression models and inputs. The latent codes learnt by the InfoGAN clearly outperform the baselines, achieving lower SMSE on all regression models when predicting all generative factors. The performance gap between the latent codes and the baselines is most obvious with the low-capacity linear regressor, where generative factors must already be disentangled in latent space in order to achieve good performance. This performance gap decreases on more flexible models like neural networks, where the added flexibility allows the regressor to perform further disentangling, extracting the 'hidden' information. This is clear when comparing the SMSE of the elevation predictions for different inputs. With principal component inputs, the SMSE decreases from 0.54 to 0.15 with the added flexibility of the neural network. In comparison, with latent code inputs, the SMSE decreases from 0.02 to 0.01. A similar comparison exists when comparing the prediction SMSEs for each colour channel, demonstrating that the InfoGAN successfully disentangles these generative factors. The same cannot be said for azimuth, with every regressor having a much higher SMSE when predicting this factor from the latent code inputs. In fact, the error in predicting the azimuth significantly decreases in the higher-capacity models, with the SMSE decreasing from 0.52 to 0.29 between the linear regression and neural network models. This implies that the InfoGAN has struggled to disentangle the azimuth of the teapot, with the added flexibility allowing the azimuth to be further disentangled.

It may be surprising that the baselines, which extract features from the image by examining the pixel-level variations, manage to predict abstract factors like azimuth and elevation so well. For example, the neural network regressor is able to explain 85% of the elevation variance with principal component inputs, and 81% of the elevation variance with independent component inputs. However, this can be attributed to the centring and lack of background scene.

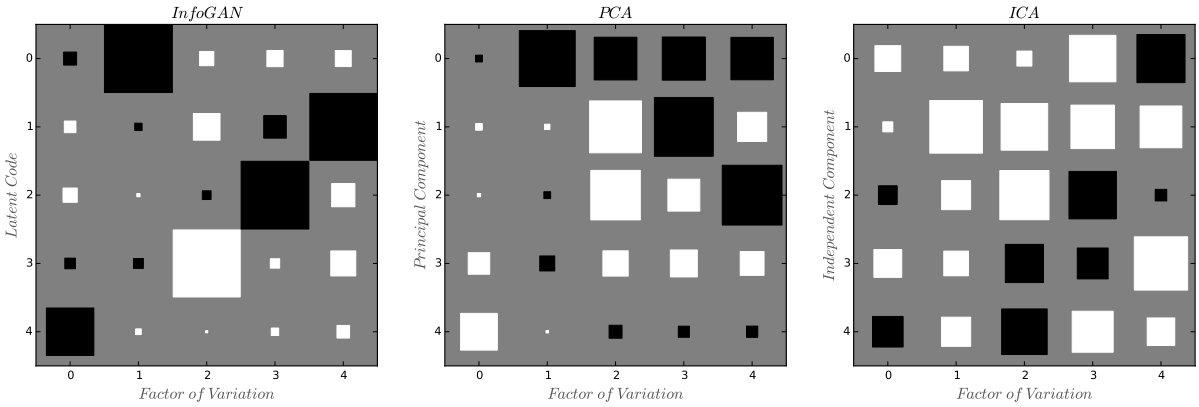
Model	Input	Az.	Elev.	R	G	B
LR	*PC*	0.74	0.54	0.21	0.21	0.21
	IC	0.74	0.54	0.21	0.21	0.21
	LC	0.52	0.02	0.05	0.02	0.05
KNN	*PC*	0.62	0.22	0.09	0.09	0.09
	IC	0.62	0.22	0.09	0.09	0.09
	LC	0.36	0.02	0.03	0.02	0.02
DT	*PC*	0.67	0.26	0.13	0.12	0.12
	IC	0.78	0.32	0.14	0.14	0.14
	LC	0.36	0.01	0.02	0.01	0.02
RF	*PC*	0.55	0.19	0.09	0.09	0.09
	IC	0.67	0.22	0.09	0.09	0.09
	LC	0.31	0.01	0.02	0.01	0.01
GP	*PC*	0.66	0.22	0.09	0.09	0.09
	IC	0.66	0.22	0.09	0.09	0.09
	LC	0.36	0.01	0.02	0.01	0.01
NN	*PC*	0.46	0.15	0.07	0.07	0.07
	IC	0.56	0.19	0.08	0.08	0.08
	LC	0.29	0.01	0.02	0.01	0.01

Table 6.1: Regression results: InfoGAN vs. baselines. Table shows SMSE on the test set. PC denotes principal components, IC denotes independent components and LC denotes latent codes.

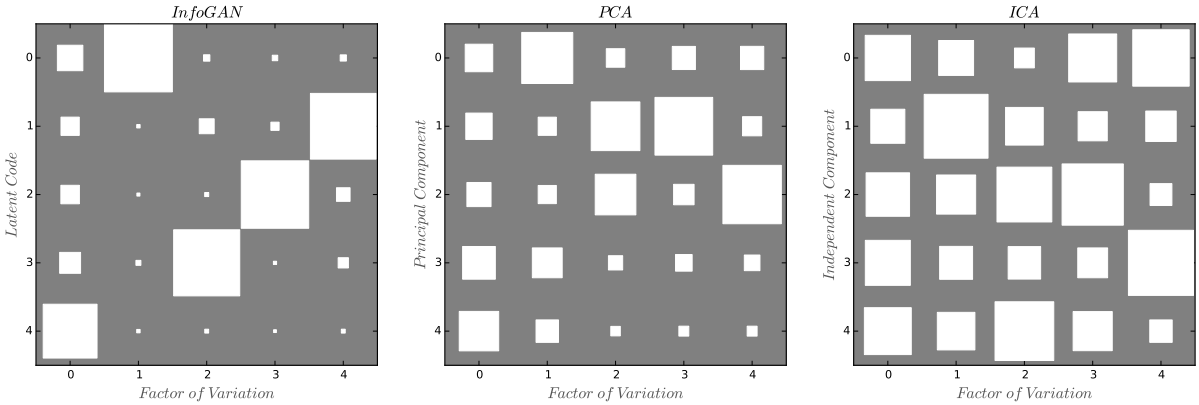
As a reminder, each regressor uses all five latent codes / principal components / independent components to predict the ground-truth value of each generative factor. Hence, a learnt factor or feature may capture information about multiple generative factors. However, as discussed in section 6.1.1, each latent factor in a completely disentangled representation should capture at most one statistically independent generative factor. As a result, we would like to determine the number of generative factor predictions that

a given latent code contributes to (is important for). That is, we would like to determine the degree of separation in the learnt representations. Qualitatively, the degree of separation can be visualized using a Hinton diagram. Hinton diagrams represent positive weights with a white square and negative weights with a black square. The size of the square indicates the magnitude of the weight. For our purposes, we are primarily interested in the magnitude of the weights to determine if there exists a single dominant weight for each learnt factor across all predictions. We visualize the linear regression weights in figure 6.2a and the random forest feature importances in figure 6.2b. Note that the features importances for the decision tree were almost identical to those of the random forest and thus are omitted for brevity. Both diagrams in figure 6.2 show similar patterns, confirming the validity of our proposed metric and the quality of the latent variables discovered by InfoGAN. These Hinton diagrams also indicate that the impressive regression results in table 6.1 were achieved when primarily using a single latent code to predict each generative factor. Although this is also desirable in our case (with an equal number of latent codes and generative factors), this is not to be confused with the degree of separation. The slight discrepancies between figure 6.2a and figure 6.2b are likely due to the flexibility differences of the models. For example, comparing the important principal components for the azimuth prediction across diagrams, one can see that more 'tangled' principal components are able to contribute to the prediction with the increased flexibility of the random forest regressor. Thus, random forest regressors may complement our proposed metric when additional flexibility is required to determine the relative importance of 'tangled' features.

