

University of Southampton Research Repository

Copyright © and Moral Rights for this thesis and, where applicable, any accompanying data are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis and the accompanying data cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content of the thesis and accompanying research data (where applicable) must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holder/s.

When referring to this thesis and any accompanying data, full bibliographic details must be given, e.g.

Thesis: Author (Year of Submission) "Full thesis title", University of Southampton, name of the University Faculty or School or Department, PhD Thesis, pagination.

Data: Author (Year) Title. URI [dataset]

UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

**On the Learning and Structure of Symmetry
Based Disentangled Representations**

by

Matthew Painter

ORCID: 0000-0003-0666-2497

*A thesis for the degree of
Doctor of Philosophy*

14th June 2022

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Doctor of Philosophy

On the Learning and Structure of Symmetry Based Disentangled Representations

by **Matthew Painter**

Representation learning is fundamental to many machine learning techniques, perhaps even more so in the subfield of deep learning. Disentangled representations introduce a form of interpretability such that both humans and our models can understand (to a degree) how decisions are made - or at least, what is important to such decisions. Symmetry based disentangled representations introduce a stricter form of interpretability which ensures the representations are structured based on how the data is observed, which is described by symmetries acting on it. In this work we explore two aspects of symmetry based representations. First we consider methods to learn such representations with a particular focus on doing so without action labelling. Secondly we consider their structure and potential benefits for downstream tasks. We find that through the use of policy gradients, we can successfully learn linear disentangled representation without knowledge of world states or action labels. Indeed our proposed method achieves similar performance to supervised methods. Subsequently, we find that linear disentangled representations are highly structured with respect to a particular symmetry group. This structure allows for better performance than standard disentangled representations on both the tasks of generative factor prediction and observed action prediction.

Contents

List of Figures	xi
List of Tables	xvii
List Of Abbreviations and Symbols	xix
Declaration of Authorship	xxi
1 Introduction	1
2 Groups and Representations	5
2.1 What is symmetry?	5
2.2 Group Theory	6
2.3 Representation Theory	11
3 Symmetry Based Disentangled Representation Learning (SBDRL)	17
3.1 Variational Auto Encoders	17
3.2 Disentanglement as a Concept	20
3.2.1 Generative Factors	21
3.2.2 Models for Disentanglement	23
InfoGAN	23
β -VAE	25
Controlled Capacity Increase VAE	27
InfoVAE	28
BetaTcVAE	29
3.2.3 Datasets for Disentanglement	31
FlatLand	32
dSprites	32
Norb	33
3.2.4 Measuring Disentanglement	34
Latent Space Visualisations	34
BetaVAE Metric	35
Mutual Information Gap (MIG)	36
DCI Informativeness, Disentanglement and Completeness	37
Modularity and Explicitness	38
SAP	39
Our Implementations	39
3.2.5 Common Assumptions	39

3.3	Towards a Definition of Disentangled Representations	40
	SBDRL Problem Setting	41
	How do Symmetry Groups Fit In	42
	Disentangled Group Actions	43
	Linear Disentangled Representations	45
	Disentanglement with respect to symmetries	46
3.3.1	Symmetries on Disentanglement Datasets	47
	FlatLand	47
	dSprites	47
	Norb	48
	Transition Datasets	48
3.3.2	VAEs as the representation function	48
3.3.3	Linear Symmetry Based vs Classical Disentanglement	49
4	Learning Linear Disentangled Representations	53
4.1	Actions on VAE Subspaces	54
4.1.1	The Idealised Model	54
	Classifying Disentangled Factors	54
	Learning a Disentangled Simulator	56
4.1.2	Approximating world space actions	56
	Learning Actions on VAE Subspaces	56
	Learning known structure	58
4.2	ForwardVAE	60
4.2.1	Learning Observed Actions	62
4.3	Unsupervised Action Estimation	63
	GroupVAE	63
	Action Selection	64
	Reinforcement - Policy Gradients	64
	Attentional Mechanisms	65
	Attention or Reinforcement?	66
	Internal Representations	66
	Change of Basis	67
	Properties	68
4.4	How do we measure LDR	69
	Independence Score	69
	Relative Latent Error	70
	SVD Overlap	71
	Factor Leakage	72
	Comparison	72
	Empirical Comparison	73
4.5	Which Spaces Admit LDR	74
	Possible Actions	75
	Method	75
	Pretrained VAEs	76
	Results	76
	Do baselines show Linear Disentanglement?	78
	Extending to dSprites	80

4.6	Evaluating GroupVAE	82
	Metrics	82
	Attention or Reinforcement	82
	Symmetry based models under disentanglement metrics	84
4.6.1	Properties	86
	Training Scheme	86
	Temporal Consistency	86
	Long Action Sequences	87
	Over Representation	88
	Robustness - Visual Noise	89
4.6.2	Backbone VAEs	90
	Training Scheme	91
	Backbones	91
4.6.3	Policy Convergence	91
	Training Schemes	91
	Learning Rates	92
	Regret	92
	Change of Basis	92
	Exploration Strategy	93
	Internal Representations	93
4.6.4	Hyperparameters	95
	Training Scheme	95
	Capacity β	95
	Prediction weight γ	96
4.7	Conclusions	96
	Can we learn linear disentangled representations?	96
	Are they learnt by standard VAE models?	97
	How do we measure them?	98
	Can we learn them without action supervision?	98
5	Structure and Benefits of LDR	99
5.1	Cyclic Structures	99
	Varying Symmetries on FlatLand	100
	Action Metrics on Flatland Variants	101
	Training Scheme	101
5.1.1	Posterior Structure	101
	Visualising the Posterior	102
	Cyclic Posterior	102
	How does the objective influence the latent structure	103
	Does the KL cause Wrapping Posteriors on FlatLandN?	104
	Low KL ($\beta < 0.2$)	104
	Medium - High KL ($\beta > 0.2$)	105
	Internal Representation	106
	Other Data	107
5.1.2	Prior Structure	108
	Normalising Flows	108
	Complex Priors with Normalising Flows	111

	Experimental Setup	111
	Basic Properties	111
	Learning the True Structure	114
	Flow Complexity	115
5.2	Distinguishing Linear Disentangled Representations	117
	Experimental Setup	117
	Disentanglement Metrics	117
	Evaluation Metrics	117
	Results	119
5.3	Downstream Tasks	121
	Experimental Setup	121
	Predicting Actions	122
	Predicting Factors	123
	Symmetry Based Models on Downstream Tasks	125
5.4	Data Efficiency	126
	Experimental Setup	126
	Are symmetry based models more efficient?	126
	Do symmetry metrics correlate with efficiency?	130
	Combined Data Efficiency	134
	Optimal Metric Combination for Data Efficiency	134
5.5	Conclusions	136
	What is the posterior structure for ForwardVAE on Flat-Land?	137
	Can we find better priors?	137
	Can we find evidence of LDRs in disentanglement metrics?	138
	How do LDRs compare to classical representations under data efficiency and downstream tasks?	138
	Can we find any correlations with data efficiency?	139
	Action Prediction vs Factor Prediction	139
6	Conclusions	141
6.1	Future Directions	142
	How can we determine the symmetry structure of a linearly disentangled representation without prior knowledge?	142
	Can we encourage LDRs in standard VAEs?	143
	Large Scale Study of LDR Efficiency	143
	Planning Tasks	143
	Representations of Continuous Groups	144
	Can we learn intermediate spaces alongside training a VAE	144
	Learning LDRs by Reinforcement Tasks	145
	Appendix A Correlation Scatters	147
	Appendix B Standard Model Architectures	151
	Appendix B.1 Shapes	151
	Appendix B.2 Celeb	151
	Appendix B.3 Forward	152

Appendix B.4	TC Model	152
Appendix B.5	MMD-VAE	152
Appendix B.6	RGrVAE Action Estimator	153
Appendix B.7	RGrVAE	153
Appendix C	Training Schemes	155
Appendix C.1	FlatLand	155
	Appendix ForwardVAE	155
	Appendix RGrVAE	155
	Appendix BetaVAE	155
	Appendix DIP-VAE	155
	Appendix FactorVAE	155
	Appendix BetaTCVAE	156
	Appendix MMD-VAE	156
Appendix C.2	dSprites	156
	Appendix RGrVAE	156
	Appendix VAE	156
	Appendix BetaVAE	156
	Appendix Dip VAE i	156
	Appendix Dip VAE ii	156
	Appendix Extental models	156
Bibliography		157