The degree of separation within the learnt representations can also be evaluated quantitatively using the separation score proposed in equation 6.1. Note that this score can be used with both regression weights and feature importances. Table 6.2 presents the separation scores for the latent codes, principal components and independent components, using both the linear regression weights and random forest feature importances. For both linear regressor and random forest, each latent code primarily contributes to the prediction of a single generative factor, as illustrated by the high separation scores. The degree of separation within the latent codes is clearly higher than the baselines.



(a) Linear regression weights.



(b) Random forest feature importances.

Figure 6.2: Visualizing the degree of separation within learnt representations. Positive weights are represented by a white square and negative by a black square. The size of the square indicates the magnitude of the weight. Each row illustrates the importance of a given learnt factor to each prediction. Each column represents the importance of each learnt factor to a given prediction. Hence, a single large weight in each row indicates a high degree of separation. Factors of variation: 0=azimuth, 1=elevation, 2=red, 3=green and 4=blue.

Input	Model	f_0	f_1	f_2	f_3	f_4	Average
PC	*LR*	0.35	0.48	0.50	0.19	0.75	0.46
	RF	0.47	0.42	0.48	0.32	0.62	0.46
IC	*LR*	0.35	0.31	0.38	0.33	0.20	0.32
	RF	0.30	0.38	0.31	0.36	0.32	0.34
LC	*LR*	0.81	0.75	0.85	0.81	0.90	0.82
	RF	0.85	0.86	0.89	0.88	0.98	0.89

Table 6.2: Separation scores: InfoGAN vs. baselines. The average separation score is given for each input, along with the separation scores for each learnt factor (f_0 - f_4). The learnt factors are the latent codes, principal components and independent components for InfoGAN, PCA and ICA respectively.

With the 'most important' latent codes for each prediction now obvious, we investigate the underlying relationship between these important codes and the corresponding generative factor value. To do so, we re-fit each regressor to predict the ground-truth value of a generative factor given the single 'most important' latent code. The resulting fits for each regressor are illustrated in figure 6.3 and figure 6.4. Figure 6.3 illustrates a clear bi-modality between the latent code important for predicting the azimuth and the $\cos(azimuth)$. This bi-modality is also noticeable for the sin(azimuth), although much less pronounced. This bi-modality may occur for a number of reasons. One such reason is the 180 degree ambiguity that exists with our object. That is, the difficulty in distinguishing the spout of the teapot from the handle. Another possibility is that the learnt latent code doesn't actually represent the azimuth angle, but a rather a function of the angle such as $\sin$ or $\cos$. Higgins et al. [6] demonstrated a similar idea, with their VAE learning a disentangled representation in which the rotation is captured by two separate latent variables, one resembling the $\sin(rotation)$, the other resembling the $\cos(rotation)$. It is entirely possible that this is also the most natural way for the InfoGAN to disentangle the azimuth. However, due to the limited number of latent codes used (5 codes for 5 generative factors), the InfoGAN may be forced to settle for a function somewhere in between $\sin$ and $\cos$ in order to maximise the mutual information. This idea is further discussed in section 7.2. Despite the plausibility of these potential sources, further investigation is required to determine the true source of the

bi-modality.

Figures 6.3 and 6.4 also illustrate the linear relationship that exists between the most important latent code and corresponding generative factor, for each generative factor that was successfully disentangled (elevation, red, green and blue). As a result, the fits of the low-capacity linear regressor are almost indistinguishable from the fits of more flexible regressors on these generative factors. The closeness of these fits validates our proposed (linear regression) metric for quantifying the degree of disentanglement achieved by different models. Furthermore, it provides empirical evidence in support of the core idea on which it is based, that is, that good regression performance can only be achieved with this low-capacity model if the generative factors have already been sufficiently disentangled in latent space.

In section 5.3.3 we provided evidence that the updated gradient penalty improved the stability of the InfoGAN, leading to stable and reliable disentanglement. We now further investigate this stability improvement with a quantitative analysis. To do so, we quantify and compare the disentanglement achieved by both penalties. To illustrate the instability or sensitivity of the InfoGAN with the original penalty, we compare the results achieved by a model trained with the updated penalty to those achieved by 4 different random runs with the original penalty. The degree of disentanglement achieved by runs 1, 2, 3 and 4 is visually assessed in figures 5.3, B.1, B.2 and B.3 respectively. Note that two additional runs were carried out with the updated penalty, achieving almost identical results. These runs are omitted to facilitate a clear, uncluttered comparison. Also note that the computational expense of training these deep ResNets prevented the analysis of a sufficient number of random runs to provide a reliable error bar. Thus, all 4 random runs are compared. Table 6.3 presents the linear regression results for these random runs and the updated gradient penalty. Similar trends are observed in the results of all other regressors. These additional results are omitted for clarity, with the linear regression results sufficiently illustrating the difference in stability. The fluctuating SMSEs for InfoGANs trained with the original penalty provides further (quantitative) evidence of the stability introduced by the updated penalty.

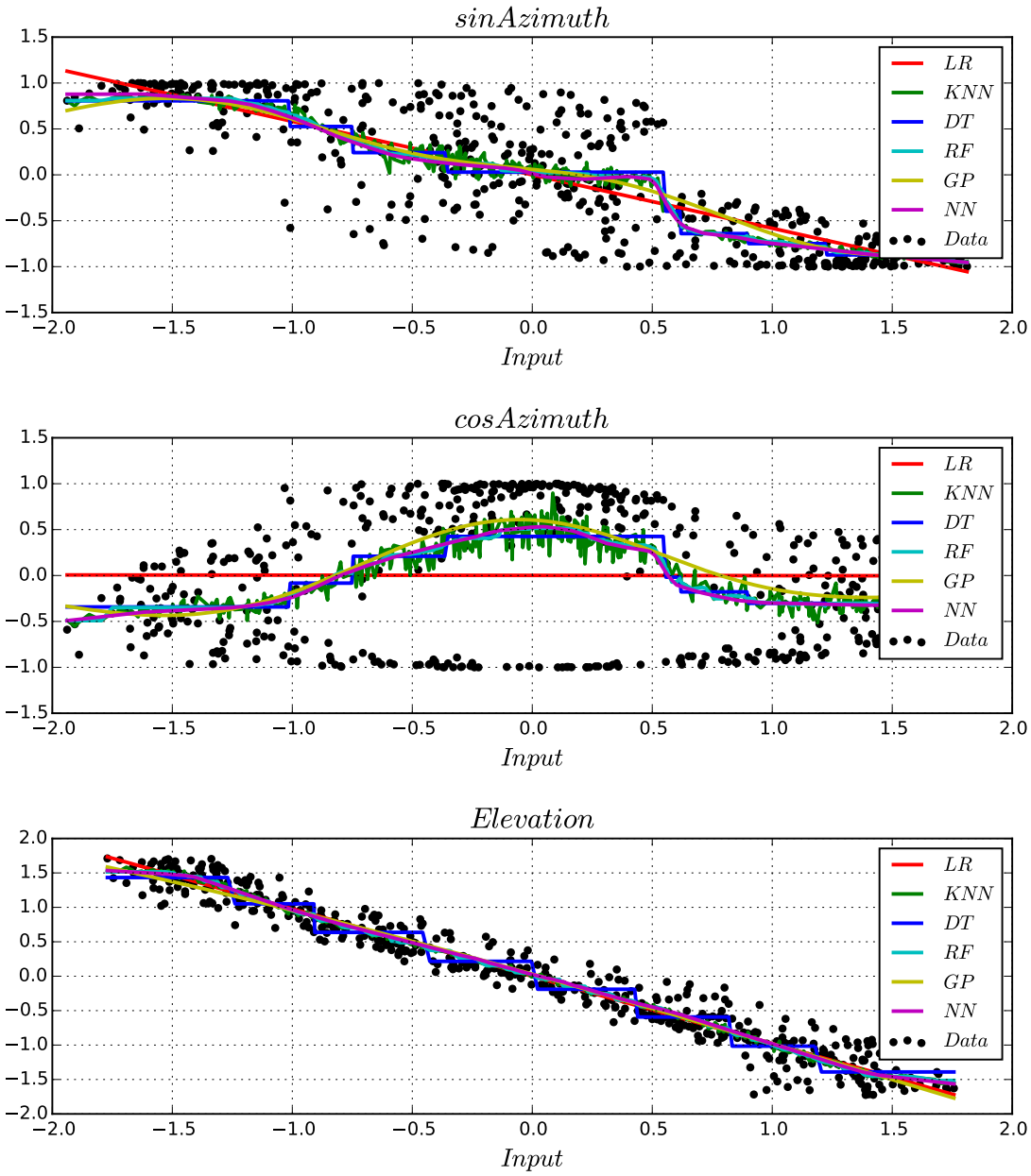


Figure 6.3: Regression fits: azimuth and elevation. A clear bi-modality is visible in the azimuth plots, particularly the $\cos Azimuth$. A clear (negative) linear relationship is illustrated between the elevation and corresponding most important latent code.

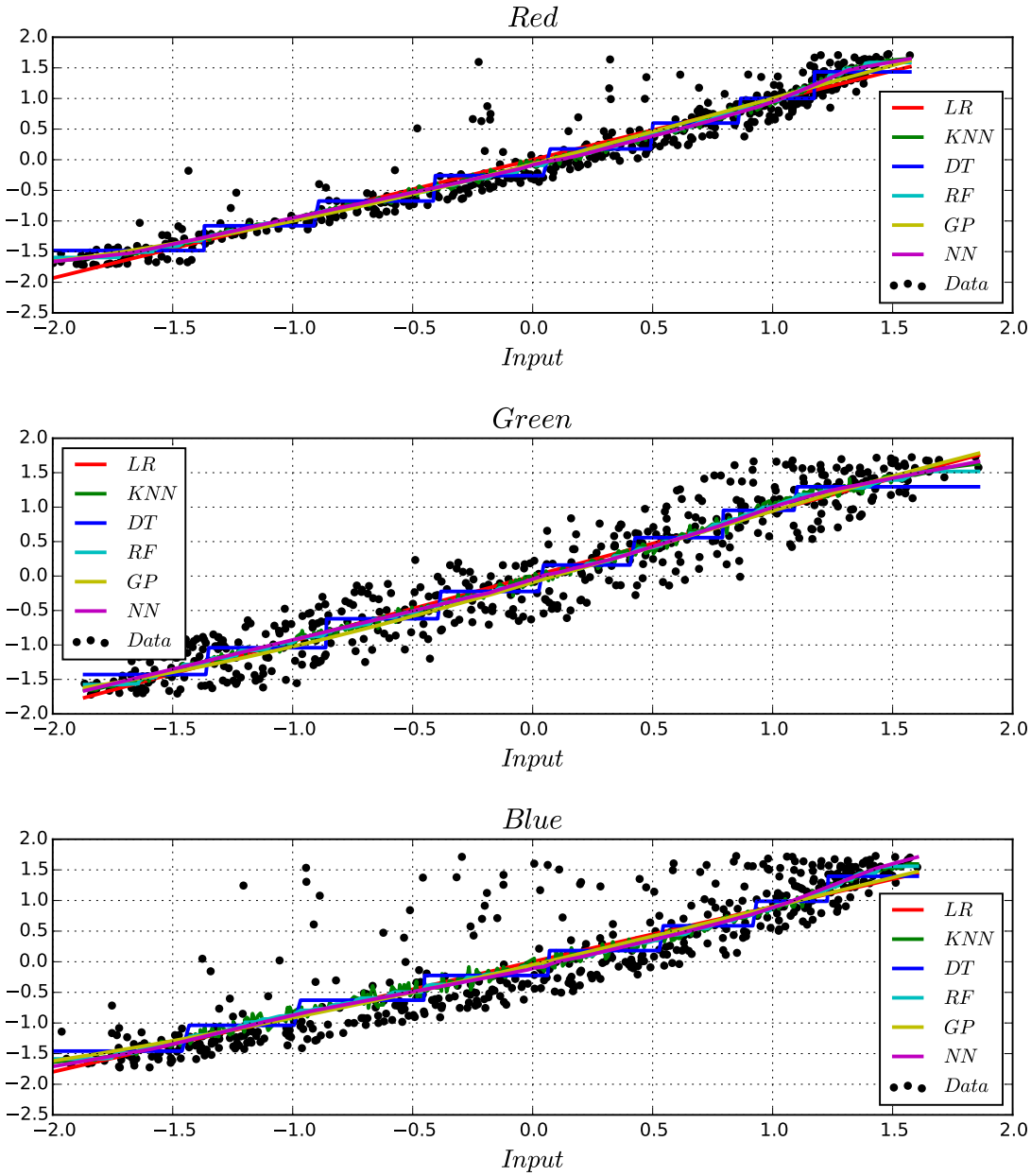


Figure 6.4: Regression fits: red, green and blue. A clear linear relationship is illustrated between all 3 colour channels and their corresponding most important latent codes.

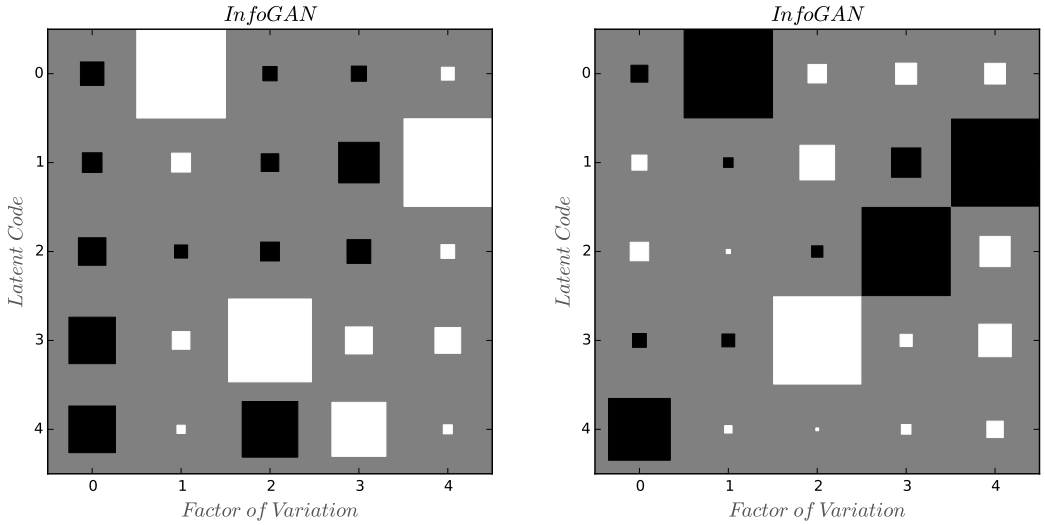
Gradient Penalty	Az.	Elev.	R	G	B
Original - 1	0.64	0.02	0.06	0.81	0.05
Original - 2	0.75	0.02	0.07	0.87	0.10
Original - 3	0.89	0.04	0.05	0.04	0.06
Original - 4	0.49	0.03	0.14	0.08	0.16
Updated	0.52	0.02	0.05	0.02	0.05

Table 6.3: Linear regression results: original penalty vs. updated penalty. Table shows SMSE on the test set. PC denotes principal components, IC denotes independent components and LC denotes latent codes. The number in the gradient penalty column indicates the random run number.