List of Figures

2.1	Symmetries of the Square.	6
2.2	Other examples of symmetries or patterns with symmetries.	6
2.3	Visual example of the composition $\sigma_0 \cdot \rho_3 = \sigma_3$	8
2.4	Visual orbit of $\alpha = (A, B, C, D)$ under subgroup of rotations $\{\rho_0, \rho_1, \rho_2, \rho_3\}$	10
3.1	Simple VAE schematic.	20
3.2	Example factors of variation as captured by Dupont [2018] on the MNIST dataset of handwritten digits. Each subfigure traverses a different latent unit and shows a different factor of variation. The ability to learn to separate these factors into different latent units is generally known as disentanglement. Figure by Dupont [2018].	22
3.3	A selection of Flatland images with the associated generative factors (x, y) (top row) and an example of an action applied to the top row with results shown in the bottom row.	32
3.4	A selection of dSprites images with their associated generative factors (shape, scale, orientation, x, y).	33
3.5	A selection of Norb images with their associated generative factors (azimuth, toy, elevation, lighting)	34
3.6	Example Disentangled and Entangled latent traversals for a simple two factor problem (x and y position) using 4 latent dimensions (rows).	35
3.7	Visualisation of the Grid World explored by Higgins et al. [2018] consisting of a circular agent moving on a grid with periodic boundary conditions alongside a circular hue axis.	41
3.8	Visual example of wrapping on sphere. Unfolding the dome (left) as you would a world map (similar to a Mercator Projection) would result in a 2D grid (right), where we can see the object wrapping. Whilst this analogy holds for a single cyclic symmetry, if we were to include the 'y' axis, then a torus would be the correct geometric depiction.	42
3.9	Function relationships for SBDRL	44
3.10	Visual schematic of the SBDRL problem setting for a VAE on dSprites	49
4.1	Idealised inference (A) and simulator (B) results. (A) We can classify dSprites generative factors directly from pixels to a high degree of accuracy. (B) We can learn to simulate dSprites images from the generative factors.	55

4.2	Comparison of unsupervised and supervised methods for learning actions a on pretrained VAE subspaces, defined by encoder ϕ , decoder θ and action estimator ψ . They differ in where the loss (\mathcal{L}) is computed. The loss is between two quantities joined by the dashed line. When supervised, the loss is computed in the latent space (blue) and when unsupervised it is in the image/input space (red).	57
4.3	Post Action Observation reconstruction error for each dSprites factor using the factor-wise and observation-wise models.	57
4.4	Some examples of an input image (top row), the observation after applying an action in world space (second row), the reconstruction after applying the learnt action (third row) and the true reconstruction of the second image (bottom row). The models learn to apply most actions fairly well, and can't learn to change the shape. Generated with CCVAE and capacity 25. Note that actions are generally visually small (change only a few pixels). Actions are most noticeable in scale or shape change.	58
4.5	Visualisation/Orbit of learnt actions (Scale, Rotation, X translation and Y translation) on dSprites. Taking the same initial image, each row shows a different internal representation / action as it is repeatedly applied. There is some degradation as actions are repeated (see x translation in the third row) although most actions are of good quality.	59
4.6	Evaluating a model trained to learn cyclic structure on pre-existing latent spaces. (left) MSE for a single action. We don't see the degradation from Fig. 4.5, suggesting that locally, the action estimation is good. (right) We see that the error gradually increases as the actions are composed. Solid lines indicate mean performance with 1 standard deviation given as the shaded area.	60
4.7	ForwardVAE	62
4.8	Schematic diagram for the GroupVAE variants. Dashed lines denote possible paths dependent on policy. Weighted lines represent the strength of attention. ■ - Learnable module. ■ - Loss. ■ - Operation without parameters.	64
4.9	Parameter count and runtime charts for RGrVAE on FlatLand, both given in excess of those of the underlying VAE. Solid lines denote GroupVAE with a change of basis matrix also learnt. Dashed lines denote GroupVAE without a change of basis. When not varying the latent dimension number, we used 8 latent dimensions, and in all experiments used cyclic internal representations. Batch times were estimated on FlatLand with a batch size of 128 over 10 batches.	68
4.10	Comparison of Symmetry Metrics	74
4.11	Minimum MSE for translations in each direction against axis aligned translations (left) and axis aligned as well as diagonal translations (right). We can see the minimum MSE decreases dramatically once we consider additional translation vectors. Step Size: 5	75
4.12	Metrics when attempting to find linear disentangled representations in VAE latent spaces. We look for actions which translate on the normal (Cartesian) and diagonal (Cartesian rotated 45 degrees) axes. In all sub-figures, blue denotes results under the normal axes and red those under the diagonal axes. Solid lines indicate mean performance with 1 standard deviation given as error bars.	77

4.13	Singular value analysis for baselines and ForwardVAE on FlatLand. (B) Solid lines indicate mean performance with 1 standard deviation given as the shaded area.	78
4.14	Scatter of individual runs of baseline and symmetry based VAEs for independence and symmetry metrics. The cluster of linear disentangled models is highlighted. (left) Scatter of independence (x), symmetry recon (y). (right) Scatter of independence (ni) (y), independence (di) (x) . .	79
4.15	Learnt action orbits on two different Dip-VAE latent spaces. We can see that sometimes BetaVAE learns a linear representation, and other times it does not. Note that even in the non-linear case, the lower two actions appear to be somewhat linear.	79
4.16	Action orbit traversals for all available actions of RGrVAE trained on dSprites and a heat-map over the actions selected over the dataset. We sampled the true actions from the dataset based on the following symmetry groups: Scale: C_3 , Rotation: C_{10} , Translation: C_8 . Thus the group acting on the data is: $G = C_3 \times C_{10} \times C_8 \times C_8$	80
4.17	Reconstruction metrics for RGrVAE and baselines on dSprites. Solid lines indicate mean performance with 1 standard deviation given as error bars.	81
4.18	Singular value analysis for baselines and RGrVAE on dSprites. (B) Solid lines indicate mean performance with 1 standard deviation given as the shaded area.	82
4.19	Action traversals for RGrVAE and AGrVAE. AGrVAE has both traversal of the internal representations directly and based on the attention mask generated once for each action.	83
4.20	Distribution of attention in expectation over the latent space.	83
4.21	Disentanglement metrics for FlatLand (blue) and dSprites (red). Solid lines indicate mean performance with 1 standard deviation given as error bars.	84
4.22	Symmetry Disentanglement metrics for FlatLand and dSprites. Solid lines indicate mean performance with 1 standard deviation given as error bars.	85
4.23	Comparing ForwardVAE and RGrVAE under temporal consistency. Solid lines indicate mean performance with 1 standard deviation given as the shaded area.	87
4.24	Long Action Sequences	88
4.25	Over Representation when we have 1, 2 or 3 internal representations per true action. (A) 1 standard deviation is given as the shaded area.	88
4.26	Metric scores for ForwardVAE and RGrVAE when trained on FlatLand under different distractors.	90
4.27	RGrVAE performance with different backbone VAE models. DIP-VAE backbones achieve slightly better independence, but slightly worse observation and symmetry errors. There is no consistent benefit.	91
4.28	Policy convergence on FlatLand. RGrVAE is sensitive to learning rate choices, but less so to regret and change of basis. Solid lines indicate mean performance with 1 standard deviation given as the shaded area. .	92

4.29	Exploration Strategies in RGrVAE. For FlatLand, entropy weighting was not particularly beneficial, but some amount of random action selection can offer faster convergence. Solid lines indicate mean performance with 1 standard deviation given as the shaded area.	93
4.30	Comparing RGrVAE internal representations on FlatLand. Solid lines indicate mean performance with 1 standard deviation given as the shaded area. Dashed lines indicate the min and max performance.	94
4.31	RGrVAE Hyperparameters. Large Beta and very small or very large gamma can result in poor performance.	95
5.1	Visualisations of the posteriors for both action planes of 10 ForwardVAE instances trained on FlatLandN.	103
5.2	Action map on the non-cyclic posterior seen in Fig. 5.1.	104
5.3	FlatLandN posterior distributions with varying KL-Divergence weightings. See the general trend towards clustering around the origin at higher KL weightings. Low weightings tend to align with the Cartesian axes.	105
5.4	Posterior matrix vs cycles	106
5.5	Example posteriors from a ForwardVAE with cyclic internal representations and 0 KL weighting. Note that the non-cyclic posteriors were significantly further away from the origin and all axes are scaled to best show structure.	107
5.6	Example flows on a Gaussian distribution. Note that the RealNVP is simply randomly initialised, and isn't representative of all possible flows. For both Planar and Radial flows, we show a single flow layer, in general models use many layers, we use 10-20 for our models.	111
5.7	The concept behind normalising flows to emulating prior change. We allow a complex VAE posterior that is gradually mapped back to the standard prior.	112
5.8	Comparison of priors under normalising flows on FlatLand. (A) Kernel density estimation for the validation KL-Divergence error term for flow based models. (B) Convergence of validation observation reconstruction error as the model is trained. Lines indicate mean whilst shaded area indicates 1 standard deviation.	112
5.9	Comparing ForwardVAE posteriors on FlatLandN to those from flow based models.	114
5.10	Exploring flow based models on FlatLandN, where there are no cycles in the data. (A) Validation observation reconstruction error over training. Lines indicate mean, shaded area indicates 1 standard deviation. (B) Boxplot of KL-Divergence error for the flow based models.	114
5.11	Flow Complexity on FlatLandG. (A) KL-Divergence error for models with different depth flows. (B) Observation reconstruction for different depth flows. (C) Convergence times for models with different depth flows.	116
5.12	Relevance of each metric to distinguishing linear from classical disentanglement. ROC and PR are computed directly on the metric values and given as AUC measures. Scores are the probability that an RBF-SVM can classify the model correctly from the metric alone. $P[L N]$ is the probability the SVM predicted a model was (L)inear disentangled when it was (N)ot/(N)ormal.	119

5.13 FlatLand and dSprites Downstream performance on action prediction for all baselines.	123
5.14 Generative factor prediction downstream performance for baselines and symmetry based models on FlatLand and dSprites using a Gradient Boosted Tree (GBT) classifier. Symmetry based models perform better on both datasets in this task.	124
5.15 Data Efficiency Scores	127
5.16 Downstream Score Comparisons	128
5.17 Scaled Data Efficiencies.	129
5.18 Correlating data efficiency scores with disentanglement metrics.	131
5.19 Correlating scaled data efficiency scores with disentanglement metrics.	133
5.20 Correlation of data efficiency with metrics for combined FlatLand and dSprites.	135
5.21 Optimal linear combination of metrics to correlate with data efficiency as the weight penalty is increased.	136
5.22 Scatter metric combinations against efficiency for L1 penalty weights 1 and 5.	136
Appendix A.1 Scatter of data efficiency (y) and disentanglement metrics (x) for FlatLand factor prediction task.	147
Appendix A.2 Scatter of data efficiency (y) and disentanglement metrics (x) for FlatLand action prediction task.	148
Appendix A.3 Scatter of data efficiency (y) and disentanglement metrics (x) for dSprites factor prediction task.	148
Appendix A.4 Scatter of data efficiency (y) and disentanglement metrics (x) for dSprites action prediction task.	149

List of Tables

2.1	Binary operation tables for symmetry group of squares and the subgroup of rotations under $\cdot : G \times G \rightarrow G$	8
3.1	Loose relationships between SBDRL spaces and spaces in the VAE framework.	49
4.1	Comparison of RGrVAE and AGrVAE reconstruction metrics.	83
5.1	Definition of FlatLand variants and example samples. Arrows indicate possible actions. Red arrow indicates action chosen for that sample. ‘None’ restricts actions at a boundary so that the object never cross it. ‘Contact’ warps the agent to the other side of the grid on contact. ‘Gradual’ allows continuous transitions through boundaries. The radius of the agent is given by $R = 15$ and the step size corresponding to an action is 5 pixels.	100
5.2	Metrics which describe how many actions are required to move between two states in each FlatLand variant, assuming a step size of 1.	101
5.3	Posteriors and action maps for FlatLand variants on the latent subspace of a ForwardVAE which corresponds to y translation on each of the dataset variants. Both posteriors and action maps have the same axis limits .	102
5.4	Additional posterior visualisations for actions.	108
5.5	Example posteriors and parametrised (i.e. after the flow layers) posteriors for FlatLandG. Note that colour scale is constant for the posteriors but not for the parametrised posteriors, which are significantly higher density than the posteriors. Black dot indicates the origin. ‘None’ refers to the baseline model with no flow layers.	113
5.6	Disentanglement metric summary.	118

List Of Abbreviations and Symbols

Abbreviations

AUC	Area Under Curve
DCI	Disentanglement, Complexity, Informativeness - We often use it to refer solely to the Disentanglement
ELBO	Evidence Lower Bound
FL	Factor Leakage
KL	Kulback-Leiber - Often in the context of KL-Diververgence
LDR	Linear Disentangled Representation
MI	Mutual Information
MIG	Mutual Information Gap
MSE	Mean Square Error
MSS	Minibatch Stratified Sampling
MWS	Minibatch Weighted Sampling
NM-AUC	Non Maximum (Factor Leakage) AUC
PR	Precision Recall
RL	Reinforcement Learning
ROC	Receiver Operating Characteristic
SBDRl	Symmetry Based Disentangled Representation Learning
SVD	Singular Value Decomposition

Conventions

a	Action a by some element $g \in G$
z	Latent Code

z_a	Latent code after applying action a to z
\hat{z}_a	Estimate of z_a predicted by a learning model
x	Data point corresponding to some world space state w
x_a	Data point corresponding to state resulting from applying action a to world space state w
\hat{x}	Reconstruction of data point x by a learning model
β	Weighting of the KL divergence loss term in the β -VAE objective

Groups

C_N	Cyclic Group of order N
D_N	Dihedral group of Order N . The symmetry group of a N sided polygon.
$GL(N)$	General Linear group of order N . Generally can be associated with Nd matrices.
$SO(N)$	Special Orthogonal Group of Order N . Often associated with rotations, i.e. $SO(2)$ is the 2D rotation group.

Models

CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
GBT	Gradient Boosted Tree
MLP	Multi-Layer Perceptron
MMD	Maximum mean discrepancy. Shorthand for MMD-VAE, an infoVae variant.
SVM	Support Vector Machine
Tc	Short hand for BetaTcVAE
VAE	Variational Auto-Encoder

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as: Matthew Painter, Jonathon Hare, and Adam Prügel-Bennett. On the structure of cyclic linear disentangled representations. In *1st NeurIPS workshop on Interpretable Inductive Biases and Physically Structured Learning*, 2020b.
Matthew Painter, Jonathon Hare, and Adam Prugel-Bennett. Linear disentangled representations and unsupervised action estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 13297–13307. Curran Associates, Inc., 2020a. URL <https://proceedings.neurips.cc/paper/2020/file/9a02387b02ce7de2dac4b925892f68fb-Paper.pdf>

Signed:.....

Date:.....

Chapter 1

Introduction

The study of symmetry has a long history in the fields of pure mathematics and theoretical / applied physics. Prior to the mathematical formulation of symmetry via Group theory, its structures were utilised by many mathematicians, notably by Euler in his study of Fermat's little theorem, manifesting as Euler's ϕ function - which forms a group with multiplication modulo n . After its formal introduction, usually accredited to Lagrange, Galois and Cauchy, Group theory went on to become an important aspect of 20th and 21st century mathematics, simplifying many problems in geometry and topology whilst simultaneously becoming a growing field of study in and of itself.

Group theory quickly spread to other fields, in particular physics and chemistry where symmetries are rife. In physics, Noether's theorem famously shows that all physical conservation laws are the direct result of a corresponding physical symmetry, which is defined in group theoretical terms. In chemistry, it was found that chemical molecules can be categorised based on their symmetry properties, again defined in terms of group theoretical concepts.

In a less formal sense, when observing natural scenes, symmetries are present at all levels, be it the bilateral symmetry of human (and many animal) bodies, radial symmetries of many flowers and plants or temporal symmetries in how things move. It is these kinds of ideas that initially motivated [Higgins et al. \[2018\]](#) to consider group theory for the study of disentanglement and disentangled variational representations.

Currently, deep variational methods lack the reconstruction fidelity of other generative models, although partly this is due to their focus not being accurate generative sampling. The main goal of variational models is to produce a smooth and meaningful representation of the data distribution, even if the original space is not smooth itself. Preferably this representation would disentangle the "generative factors" of the data distribution, i.e. render them separable in the representation space.