To further evaluate this improved stability, we assess the degree of separation achieved by each model. To avoid a clutter of diagrams, we compare the separation achieved by a single random run (run 1) to that which is achieved by the updated gradient penalty. As before, the degree of separation is assessed qualitatively using Hinton diagrams (see figure 6.5). In these diagrams, latent codes 2, 3 and 4 are clearly more separated for the InfoGAN trained with an updated gradient penalty (note the single dominant weight in each row). The degree of separation achieved by each penalty is evaluated quantitatively using the separation scores in table 6.4. Latent codes 2, 3 and 4 have a much higher separation score for the InfoGAN trained with an updated gradient penalty. The significantly higher average separation score provides quantitative evidence in support of improved stability. Taking all the qualitative and quantitative evidence into account, it is clear that the updated penalty results in more stable and reliable disentanglement.

Gradient Penalty	f_0	f_1	f_2	f_3	f_4	Average
Original	0.85	0.70	0.28	0.61	0.36	0.56
Updated	0.81	0.75	0.85	0.81	0.90	0.82

Table 6.4: Separation scores: original penalty vs. updated penalty. The average separation score is given for each input, along with the separation scores for each learnt factor (f_0 - f_4), i.e. latent code.



(a) Original GP **(b) Updated GP**

Figure 6.5: Visualizing the degree of separation within learnt representations for different gradient penalties. Both diagrams illustrate linear regression weights. Positive weights are represented by a white square and negative by a black square. The size of the square indicates the magnitude of the weight. Each row illustrates the importance of a given learnt factor to each prediction. Each column represents the importance of each learnt factor to a given prediction. Hence, a single large weight in each row indicates a high degree of separation. Factors of variation: 0=azimuth, 1=elevation, 2=red, 3=green and 4=blue.

Finally, we investigated the effect of data size on the InfoGANs ability to disentangle the factors of variation, comparing the SMSEs achieved with 100000 samples to the SMSEs achieved with 200000 samples. Table 6.5 presents these results, illustrating that more data only aids the models ability to disentangle azimuth, which was only the factor that was not not sufficiently disentangled with 100000 training examples. Despite explaining over 80% of the azimuth variance with 200000 samples, the InfoGAN clearly struggles to disentangle the azimuth more than any other factor. We discuss possible reasons for this in section 7.2.

Model	**Dataset size** (K)	**Az.**	**Elev.**	**R**	**G**	**B**
LR	*100*	0.52	0.02	0.05	0.02	0.05
	200	0.41	0.04	0.05	0.03	0.05
KNN	*100*	0.36	0.02	0.03	0.02	0.02
	200	0.22	0.03	0.02	0.01	0.03
DT	*100*	0.36	0.01	0.02	0.01	0.02
	200	0.22	0.03	0.02	0.02	0.03
RF	*100*	0.31	0.01	0.02	0.01	0.01
	200	0.19	0.03	0.01	0.01	0.02
GP	*100*	0.36	0.01	0.02	0.01	0.01
	200	0.22	0.03	0.01	0.01	0.02
NN	*100*	0.29	0.01	0.02	0.01	0.01
	200	0.17	0.02	0.01	0.01	0.01

Table 6.5: Regression results for different dataset sizes. Table shows SMSE on the test set. PC denotes principal components, IC denotes independent components and LC denotes latent codes.

6.2 Further evaluation

Disentangled representations should enable a model to perform zero-shot inference, that is, generalize its knowledge beyond the training distribution by recombining previously-

learnt factors. While the previous section quantified the impressive degree of disentanglement achieved by InfoGAN, this section looks to further evaluate the quality of the learnt representations by testing their ability to perform zero-shot inference and zero-shot reconstruction. Zero-shot inference performance is measured by regression SMSE on unseen factor combinations, while zero-shot reconstruction quality is evaluated by visually examining the models ability to generate unseen samples. By testing for these desirable characteristics of disentangled representations, we further evaluate the quality of the latent representations discovered by InfoGAN.

6.2.1 Creating 'gaps' in the data

Rather than re-generate two new datasets, each containing different factor combinations, we use the ground-truth values of the generative factors to create two different data distributions. More specifically, we create a 'gap' in the original dataset which will then serve as our test distribution containing unseen factor combinations.

To create this gap, we isolate all images whose generative factors have ground-truth values in a particular range. Informally, the images in this gap can be described as all 'red' teapots from 'above'. Formally, the generative factors of these images have ground-truth values that satisfying all of the following conditions:

- Red $>$ Green $+ 0.15$ and
- Red $>$ Blue $+ 0.15$ and
- Elevation $> \frac{\pi}{4}$

Figure 6.6 depicts samples at the extreme end of this gap for illustrative purposes, i.e. the most 'red' teapots with the highest angle of elevation. We will now refer to the images in this gap as the test set and the remaining images as the training set. However, it is important to note that this new training set is further divided into its own training, validation and test sets in order to fit the subsequent regression models. In other words, the SMSEs reported in subsequent tables were not calculated on the same data on which the regressor was trained. To give some indication of the difference in factor distributions between the new training and test sets, table 6.6 reports the SMSE of a constant regressor which guesses the expected value of the *training set*. In other words, this table indicates how much the (squared) expected value of a given factor has shifted, with a SMSE of 1 indicating no shift. Note that the SMSEs reported in

Figure 6.6: Samples from the test distribution.

subsequent tables are still relative to the same constant regressor as before, i.e. one that guesses the expected value of the *test set*. In order to facilitate a larger test set ($\approx$ 20000 samples), we used the dataset containing 200000 samples.

Input	**Az.**	**Elev.**	**R**	**G**	**B**
Constant	1.0	4.25	4.85	1.96	1.96

Table 6.6: Zero-shot inference: Constant regressor. Table shows SMSE on the test distribution, i.e. the gaps. Indicates how much the factor combinations vary between training and test distributions, with the constant regressor trained on the former. The SMSE indicates how much the (squared) expected value of a given factor has shifted.

6.2.2 Zero-shot inference

Table 6.7 presents the zero-shot inference results, comparing the SMSE on the training and test sets. Again, it is important to note that the training set is further divided into training, validation and test sets in order to fit the regression models, reporting the relevant SMSE on the test set. Only the highest and lowest capacity regressors are included for clarity, with all regressors illustrating similar trends.

With the linear regressor, the representations learned by the InfoGAN significantly out-

perform those learned by PCA and ICA. This can be seen by the very slight increases in SMSE when predicting the ground-truth values of unseen generative factors. For example, both PCA and ICA achieved a SMSE that was almost 4000% higher than that of the latent codes when predicting the elevation of teapots in the (unseen) test set. In fact, the SMSE of the baselines (2.95) is significantly higher than the constant regressor (1.0) which guesses the expected value of the test set. Once again, the InfoGAN struggles to disentangle the azimuth. In fact, the SMSE in predicting the azimuth is significantly higher on this new training distribution, with the linear model only able to explain 25% of azimuth variance. This may indicate that the InfoGAN found it more difficult to determine the azimuth of teapots at lower angles of elevation where the spout/handle ambiguity is heightened. For example, consider a teapot from above (elevation=$\frac{\pi}{2}$, teapots in figure 6.6). In this case, it is relatively clear which protrusion is the spout and which is the handle. The same cannot be said if the elevation=0 and the spout/handle is facing the camera (most teapots in figure 6.8). These discriminative characteristics likely help the InfoGAN to extract the value of the azimuth. Hence, the InfoGAN is likely to perform worse on a dataset with lower angles of elevation, such as the new training set. For the latent codes, the only SMSE which is significantly higher on the unseen data is that of the red channel. However, the generalization performance is still much better than the baselines.

With the neural network regressor, much of the same trends are observed. One difference is observed for the baselines, such as the increased SMSE on the test set when predicting the red colour channel. As before, the increased flexibility is more useful for the generative factors which have not been sufficiently disentangled in latent space, such as azimuth. Unsurprisingly, the more flexible fit to the training set does not generalize as well to the test set of unseen factor combinations.

Model	Input	Az.(train,test)	Elev.(train,test)	R(train,test)	G(train,test)	B(train,test)
LR	*PCs*	$0.71, 0.89$	$0.54, 2.94$	$0.22, 0.72$	$0.21, 0.68$	$0.20, 0.69$
	ICs	$0.71, 0.90$	$0.54, 2.95$	$0.22, 0.73$	$0.21, 0.67$	$0.20, 0.68$
	LCs	$0.76, 0.73$	$0.02, 0.08$	$0.04, 0.30$	$0.09, 0.13$	$0.03, 0.06$
NN	*PCs*	$0.47, 0.78$	$0.22, 1.25$	$0.15, 1.15$	$0.16, 0.32$	$0.15, 0.30$
	ICs	$0.49, 0.82$	$0.23, 1.60$	$0.15, 1.09$	$0.16, 0.30$	$0.15, 0.29$
	LCs	$0.55, 0.75$	$0.01, 0.05$	$0.01, 0.19$	$0.01, 0.04$	$0.01, 0.02$

Table 6.7: Zero-shot inference: InfoGAN vs. baselines. Table shows SMSE on the training and test sets. The test SMSE indicates performance on samples in the 'gap' that was created, while the train SMSE indicates performance on the remaining images. Again, note that the 'train' SMSE does not represent the SMSE on the same data that the regressor was trained on. The SMSE indicates each models ability to generalize to unseen factor combinations by recombining previously-learnt factors. PC denotes principal components, IC denotes independent components and LC denotes latent codes.

6.2.3 Zero-shot reconstruction

The promising zero-shot inference results in the previous section suggest that InfoGANs are indeed able to reason about unseen factor combinations by combining previously-learnt factors. More specifically, the results suggest that the neural network Q, which parametrizes the auxilary distribution $Q(\boldsymbol{c}|\boldsymbol{x})$, was able to reason about unseen factor combinations to produce latent codes which accurately captured the values of the underlying generative factors. In other words, given unseen combinations of factors in image space, Q produced latent codes which accurately represented the ground-truth values of these underlying factors. To test the other 'half' of the InfoGAN, the generator G, we wish to investigate if it can produce images which accurately represent the values of the underlying generative factors, as described by the (unseen) latent codes. In addition, we evaluate the sample quality of these images which are generated using unseen combinations of latent codes.

To do so, we selected a batch of images at the very edge of the test (gaps) distribution

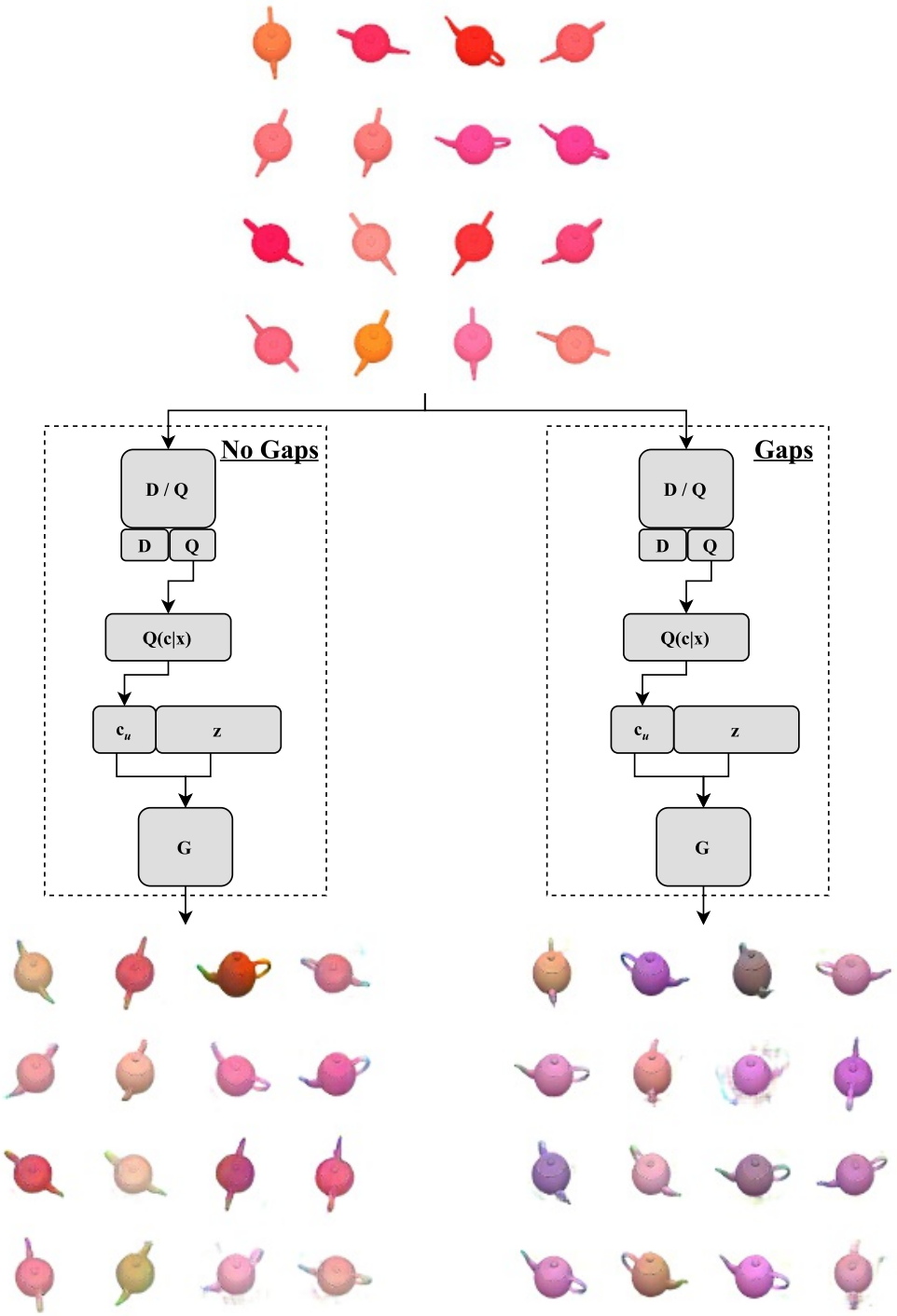


Figure 6.7: Zero-shot reconstruction quality. 'No Gaps' denotes an InfoGAN trained on the full data distribution. 'Gaps' denotes an InfoGAN trained on the data distribution with gaps. The image at the top depicts the ground-truth samples which were passed through Q to retrieve the parameters of $Q(\boldsymbol{c}|\boldsymbol{x})$. $\boldsymbol{c}_\mu$ denotes the mean of $Q(\boldsymbol{c}|\boldsymbol{x})$, i.e. the most likely latent codes given the ground-truth images x.

(a) No gaps **(b) Gaps**

Figure 6.8: Sharp random samples generated by the InfoGAN. (a) Random samples generated by the InfoGAN trained on the full dataset (no gaps). (b) Random samples generated by the InfoGAN trained on the dataset with gaps.

(i.e. the most 'red' teapots with the highest angle of elevation) and propagated them through both networks of the InfoGAN (see figure 6.7). As illustrated in this figure, the 'redness' of the teapots in the reconstructed images is clearly reduced for the model that has never seen such factor combinations, but also for the model which has. Both models also generate some blurry samples, which makes it difficult to conclude that the unseen combination of latent codes is the source of this blurriness. Perhaps the reduced sample quality is a result of generating images with codes that are at the edges of latent space. This would explain why both models produce blurry samples when asked to reconstruct these extreme or unseen images, but not when generating random samples (see figure 6.8).

Despite the relatively equal level of blurriness observed with both models, the 'redness' and elevation is clearly better-preserved by the InfoGAN which was seen such combinations before. For example, take the third teapot in the first column. The model which was trained on the full dataset (i.e. with no gaps) manages to preserve the ground-truth values of the generative factors for the most part, with the reconstructed image clearly still a red teapot from above. The same cannot be said for the image reconstructed by the model trained on the dataset with gaps. Similar trends can be seen for the second and third teapots in the first row. These results indicate that the generator is not able to perform zero-shot reconstruction to a high degree of accuracy, where accuracy refers to the extent to which the ground-truth values (described by the latent codes) are

preserved in the generated image.