It is of no surprise then that disentanglement has been the focus of the majority of work in this area in recent years.

Despite this, there is not a strong understanding of exactly how to define or quantify disentanglement, despite a number of metrics that estimate how “disentangled” a given representation would be considered by a human. Such metrics are often based on the ability of a simple classifier to distinguish changes in individual generative factors, or concepts such as the total correlation between latent dimensions. Whilst individually, they stem from an understanding of disentanglement which makes sense relative to the general consensus, they do not all agree, and do not always correlate with common techniques to demonstrate disentanglement (i.e. latent traversals).

The definition by [Higgins et al. \[2018\]](#), stated in group theoretic terms, is the first attempt to rigorously define and study disentanglement, forming the basis and motivation for much of my work. At its broadest level, their notion of disentanglement says that we can relate generative factors to symmetry groups and each group should represent and “act on” independent subspaces of the representation, i.e. generative factors are separated in this space. [Higgins et al. \[2018\]](#) did not explicitly demonstrate this with deep variational models, however, recently this has been done in the simplest case by [Caselles-Dupré et al. \[2019\]](#). They show that to effectively learn in the manner required by Higgins definition, the model needs to observe “transitions”, image pairs which differ by action of one symmetry group. The advantage of the symmetry based definition over those based on concepts such as the total correlation is that for a given symmetry structure, the structure of the expected representation can be defined exactly. This cannot be said of the classical definitions, which encompass a broad range of representational structures.

It is from the basis of symmetry based disentanglement that our work will begin. We will now give an overview of each chapter in this work.

Chapter 1 - Introduction The current chapter, an introduction to the following work

Chapter 2 - Groups and Representations Symmetry based disentangled representation learning uses the language of group and representation theory. This chapter will briefly cover necessary definitions and useful relations that will be useful when discussing SBDRL and models based on the SBDRL framework.

Chapter 3 - Symmetry based Disentangled Representation Learning This chapter will provide an overview of classical and symmetry based disentanglement (SBDRL). Our work will relate to variational auto-encoders (VAEs), so it will focus on disentanglement in that area. It will begin by introducing the original VAE definition, before introducing subsequent improvements to the VAE which aimed to improve disentanglement performance. We will then introduce the standard methods to measure disentanglement and introduce the only large

scale (classical) disentanglement study and its findings. Finally we will introduce the SBDRL framework from which the subsequent chapters shall work.

Chapter 4 - Learning Linear Disentangled Representations This chapter will cover the contents of my NeurIPS 2020 publication ‘Linear Disentangled Representations and Unsupervised Action Estimation’ [Painter et al., 2020a]. We will begin with a feasibility study, which we performed prior to the NeurIPS work which explores learning linear disentangled representations in the ideal case. Section 4.2 will then give a brief overview of ForwardVAE[Caselles-Dupré et al., 2019], which was the first model to explicitly demonstrate linear disentangled representations learnt by a VAE. We will then provide an expanded report of my NeurIPS work which covers Sections 4.3-4.5. Section 4.6 includes experiments present in our NeurIPS paper (Section 4.6.1) and additional exploration of our model.

- Independence, Mean Component Overlap and Factor Leakage Metrics
- Method to supervised learn a linearly disentangled intermediate space from existing representation spaces
- GroupVAE to learn LDR end-to-end without action labels
- Proof that LDRs are sometimes learnt in baseline VAEs, but not consistently

Chapter 5 - Structure and Benefits This chapter will cover and expand upon the contents of our paper ‘On the Structure of Cyclic Linear Disentangled Representations’ published in the 1st NeurIPS 2020 workshop on Interpretable Inductive Biases and Physically Structured Learning [Painter et al., 2020b]. It will begin with an exploration of posterior and prior structure on a simple dataset on which we can vary the symmetry structure. We will then determine if we can find evidence of linear disentangled representations through classical (and symmetry based) disentanglement metric scores. This will lead on to evaluating symmetry based models on two downstream tasks - predicting actions and predicting generative factor values. Finally, we will consider the data efficiency of symmetry based models on these tasks. In particular, looking to correlate efficiency with symmetry based metrics, since it was found by Locatello et al. [2018] that efficiency did not correlate with classical metrics (at least on the dataset and models they tested).

- FlatLand variants
- Method to learn observed structure (the true generative process) rather than the simplest structure, through normalising flows
- Exploration of whether individual disentanglement metric scores can be used to determine if a representation is linearly disentangled or not

- Evidence that LDRs improve downstream scores and data efficiency on factor prediction
- Evidence that LDRs improve data efficiency on action prediction

Chapter 6 - Conclusions This chapter will provide concluding remarks and thoughts on future directions in the area of symmetry based disentangled representation learning.

Chapter 2

Groups and Representations

This chapter will briefly motivate and detail the algebraic structures required to explore the symmetry based variational learning in following chapters. It will begin with the basic definitions and intuitions behind simple groups before progressing to the representation theory that is the basis of the [Higgins et al. \[2018\]](#) definition of disentanglement.

2.1 What is symmetry?

At a young age, most of us are taught about symmetry and lines of symmetry in simple 2D shape such as squares. Often we are tasked with drawing the lines of symmetry through an object to show we understand its symmetries. Despite this, we might struggle to concisely write down an exact definition of what a symmetry is. For the particular case of lines of symmetry (reflectional symmetry, Figure 2.1a) this might be quite simply, *a line through a shape on either side of which is the same shape, reflected, or something similar*. Lines of symmetry aren't the only form of symmetry however, we could also consider rotational symmetry (Figure 2.1b). Again, a definition can be written fairly concisely, *the property of a shape where rotation by a fixed angle results in the same shape*. But there are many more forms of symmetry such as translational, helical, scale, etc. (see Figure 2.2).

If we want to define the term *symmetry* it must encompass all the different forms of symmetry at once. The Oxford English dictionary has a number of definitions, beginning with *The quality of being made up of exactly similar parts facing each other or around an axis*. This covers rotational and reflectional symmetries but not any of the more complex forms listed. A second definition provided is *Similarity or exact correspondence between different things*. This is more encompassing, however it is

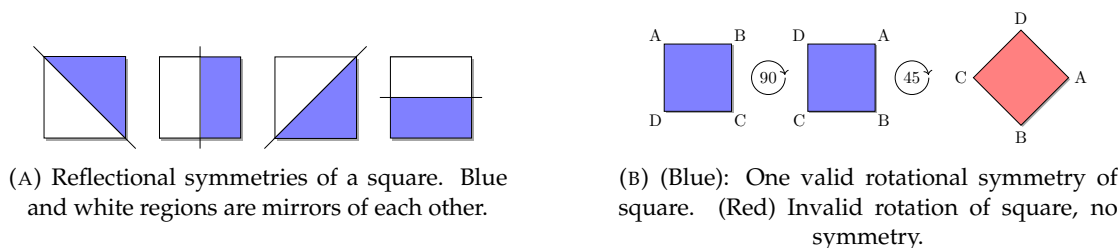


FIGURE 2.1: Symmetries of the Square.

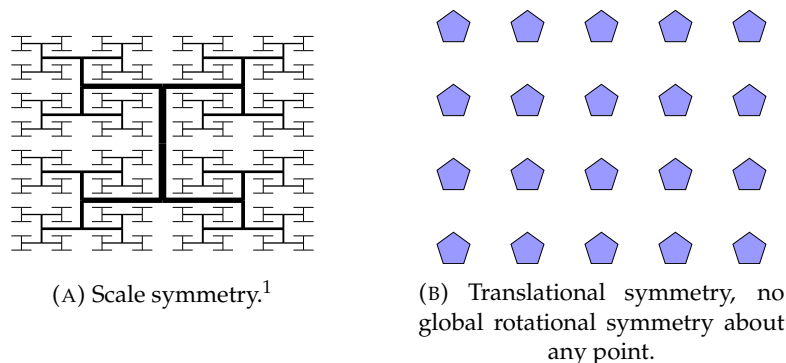


FIGURE 2.2: Other examples of symmetries or patterns with symmetries.

perhaps too vague in that it doesn't lead you to be able to define any of the particular forms.

Similar problems led mathematicians to algebraic structures which have the flexibility to describe symmetry properties. As mentioned in the Introduction, this area became known as *Group Theory* and the algebraic structures which describe symmetries became known as *Groups*.