Note that the primary source of these inaccurate reconstructions is not clear. Although the impressive zero-shot inference results of the previous section suggest that the latent codes produced by Q accurately represent the values of the generative factors in the input image, the SMSE increased from 0.04 on the training set to 0.30 on the test set when predicting the value of the red generative factor. Hence, these inaccurate reconstructions may stem from both inaccurate latent codes and a generation process which does not sufficiently preserve the information in 'extreme' latent codes (those on the edges of latent space or with previously-unseen combinations).

Finally, to determine whether or not G is in fact the primary cause of these inaccurate reconstructions, we create the 'gaps' in latent space rather than image space. In other words, we directly select values for the latent codes which correspond to ground-truth values which lie in original gap in image space. This direct selection was facilitated by figure 6.3 and figure 6.4, which reveal the positive and negative correlations that exist between the generative factors and important latent codes. By removing Q from the process, we are left with a controlled investigation. Hence, a single network, G, can be held accountable for poor zero-shot reconstructions. Figure 6.9 depicts the images generated with these selected latent codes. The InfoGAN trained on the full distribution (no gaps) manages to produce several red and 'almost red' teapots from above. The same cannot be said for the InfoGAN which was trained on the dataset with gaps, indicating that the generator is unable to reason about unseen combinations of latent codes to perform zero-shot reconstruction.

(a) No gaps **(b) Gaps**

Figure 6.9: Zero-shot reconstruction: latent space gaps. The generators of both InfoGANs received latent codes that were selected in the same manner. These codes were not identical as the latent variables of each model corresponded to different generative factors. (a) Images generated by the InfoGAN trained on the full dataset (no gaps), i.e. this models generator has seen similar factor combinations before. (b) Images generated by the InfoGAN trained on the dataset with gaps, i.e. this models generator has never seen such factor combinations.

Chapter 7

Conclusion

In chapter 5, we detailed the process of adding the mutual information cost of InfoGAN to the stable value function of the improved WGAN. We also demonstrated that the stability can be further improved with an updated gradient penalty. In chapter 6, we quantified this stability, providing empirical evidence that the updated penalty leads to stable and reliable disentanglement. We also thoroughly evaluated the quality of the latent variables discovered by InfoGAN, using our proposed metric to quantify the degree of disentanglement within the learnt representations before evaluating the models ability to reason about new factor combinations.

In this final chapter, we place our overall findings into context, discussing the main findings of our work along with their contributions to the field of unsupervised representation learning. In section 7.1, we summarize our findings and main contributions, outlining the relative strengths and weaknesses of our work in comparison to related research on learning disentangled representations. In section 7.2, we detail some possible shortcomings of our work, including the limitations of our findings and chosen methods. Finally, suggestions for further research are provided in section 7.3.

7.1 Contributions and discussion

Within this dissertation, the mutual information objective of InfoGAN has been combined with a recent and more stable value function to achieve stable and reliable disentanglement. Empirical evidence then suggested that this combination, along with an updated gradient penalty, takes another step towards truly unsupervised disentan-

gled factor learning by removing InfoGANs hyper-sensitivity to random initialization, network architecture and exact hyperparameter values. A new metric to quantify the degree of disentanglement achieved by different models is also proposed, with experimental results indicating that it accurately quantifies the two defining characteristics of disentangled representations. Although this new metric alone is sufficient to quantify the degree of disentanglement within learnt representations, visualizing the degree of separation through Hinton diagrams may still be useful for certain problems. In contrast to the recent 'factor change classification' metric of Higgins et al. in [6], this metric is simple and intuitive, requires no additional data, naturally generalizes to high-dimensional factors of variation, quantifies the amount of information captured and quantifies the degree of separation.

Additionally, the quality of the latent variables discovered by InfoGAN is elucidated through a number of qualitative and quantitative analyses, providing strong evidence in support of the model's ability to learn meaningful and interpretable disentangled representations. In stark contrast to the original work [11], the degree of disentanglement is evaluated quantitatively and evidence is provided that this disentanglement allows InfoGAN to understand the factorial structure of the data and thus reason about unseen factor combinations. In short, the evidence strongly suggests that extending generative models with a mutual information cost is a promising approach to disentangled factor learning.

7.2 Possible shortfalls

In this section, we detail some possible shortcomings of our work, including the limitations of our findings and chosen methods.

7.2.1 Use of deep ResNets

In retrospect, deep ResNets may not have been the best choice for our purposes. The added computational expense far outweighed the benefit of generating sharper samples and limited the breadth and depth of the hyperparameter searches. Furthermore, it limited the number of random runs that could feasibly be carried out in the given time frame, thus preventing a conclusive analysis on the stability improvement of the up-

dated gradient penalty. Although many hyperparameter searches were carried out with the initial InfoGAN model and the initial InfoWGAN-GP model (original gradient penalty), their respective instabilities may have masked the effect of certain hyperparameters. Thus, it would be interesting to re-run these experiments with the final model to discover the optimum number of latent codes, the precise effect of λ_{mi}, the optimum number of FC layers in Q, etc. We leave this to future work.

7.2.2 Fixed number of latent codes

As mentioned in the previous section, the computational expense of deep ResNets limited the number of experiments which could be carried out with the final model (updated gradient penalty). Perhaps the most important InfoGAN parameter which was not sufficiently explored was the number of latent codes. By using a fixed number of latent codes with this final model, we failed to conclusively show the models ability to learn meaningful and interpretable representations in a truly unsupervised manner, that is, without prior knowledge about the number of underlying factors of variation in the data. As further discussed in section 7.3.1, this is perhaps the largest unanswered question about the InfoGAN's ability to learn such meaningful and interpretable representations without supervision.

7.2.3 Redundant calculation

Finally, there is a small redundancy in the calculation of the updated gradient penalty. The update involves adding the per-image mutual information term $L_{\hat{I}}$ in equation 5.2 to the discriminator cost before calculating the gradient norm with respect to its input ($\hat{\boldsymbol{x}}$). However, the term $\log P(\boldsymbol{c})$ in equation 5.2 is independent of $\hat{\boldsymbol{x}}$, thus it does not need to be calculated.

7.3 Suggestions for future research

In this section, we propose some ideas for improving the InfoGAN and further evaluating the quality of latent variables discovered.

7.3.1 Excessive latent codes with a redundancy pressure

Training the InfoGAN with an excessive number of latent codes would determine its ability to learn highly semantic representations with no prior knowledge about the underlying factors of variation. This may also reveal whether or not the InfoGAN captures certain generative factors with multiple latent variables in order to maximize the mutual information. For example, the InfoGAN may have learned to represent the azimuth with two variables, representing the $\sin$ and $\cos$ of the azimuth (akin to the VAE in [6]), if there had been an additional latent code.

However, we speculate that the mutual information cost alone may not be sufficient to learn such meaningful and highly semantic representations with an excessive number of latent codes ($D >> K$). More specifically, we conjecture that this cost would not be sufficient to force individual dimensions of the learnt representation to correspond to salient semantic features of the data if there are many more latent variables than factors of variation. For example, if we retrained the InfoGAN on our dataset with 100 latent codes, maximum mutual information may still be achieved if generative factors such as elevation are captured by a combination of many latent variables.

Thus, to ensure that individual dimensions of the representation correspond to semantic features of the data and thus take another step towards black-box disentanglement, we propose adding a 'redundancy pressure' to the objective function as was done in [6]. The mutual information objective already reduces the redundancy between the latent codes and generated image, however, with excessive latent codes, there is likely to be redundancy within the latent codes themselves. Adding a redundancy pressure may encourage the network to capture the generative factors with the lowest possible number of latent variables, thus preventing; i) multiple factors capturing the same information, ii) a generative factor being captured across an excessive number of latent variables. We conjecture that this would allow the InfoGAN to learn meaningful and highly semantic representations without any prior knowledge about the number of underlying factors of variation.