2.2 Group Theory

We begin with the definition of a Group.

Definition 2.1. A **Group** is a pair (G, \cdot) of a set G and binary relation $\cdot : G \times G \rightarrow G$ which satisfies the following,

- Closure: $g_1 \cdot g_2 \in G \quad \forall g_1, g_2 \in G$
- Associativity: $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3) \in G \quad \forall g_1, g_2, g_3 \in G$
- Identity: $\exists I \in G$ s.t. $I \cdot g = g \cdot I = g \quad \forall g \in G$. Often we denote $1 := I$.
- Inverse: $\forall g \in G, \exists g^{-1} \in G$ s.t. $g^{-1} \cdot g = g \cdot g^{-1} = I$

Remark 2.2. Often we refer to the group (G, \cdot) simply as G for brevity.

¹Source: <http://texample.net/tikz/examples/tag/fractals/>

Introductory group theory courses often follow this definition with an illustrating example, commonly the group of symmetries of the square². We shall follow suit.

To form a group we need to enumerate (label) elements of the set G . To do this for the group of symmetries of the square we introduce some notation. In Figure 2.1b, viewing the first square, we have labelled each corner (A, B, C, D) ³. We can now denote each symmetry by how it affects these labels. Note that valid symmetries are only those that result in the square looking exactly the same (disregarding the corner labels), thus the red square in Figure 2.1b is invalid since it is orientated like a diamond, not a square.

Lets first enumerate the reflectional symmetries in Figure 2.1a. From left to right we label, $\sigma_0 := (A, D, C, B)$, $\sigma_1 := (B, A, D, C)$, $\sigma_2 := (C, B, A, D)$ and $\sigma_3 := (D, C, B, A)$. We now denote the anti-clockwise rotations by 0, 90, 180 and 270 degrees by $\rho_0, \rho_1, \rho_2, \rho_3$ respectively. In (A, B, C, D) notation, these cycle the labels through the list backwards, *e.g.* $\rho_1 = (B, C, D, A)$ and $\rho_2 = (C, D, A, B)$.

Note that if we applied ρ_1 twice, we would achieve the same result as ρ_2 , $\rho_1 \cdot \rho_1 = \rho_2$. Similarly, for the square, if we applied ρ_1 4 times we would return to the same position, *i.e.* it is the identity. It is often useful to denote applying the same group element to itself by powers, an example in this case, $\rho_1^4 = 1$, this is known as the order.

Definition 2.3. The **order** of a group element $g \in G$ is given by,

$$|g| = n \in \mathbb{Z} \quad \text{such that} \quad g^n = 1_G \quad \text{where } n \text{ is minimal} \quad .$$

Remark 2.4. Reflections σ_i all have order 2, except the identity element, which has order 1.

Having enumerated each valid symmetry operation on the square we now need to define the binary relation \cdot on G . We can use our previously defined notation to consider an example before presenting the operation table. Consider the rotation $\rho_3 = (D, A, B, C)$ and the reflection $\sigma_0 = (A, D, C, B)$, then the composition of first applying the rotation and then the reflection, $\sigma_0 \cdot \rho_3 = (D, C, B, A)$ is equal to the reflection σ_3 (see Figure 2.3). This demonstrates the closure property for this particular pair of elements. From the operation table (Table 2.1a) we can see that the operation as defined over the corner labels is indeed closed. We could also demonstrate the other properties of the group if desired.

Having introduced a simple (and visual) example of a group, we will now use this to demonstrate some other group theoretical concepts.

²Usually this group is denoted D_4 , the Dihedral group of order 4.

³This notation is similar to Cauchy's two line notation for permutations [Wussing, 2007, p.94], although not the one line version.

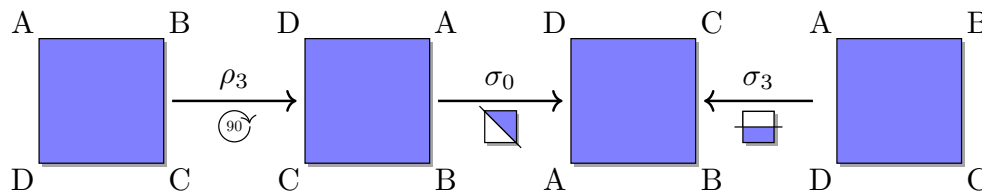


FIGURE 2.3: Visual example of the composition $\sigma_0 \cdot \rho_3 = \sigma_3$

\cdot	ρ_0	ρ_1	ρ_2	ρ_3	σ_0	σ_1	σ_2	σ_3
ρ_0	ρ_0	ρ_1	ρ_2	ρ_3	σ_0	σ_1	σ_2	σ_3
ρ_1	ρ_1	ρ_2	ρ_3	ρ_0	σ_1	σ_2	σ_3	σ_0
ρ_2	ρ_2	ρ_3	ρ_0	ρ_1	σ_2	σ_3	σ_0	σ_1
ρ_3	ρ_3	ρ_0	ρ_1	ρ_2	σ_3	σ_0	σ_1	σ_2
σ_0	σ_0	σ_3	σ_2	σ_1	ρ_0	ρ_3	ρ_2	ρ_1
σ_1	σ_1	σ_0	σ_3	σ_2	ρ_1	ρ_0	ρ_3	ρ_2
σ_2	σ_2	σ_1	σ_0	σ_3	ρ_2	ρ_1	ρ_0	ρ_3
σ_3	σ_3	σ_2	σ_1	σ_0	ρ_3	ρ_2	ρ_1	ρ_0

(A) Symmetry group of squares

\cdot	ρ_0	ρ_1	ρ_2	ρ_3
ρ_0	ρ_0	ρ_1	ρ_2	ρ_3
ρ_1	ρ_1	ρ_2	ρ_3	ρ_0
ρ_2	ρ_2	ρ_3	ρ_0	ρ_1
ρ_3	ρ_3	ρ_0	ρ_1	ρ_2

(B) Subgroup $(\{\rho_0, \rho_1, \rho_2, \rho_3\}, \cdot)$

TABLE 2.1: Binary operation tables for symmetry group of squares and the subgroup of rotations under $\cdot : G \times G \rightarrow G$.

Definition 2.5. A **subgroup** H of G denoted $H \leq G$ is a subset H of G such that (H, \cdot) also forms a group.

Remark 2.6. It can be shown that if $H \subseteq G$ then (H, \cdot) is a subgroup if and only if it is closed, contains the identity (on H) and contains inverses. *i.e.* we don't need to check associativity.

There is a nice example of a subgroup in our symmetry group of the square, comprised of the rotations. Intuitively we might expect this since it's visually obvious that if we compose any two rotations then we get another rotation. Similarly the identity (rotation by 0 degrees) and inverses (rotations by $360 - X$ degrees) are also rotations. We can actually see this in the operation table for this subset in Table 2.1b since it contains no reflections (closed), $\rho_0 \cdot g = g$ for any g in the subset (identity) and ρ_0 appears on every row (inverses). There are also a number of other subgroups that we can define, such as $(\{\rho_0, \rho_2\}, \cdot)$ however they don't add much more insight.

Note that Table 2.1a is not symmetric about the leading diagonal. This means that the operation \cdot is not commutative, *i.e.* $g_1 \cdot g_2 \neq g_2 \cdot g_1 \quad \forall g_1, g_2 \in G$. However, Table 2.1b is symmetric, and so \cdot does commute for the subgroup $(\{\rho_0, \rho_1, \rho_2, \rho_3\}, \cdot)$.

Definition 2.7. We call a group (G, \cdot) **abelian** if $g_1 \cdot g_2 = g_2 \cdot g_1 \quad \forall g_1, g_2 \in G$. *i.e.* G is abelian if \cdot is commutative over the whole group.

Remark 2.8. The subgroup $(\{\rho_0, \rho_1, \rho_2, \rho_3\}, \cdot)$ is abelian.

Along with the fact that the subgroup of rotations is abelian, it is also an instance of a cyclic group. Cyclic groups are important to later chapters and will be discussed in more detail later on in this chapter.

So far we have covered groups and operations from groups to themselves. We now consider operations which act on a group and an arbitrary set. This kind of mapping is known as a group action.

Definition 2.9. A (left) **group action** ϕ of (group) G on set X is a mapping

$$\phi : G \times X \rightarrow X, (g, x) \mapsto \phi(g, x) \quad \text{for } g \in G, x \in X,$$

which satisfies,

- Identity: $\phi(1_G, x) = x \quad \forall x \in X.$
- Compatibility: $\phi(g \cdot h, x) = \phi(g, \phi(h, x)) \quad \forall g, h \in G, x \in X$

For simplicity we will generally denote an action by $g \circ x = \phi(g, x).$

Remark 2.10. A simple case of an action is when a group acts on itself, *i.e.* $X = G$. In this case the group operation is also an action, $g_1 \circ g_2 = g_1 \cdot g_2$. Actions are sometimes denoted by \cdot , the same as the group operation, and it is when $X = G$ that this makes the most sense. In the general case however, $X \neq G$, and so we will avoid using \cdot to denote an action.

Remark 2.11. Actions can be extended to act on spaces with structure, such as a vector space V , as long as they preserve the appropriate properties. For example, an action on a vector space is required to preserve linearity,
 $g(\alpha x + \beta y) = \alpha(g \circ x) + \beta(g \circ y), \quad g \in G, x, y \in V, \alpha, \beta \in \mathbb{R}.$

We can now have a look at how the symmetry group of the square acts on the set of labels, one simple action that we can define. We know that the possible labellings of the square are a subset of $X = \{(x_1, x_2, x_3, x_4) \mid x_i \in \{A, B, C, D\}, x_i \neq x_j, i \neq j\}$. We can now define actions by the effect on the labels, if they were placed on a square. For example, we can consider a rotation acting on (A, B, C, D) by $\rho_1 \circ (A, B, C, D) = (B, C, D, A)$, or a reflection $\sigma_0 \circ (A, B, C, D) = (A, D, C, B)$.

Definition 2.12. The **orbit** of an element $x \in X$ under group G is the path that it takes when acted upon by G , defined by,

$$G(x) = \{g \circ x \mid g \in G\}$$

Returning to our previous example of an action, consider the orbit of (A, B, C, D) under G . The result is the set of all states that can be reached under a single symmetry

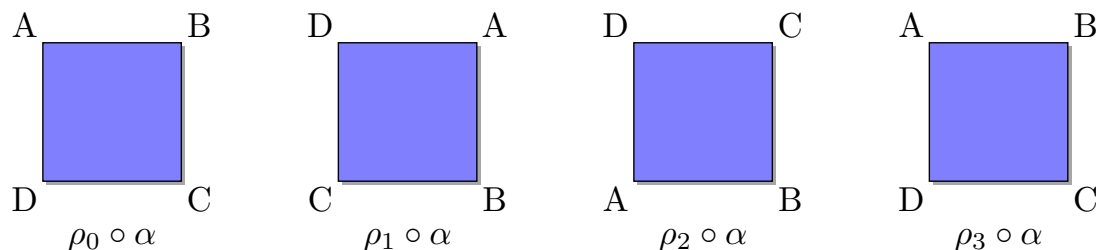


FIGURE 2.4: Visual orbit of $\alpha = (A, B, C, D)$ under subgroup of rotations $\{\rho_0, \rho_1, \rho_2, \rho_3\}$.

operation. A more interesting example might be the orbit of $\alpha := (A, B, C, D)$ under the subgroup of rotations. This results in the set of all rotated versions of the square, $\{\rho_0 \circ \alpha = \alpha, \rho_1 \circ \alpha = (B, C, D, A), \rho_2 \circ \alpha = (C, D, A, B), \rho_3 \circ \alpha = (D, A, B, C)\}$. Orbits of this kind will become interesting to study in later chapters where we can demonstrate the quality of *learnt* representations of actions.

A final structure that will become useful later is the direct product of groups, which allows construction of a single group from several smaller groups.

Definition 2.13. The **direct product** $G \times H$ of groups G and H , is a group where,

- The underlying set is defined as the Cartesian product of the sets G and H , $(g, h) \in G \times H, g \in G, h \in H$.
- The group operation is defined element-wise as the corresponding operation for each group, $(g_1, h_1) \cdot (g_2, h_2) = (g_1 \cdot_G g_2, h_1 \cdot_H h_2)$.

Remark 2.14. $G \times H$ has G and H as subgroups since $G \times 1$ and $1 \times H$ behave in the same way as groups G and H .

Before progressing onto the representation theory of groups, it is useful to cover some other simple groups that we will make use of throughout this work. We will begin with cyclic groups, which we have already seen an example of.

Definition 2.15. The **cyclic group** C_N of order N is given by

$$C_N = \langle g \rangle = \{g^n \mid g^N = 1, g^i \neq g^j, i \neq j\} \quad ,$$

where g is called the *generator* of C_N .

Remark 2.16. An oft considered example of a cyclic group is the group of integers modulo N , denoted $(\mathbb{Z}/N\mathbb{Z}, +)$. The generators of this group are 1 and -1 .

$$\mathbb{Z}/N\mathbb{Z} = \{0, \dots, N-1\}$$

Example 2.1. Recall the subgroup of rotations that we considered earlier. As mentioned, this is in fact an example of a cyclic group of order 4, C_4 , with generator $g = \rho_1$ and identity $1 = \rho_0$.

Another particularly useful group that we will require later on is the General Linear group $GL_n(\mathbb{R})$.

Definition 2.17. The **General Linear** group $GL_n(\mathbb{R})$ is the set of invertible $n \times n$ matrices whose elements have values in \mathbb{R} .

Remark 2.18. There exists a subgroup of the general linear group known as the Special linear group, the group of all invertible $n \times n$ matrices with determinant 1.

Two groups with obvious applications to vision are the 2D and 3D rotation groups, $SO(2)$ and $SO(3)$, we will introduce the general case $SO(n)$.

Definition 2.19. The general orthogonal group $O(n)$ is the group of all distance preserving transformations (with respect to a defined basis), *group of $n \times n$ orthogonal matrices*. The **Special Orthogonal** group $SO(n)$ is then the subgroup of $O(n)$ with all matrices of determinant 1.

$$SO(n) = \{A \in GL(n) \mid AA^T = A^T A = I, |A| = 1\}$$

Remark 2.20. The set of all distance preserving transformations is the set of rotations and reflections.

Remark 2.21. $SO(2)$ is equivalent to the circle group $T = \{z \in \mathbb{C} \mid |z| = 1\}$ and, unlike higher degree SO groups, is abelian.

A final class of group that might be of interest is the Dihedral group, one instance of which is the group of symmetries of the square.

Definition 2.22. The **Dihedral Group** D_n is the symmetry group of an n -sided regular polygon. It can be considered as being generated by a rotation (of order n) and a reflection (of order 2).

$$D_n = \langle \rho, \sigma \mid \sigma^2 = 1, \rho^n = 1, (\sigma \cdot \rho)^2 = 1 \rangle$$

Remark 2.23. We can see an example of this final condition in Figure 2.3.

Remark 2.24. This is another example of so called generator notation⁴ with angled braces. In this case it means the set of powers of the elements and powers of the products of the elements.

2.3 Representation Theory

In the visual example given in the previous section it was very easy to explore the group and its properties since we can always imagine the operations happening on a

⁴Also called a group presentation

real object. Generally it is not nearly as easy to visualise structures in other groups without significant experience and time working with them. On the other hand, linear algebra is a much easier area to grasp, since it deals with much more concrete objects, linear functions, vectors and matrices.

Representation theory provides the framework necessary to explore concepts in algebraic group theory purely by dealing with structures from linear algebra. Group elements are “represented” by matrices and the group operation then becomes matrix multiplication (or addition). Since representations allow group problems to be reduced to corresponding linear algebra problems and linear algebra is well understood, it has led mathematicians to some notable results (e.g. [Burnside, 1904]) that are complicated greatly by solely group theoretic principles.

Definition 2.25. A **representation** of group G (or **group representation**) is a mapping,

$$\rho : G \rightarrow GL(V) \quad ,$$

which satisfies:

$$\rho(g_1 \cdot g_2) = \rho(g_1) \cdot \rho(g_2) \quad \forall g_1, g_2 \in G \quad .$$

Remark 2.26. This mapping is called a group homomorphism which preserves the identity and inverse. These are easy to show:

$$1 = \rho(1)^{-1}\rho(1) = \rho(1)^{-1}\rho(1 \cdot 1) = \rho(1)^{-1}\rho(1) \cdot \rho(1) = \rho(1) \quad .$$

$$\rho(g)^{-1} = \rho(1)\rho(g)^{-1} = \rho(g^{-1} \cdot g) \cdot \rho(g)^{-1} = \rho(g^{-1}) \cdot \rho(g) \cdot \rho(g)^{-1} = \rho(g^{-1}) \quad .$$

Definition 2.27. We call a representation ρ **faithful** if it is injective. *i.e.*

$\{g \mid \rho(g) = 1\} = \{1 \in G\}$, the kernel is equal to the identity.