7.3.2 Unseen objects

As illustrated by Higgins et al. in [6], models that understand the factorial structure of data should be able to reason about the properties of unseen objects. To further evaluate

the latent variables discovered by InfoGAN, one could test their ability reason about unseen objects such as mugs, which can be rendered using the same process employed in this dissertation [40].

7.3.3 Richer data

The render from [40] also has the ability to generate richer data, including teapots with varying shape, lighting and background scenes. In theory, the CNNs of the InfoGAN would allow it to be invariant to such background noise. However, evaluating the InfoGAN on such a dataset would further provide further evidence of its capabilities.

Appendix A

GAN instability

A.1 Initial InfoGAN architecture

D/Q	G
Input 64x64x3 image	Input $\in \mathbb{R}^{133}$
conv4x4. 64. IReLU. s=2.	FC (4x4x1024). BN
conv4x4. 128. IReLU. s=2. BN	upconv4x4. 512. ReLU. s=2. BN
conv4x4. 256. IReLU. s=2. BN	upconv4x4. 256. ReLU. s=2. BN
conv4x4. 512. IReLU. s=2. BN	upconv4x4. 128. ReLU. s=2. BN
FC output D/Q	upconv4x4. 64. ReLU. s=2. BN
	upconv4x4. 3. tanh

Table A.1: Architecture of the initial InfoGAN. 'conv4x4. 128. IReLU. s=2. BN' indicates a convolutional layers with a stride of 2 and 128 feature maps, followed by a batch normalization layer, followed by an 'inverse' ReLU activation function.

A.2 Samples from the original dataset.

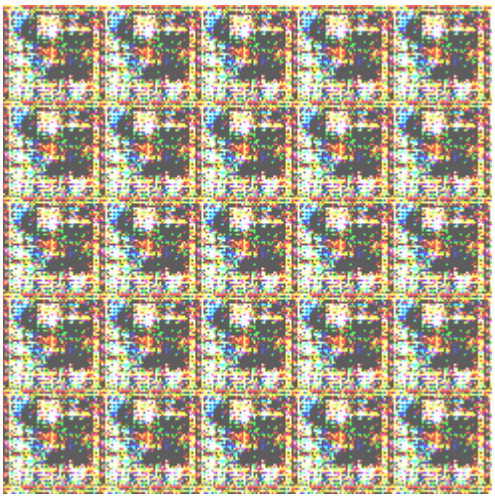


Figure A.1: Best samples generated with the initial InfoGAN and original dataset.

A.3 Samples from the simplified dataset.

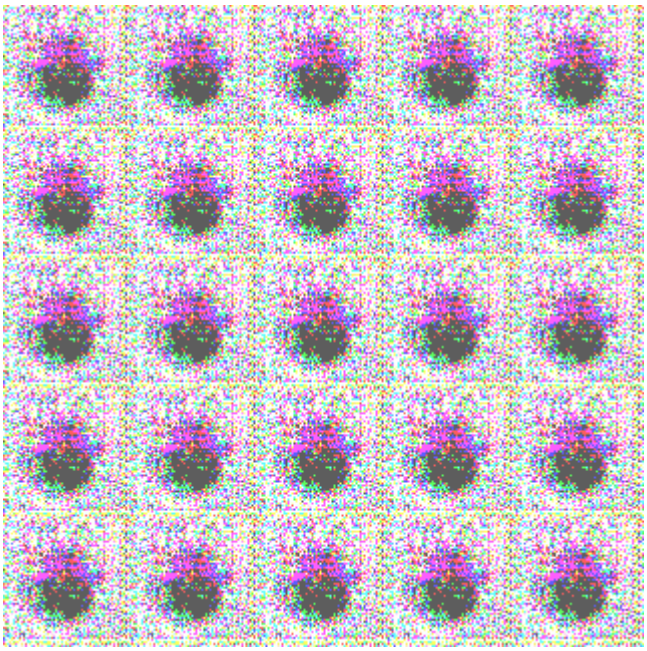


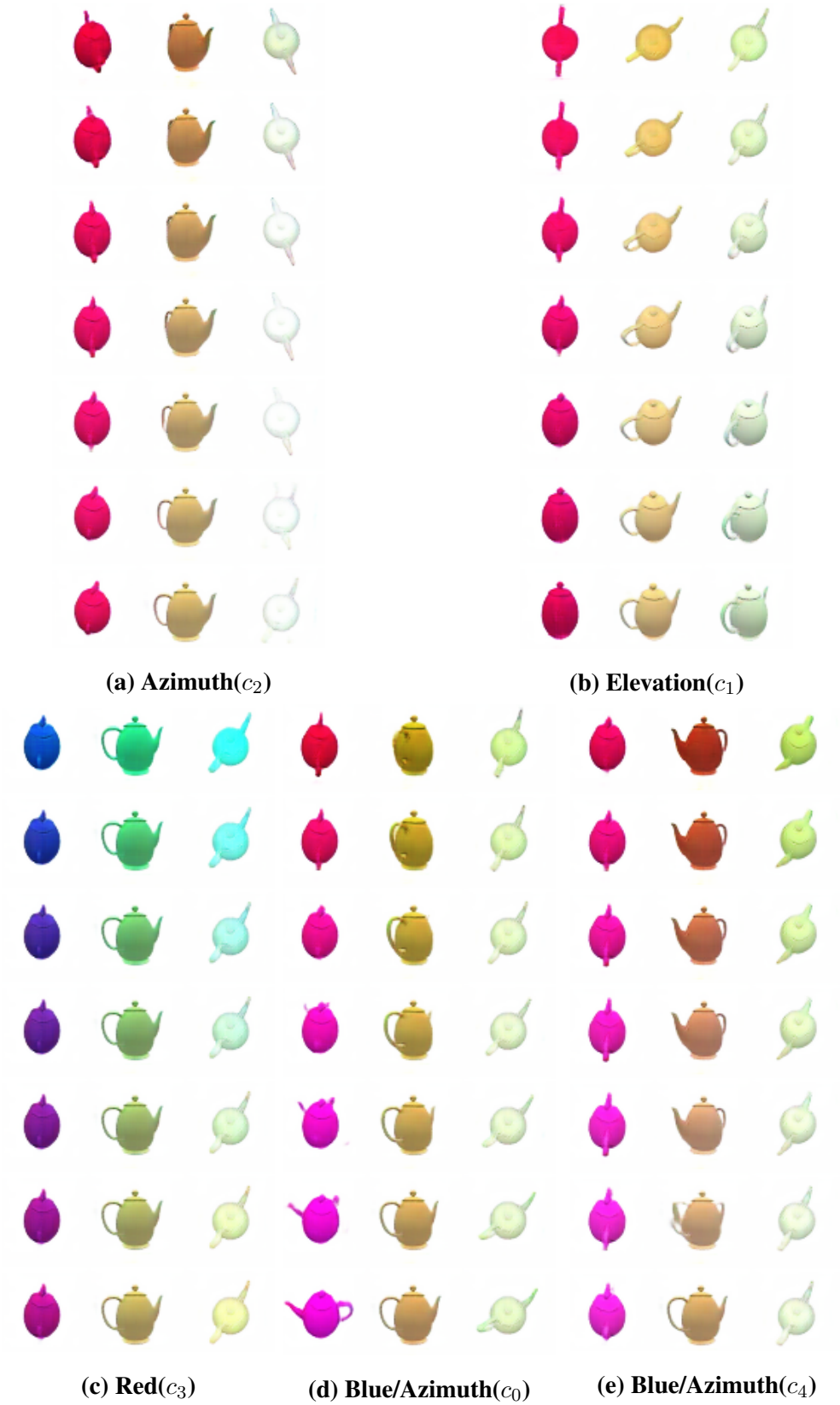
Figure A.2: Best samples generated with the initial InfoGAN and simplified dataset.