We will mostly be concerned with real representations where $V = \mathbb{R}^n$. In this setting, faithful representations allow us to view our group G as a subgroup of $GL_n(\mathbb{R})$, resulting in group elements being mapped to invertible matrices.

Example 2.2. Let us consider representations of our example from the previous section, the symmetry group of the square. Since the square is two dimensional, we can look for representations on \mathbb{R}^2 , standard 2D matrices. Recall that symmetries of the square are either rotations or reflections. The rotation and reflection matrices are well known to computer vision practitioners due to the heavy use of affine transforms in, for example, image alignment. The rotation of 90 degrees ($\frac{2\pi}{4}$ radians) and reflection about x-axis matrices are given by:

$$P = \begin{pmatrix} \cos\left(\frac{2\pi}{4}\right) & -\sin\left(\frac{2\pi}{4}\right) \\ \sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

We can now relate our known group elements in the symmetry group of the square to elements of $GL_2(\mathbb{R})$ through these two matrices. For example, $\rho(\rho_i) = P^i$, $\rho(\sigma_i) = \Sigma \cdot P^i$. Note that the

use of ρ for both the group elements ρ_i and the representation ρ is notational/conventional only and represents no general relationship.

It is often useful to consider representations in terms of actions, and indeed, this is the most useful interpretation for later chapters. In this setting, a representation $\rho : G \rightarrow GL(V)$ can be thought about through the action $\circ : G \times V \rightarrow V, (g, v) \mapsto \rho(g)(v)$. We will often see that it is much easier to find and think about individual $\rho(g)$ rather than ρ itself. The previous example perfectly exemplifies how it can be easy to reason about $\rho(g)$ but the general mapping ρ is less intuitive.

Definition 2.28. The **degree** or **dimension** of the representation is given by the dimension of V . *i.e.* If $V = \mathbb{R}^n$ then the dimension of a representation on this space is n .

Recall that for groups we have a notion of a subgroup that is contained inside a group. In the same sense there exists the notion of a subrepresentation that exists inside a representation.

Definition 2.29. Given a representation $\rho : G \rightarrow GL(V)$, then a **subrepresentation** of ρ is a subspace W of V such that,

$$\rho(g)(w) \in W \quad \forall g \in G, w \in W \quad .$$

This is known as W being invariant under the group action \circ .

Remark 2.30. The trivial examples for W are the zero subspace $\{0\}$ and V itself.

Definition 2.31. A representation ρ is known as **irreducible** if there exist no subrepresentations other than the trivial representation.

We will see later that it is important to consider representations as combinations of irreducible subrepresentations. Since subrepresentations are of lower dimension (although not strictly) than the main representation, we need ways of combining lower dimensional representations into larger dimensional ones. For us there exist two important operations, the direct sum \oplus and the tensor product \otimes .

To understand these operations we first must define them on vector spaces.

Definition 2.32. The **sum** of $V_1, V_2 \subseteq V$ is given by,

$$V_1 + V_2 = \{v_1 + v_2 \mid v_1 \in V_1, v_2 \in V_2\} \quad .$$

This a **direct sum** denoted $V = V_1 \oplus V_2$ iff $\forall v \in V, \exists$ unique $v_1 \in V_1, v_2 \in V_2$ such that $v = v_1 + v_2$

Remark 2.33. The direct sum of (finite dimensional) vector spaces V and W with bases $\{v_1, \dots, v_n\}$ and $\{w_1, \dots, w_m\}$ has dimension $n + m$ and basis $\{v_1, \dots, v_n, w_1, \dots, w_m\}$.

Definition 2.34. Given two representations $\rho_1 : G \rightarrow GL(V)$ and $\rho_2 : G \rightarrow GL(W)$, then their **direct sum** is a representation $\rho_1 \oplus \rho_2 : G \rightarrow GL(V \oplus W)$, defined by,

$$\rho_1 \oplus \rho_2(g)(v, w) = (\rho_1(g)(v), \rho_2(g)(w))$$

Example 2.3. Returning to the case of D_4 , we can consider two representations of the rotation subgroup, π and ρ which represent rotation matrices in the clockwise and anticlockwise directions.

$$\rho(\rho_i) = \begin{pmatrix} \cos\left(\frac{2\pi}{4}\right) & -\sin\left(\frac{2\pi}{4}\right) \\ \sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \end{pmatrix}^i \quad \text{and} \quad \pi(\rho_i) = \begin{pmatrix} \cos\left(\frac{2\pi}{4}\right) & \sin\left(\frac{2\pi}{4}\right) \\ -\sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \end{pmatrix}^{4-i}$$

Obviously if we evaluate the individual trigonometric functions then the resulting matrices are the same for each rotation, but we shall leave them unevaluated so we can observe the result of the direct product more explicitly. In this case the direct sum is given by,

$$\rho(\rho_i) \oplus \pi(\rho_j) = \begin{pmatrix} \begin{pmatrix} \cos\left(\frac{2\pi}{4}\right) & -\sin\left(\frac{2\pi}{4}\right) \\ \sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \end{pmatrix}^i & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \begin{pmatrix} \cos\left(\frac{2\pi}{4}\right) & \sin\left(\frac{2\pi}{4}\right) \\ -\sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \end{pmatrix}^{4-j} \end{pmatrix}$$

In general it is true that for representations $\rho : G \rightarrow GL_n(\mathbb{R})$ and $\pi : G \rightarrow GL_m(\mathbb{R})$ that the direct sum of group elements $\rho(g)$ and $\pi(g)$ with matrices A and B is,

$$\rho(g) \oplus \pi(g) = \begin{pmatrix} A & \mathbf{0} \\ \mathbf{0} & B \end{pmatrix}$$

Before giving a formal definition for the tensor product of two vector spaces, we will give the result for two (finite dimensional) vectors v, w in \mathbb{R}^n and \mathbb{R}^m .

Definition 2.35. For vector spaces V and W with known bases then for two (row) vectors $v = (v_1, \dots, v_n) \in V$ and $w = (w_1, \dots, w_m) \in W$, their **tensor product** is the outer product:

$$v \otimes w = v^T w = \begin{pmatrix} v_1 w_1 & v_1 w_2 & \dots & v_1 w_m \\ v_2 w_1 & v_2 w_2 & \dots & v_2 w_m \\ \vdots & \vdots & \ddots & \vdots \\ v_n w_1 & v_n w_2 & \dots & v_n w_m \end{pmatrix}$$

Remark 2.36. Given this definition of tensor products, we can see that when equating the equivalence relation to the usual equality in $\mathbb{R}^{m \times n}$, all these conditions hold. *i.e.* for

distributivity of addition,

$$(v, w)_{ij} + (v', w)_{ij} = v_i w_j + v'_i w_j = (v_i + v'_i) w_j = (v + v', w)_{ij}$$

Remark 2.37. For given bases $\{v_1, \dots, v_n\}$ and $\{w_1, \dots, w_m\}$ of V and W , then $\{v_i \otimes w_j \mid i \in \{1, \dots, n\}, j \in \{1, \dots, m\}\}$ is a basis for $V \otimes W$