Appendix B

Additional Samples

B.1 Disentangled representations.

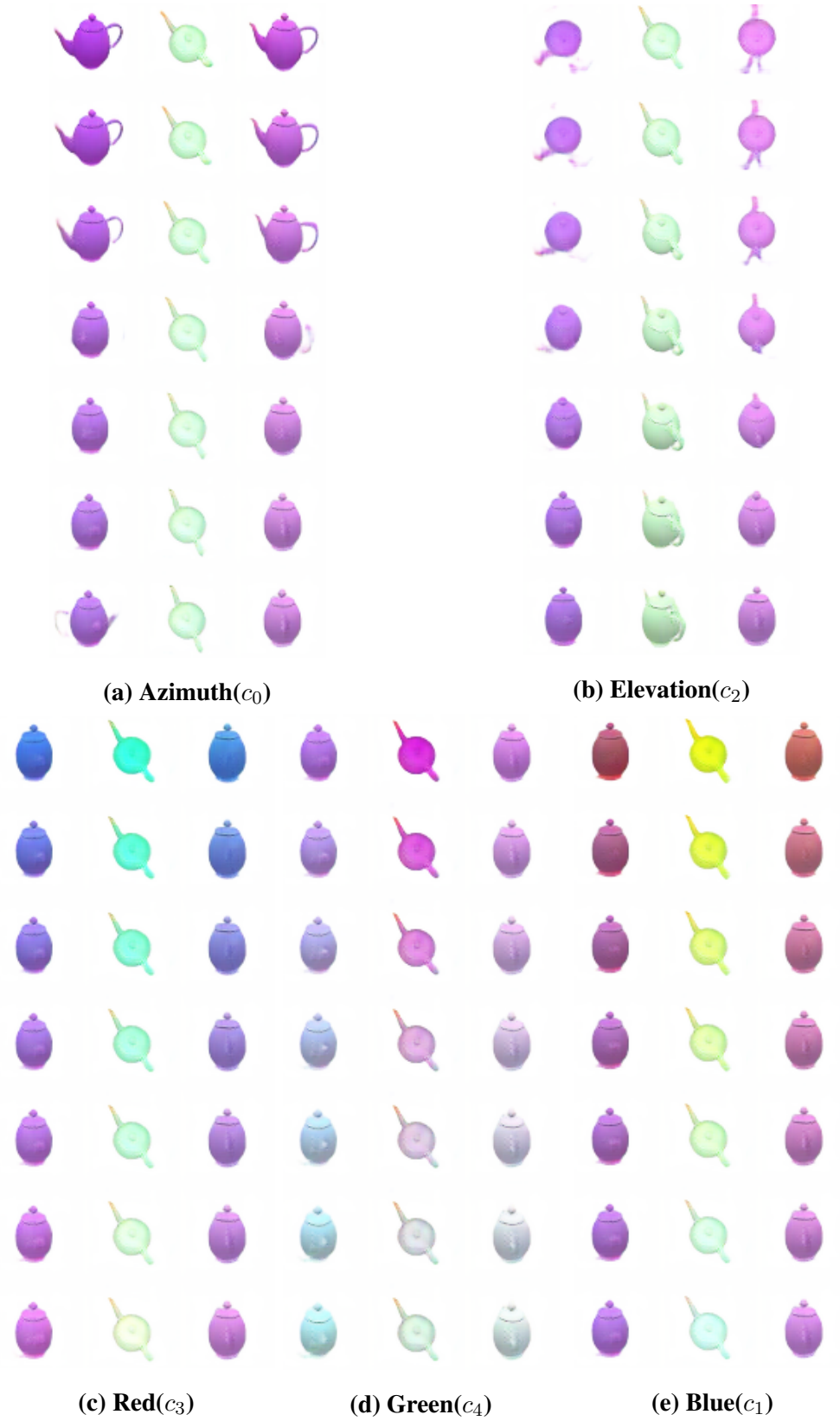
B.2 Large mutual information coefficient



(a) Azimuth(c_2) **(b) Elevation(c_1)**

(c) Red(c_3) **(d) Blue/Azimuth(c_0)** **(e) Blue/Azimuth(c_4)**

Figure B.1: Manipulating the latent codes of run 2. The model appears to have successfully disentangled (b) elevation, (d) green and (e) blue. However, azimuth and red are clearly tangled in (a) and (c).



(a) Azimuth(c_0) **(b) Elevation(c_2)**

(c) Red(c_3) **(d) Green(c_4)** **(e) Blue(c_1)**

Figure B.2: Manipulating the latent codes of run 3. The model appears to have successfully disentangled (b) elevation, (d) green and (e) blue. However, azimuth and red are clearly tangled in (a) and (c).

(a) Azimuth(c_2) **(b) Elevation(c_4)**

Figure B.3: Manipulating the latent codes of run 4. The model appears to have successfully disentangled (a) azimuth and (b) elevation.

(c) Red/Green(c_1) **(d) Green/Azimuth(c_3)** **(e) Blue(c_0)**

Figure B.3: Manipulating the latent codes of run 3. The model appears to have successfully disentangled appearance from pose. However, the colour channels are tangled. c_1 increases the red channel and decreases the green channel as it ranges from -1 to 1. c_0 increases the blue channel and decreases the red channel. c_3 decreases the green channel.

Figure B.4: Samples generated with $\lambda = 25$.

Bibliography

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, pp. 1–101, 2016.

[6] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early visual concept learning with unsupervised deep learning," *arXiv preprint arXiv:1606.05579*, 2016.

[7] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th International Conference on Machine learning*, pp. 759–766, ACM, 2007.

[8] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[9] G. Desjardins, A. Courville, and Y. Bengio, "Disentangling factors of variation via generative entangling," *arXiv preprint arXiv:1210.5474*, 2012.

[10] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[11] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[13] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[14] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016.

[15] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[16] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*, pp. 44–51, Springer, 2011.

[17] S. Reed, K. Sohn, Y. Zhang, and H. Lee, "Learning to disentangle factors of variation with manifold interaction," in *International Conference on Machine Learning*, pp. 1431–1439, 2014.

[18] Z. Zhu, P. Luo, X. Wang, and X. Tang, "Multi-view perceptron: a deep model for learning face identity and view representations," in *Advances in Neural Information Processing Systems*, pp. 217–225, 2014.

[19] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.

[20] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," *arXiv preprint arXiv:1704.00028*, 2017.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[23] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, pp. 9–50, Springer, 1998.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, 2015.

[26] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5353–5360, 2015.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*, pp. 630–645, Springer, 2016.

[28] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

[30] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[31] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[32] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[33] A. Srivastava, L. Valkov, C. Russell, M. Gutmann, and C. Sutton, "VEEGAN: Reducing mode collapse in GANs using implicit variational learning," *arXiv preprint arXiv:1705.07761*, 2017.

[34] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The Cramer distance as a solution to biased Wasserstein gradients," *arXiv preprint arXiv:1705.10743*, 2017.

[35] T. S. Cohen and M. Welling, "Transformation properties of learned visual representations," *arXiv preprint arXiv:1412.7659*, 2014.

[36] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[37] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[38] OpenAI, "InfoGAN code." `https://github.com/openai/InfoGAN`. Accessed: 2017-04-05.

[39] "Improved WGAN training code." `https://github.com/igul222/improved_wgan_training`. Accessed: 2017-01-07.

[40] P. Moreno, C. K. I. Williams, C. Nash, and P. Kohli, "Overcoming occlusion with inverse graphics," in *ECCV Geometry Meets Deep Learning Workshop 2016*, pp. 170–185, Springer, 2016.

[41] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[42] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness:

Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[43] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.

[44] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[45] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[46] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*, vol. 1. MIT press Cambridge, 2006.