Definition 2.38. The tensor product of representations $\rho_1 : G \rightarrow GL_n(\mathbb{R})$ and $\rho_2 : G \rightarrow GL_m(\mathbb{R})$ is a representation $\rho_1 \otimes \rho_2 : G \rightarrow GL(V \otimes W)$, given by,

$$(\rho_1 \otimes \rho_2)(g)(v \otimes w) = \rho_1(g)(v) \otimes \rho_2(g)(w)$$

Example 2.4. Considering the same example as Example 2.2, subgroups of D_4 ,

$$\rho(\rho_i) = \begin{pmatrix} \cos\left(\frac{2\pi}{4}\right) & -\sin\left(\frac{2\pi}{4}\right) \\ \sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \end{pmatrix}^i \quad \text{and} \quad \pi(\rho_i) = \begin{pmatrix} \cos\left(\frac{2\pi}{4}\right) & \sin\left(\frac{2\pi}{4}\right) \\ -\sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \end{pmatrix}^{4-i},$$

then,

$$\rho(\rho_1) \otimes \pi(\rho_3) = \begin{pmatrix} \cos\left(\frac{2\pi}{4}\right) \cos\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \sin\left(\frac{2\pi}{4}\right) & -\sin\left(\frac{2\pi}{4}\right) \cos\left(\frac{2\pi}{4}\right) & -\sin\left(\frac{2\pi}{4}\right) \sin\left(\frac{2\pi}{4}\right) \\ \cos\left(\frac{2\pi}{4}\right) - \sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \cos\left(\frac{2\pi}{4}\right) & -\sin\left(\frac{2\pi}{4}\right) - \sin\left(\frac{2\pi}{4}\right) & -\sin\left(\frac{2\pi}{4}\right) \cos\left(\frac{2\pi}{4}\right) \\ \sin\left(\frac{2\pi}{4}\right) \cos\left(\frac{2\pi}{4}\right) & \sin\left(\frac{2\pi}{4}\right) \sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \cos\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \sin\left(\frac{2\pi}{4}\right) \\ \sin\left(\frac{2\pi}{4}\right) - \sin\left(\frac{2\pi}{4}\right) & \sin\left(\frac{2\pi}{4}\right) \cos\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) - \sin\left(\frac{2\pi}{4}\right) & \cos\left(\frac{2\pi}{4}\right) \cos\left(\frac{2\pi}{4}\right) \end{pmatrix}$$

Again, in general, for representations $\rho : G \rightarrow GL_n(\mathbb{R})$ and $\pi : G \rightarrow GL_m(\mathbb{R})$ the tensor product of group elements $\rho(g)$ and $\pi(g)$ represented with matrices A and B is

$$\rho(g) \otimes \mu(g) = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ a_{21}B & a_{22}B & \dots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \dots & a_{nm}B \end{pmatrix}$$

Remark 2.39. We now have methods to combine representations. There are two important relations that we shall use in future chapters. The first is that if some representation V is reducible then we can reduce it into the direct sum of irreducible representations. $V = V_1 \oplus V_2 \oplus \dots \oplus V_m$. The second is that if some group G is a direct product of groups G_1, G_2 then the irreducible representations of G are the representations $\rho_1 \otimes \rho_2$ where ρ_1 is an irreducible representation of G_1 and ρ_2 is an irreducible representation of G_2 .

We end by giving examples of irreducible representations for common groups some of which will be useful in later chapters.

Example 2.5. *The irreducible representations of C_N are finite in number and one of*

$$\rho(g^n) = e^{\frac{2\pi i}{N}n} \quad \text{or} \quad \rho(g^n) = \begin{pmatrix} \cos \frac{2\pi n}{N} & -\sin \frac{2\pi n}{N} \\ \sin \frac{2\pi n}{N} & \cos \frac{2\pi n}{N} \end{pmatrix}$$

depending on if we are on the field \mathbb{C} or \mathbb{R} respectively.

Example 2.6. *The irreducible representations of $SO(2)$ are infinite in number and one of*

$$\rho(g_\theta) = e^{i\theta} \quad \text{or} \quad \rho(g_\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

depending on if we are on the field \mathbb{C} or \mathbb{R} respectively.

Example 2.7. *The irreducible representation of $SO(3)$ are infinite in number and relate to the spherical harmonics.*

Example 2.8. *The irreducible representations of D_n are real and there are $\frac{n-2}{n}$ two-dimensional representations, each indexed by k and given by,*

$$\rho(\rho) = \begin{pmatrix} \cos \frac{2\pi k}{N} & -\sin \frac{2\pi k}{N} \\ \sin \frac{2\pi k}{N} & \cos \frac{2\pi k}{N} \end{pmatrix} \quad \rho(\sigma) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (2.1)$$

There also exist a number of 1 dimensional representations, however they are not faithful and will not be particularly useful/informative for our case.

Chapter 3

Symmetry Based Disentangled Representation Learning (SBDRL)

This chapter will introduce disentanglement and the symmetry based disentanglement framework as set out by [Higgins et al. \[2018\]](#).

To begin, we will aim to describe the classical or standard deep learning approach to disentanglement. First we will introduce the VAE framework, since all the models we cover in this work will make use of it. Subsequently we will give a high level overview of what disentanglement is, and what ‘generative factors’ are. We will then introduce a number of VAE models which were designed to produce disentanglement through the VAE framework. In general, this will be achieved through learning a latent space, where each latent dimension relates to one generative factor of the dataset. Later in this work, we will want to compare disentanglement of VAE models, for which we will use a number of proposed disentanglement metrics. Such metrics often require access to the generative factors, so before defining metrics, we define standard datasets for disentanglement research, and their generative factors. Alongside the metrics, we also describe common empirical demonstrations that help provide us intuition on whether a representation is disentangled or not.

Having the foundation of classical disentanglement, we then move on to describing symmetry based disentanglement as per [Higgins et al. \[2018\]](#), working towards the main definitions of linear disentanglement (Definitions 3.7 and 3.9).

3.1 Variational Auto Encoders

Variational auto-encoders are often used to explore disentanglement in deep learning, and will be used throughout this work. As such, we need to introduce the model first

proposed by Kingma and Welling [2014]. Rather than following the original work, we will follow the intuitive introduction to VAEs that is provided by Doersch [2016].

VAEs are a particular form of generative model which aim to model a dataset X consisting of data points x . It is assumed that the data are generated by some random process depending on unobserved continuous random variable z . The generative process of Kingma and Welling [2014] has two steps, first a sample z is drawn from a prior distribution $p_{\theta^*}(z)$, before a sample x is drawn from $p_{\theta^*}(x|z)$.

VAEs, as auto-encoders, in implementation have an encoder and a decoder. For this section, it may be useful to keep in mind that the estimated generative parameters θ are the parameters of the decoder network whilst the inference parameters ϕ are those of the encoder network.

The aim of the VAE is to learn to approximate the true generative parameters θ^* whilst also learning to approximate posterior inference of x under the estimated generative parameters θ . In order to estimate parameters θ^* , the VAE maximises the likelihood of the data points x under the parameters θ ,

$$p_{\theta}(x) = \int p_{\theta}(x, z) dz = \int p_{\theta}(x|z)p(z) dz \quad (3.1)$$

It is assumed that maximising the likelihood of real data points will result in the model generating data that appears similar to real samples. For image problems with generative process $f : \mathcal{Z} \times \theta \rightarrow X$, Kingma and Welling [2014] set $p_{\theta}(x|z)$ to either a multivariate Gaussian $\mathcal{N}(X|f(z; \theta), \text{diag}(\sigma^2))$ for real valued data¹ or Bernoulli $B(f(z; \theta))$ for binary data.

Note that in general, and for most z , we expect that $p_{\theta}(x|z)$ will be very small, since assuming the dataset is of non-trivial size, the inference function will be fairly complex, with varied outputs. As such, an accurate estimate of $p(x)$ will require a large number of samples of z . Certainly, we could not approximate it with a single sample, as would be desired for fastest estimation. One way to avoid having to sample a large number of z is to introduce a model $q_{\phi}(z|x)$, to estimate the space of z which is likely to produce inference outputs similar to x . Introducing q and parametrising it with a neural network (with parameters ϕ) is a core concept behind VAEs.

Recalling that we want to maximise $p_{\theta}(x)$, we can relate it to $q_{\phi}(z|x)$ by examining the KL-Divergence [Kullback and Leibler, 1951],

¹Generally the hyperparameter σ^2 is just set to 0.5.

$$\begin{aligned}
D_{KL} [q_\phi(z|x)||p_\theta(z|x)] &= \mathbb{E}_{z \sim q_\phi} [\log q_\phi(z|x) - \log p_\theta(z|x)] \\
&= \mathbb{E}_{z \sim q_\phi} [\log q_\phi(z|x) - \log p_\theta(x|z) - \log p_\theta(z)] + \log p_\theta(x) \\
&= \mathbb{D}_{KL} [q_\phi(z|x)||p_\theta(z)] - \mathbb{E}_{z \sim q_\phi} [\log p_\theta(x|z)] + \log p_\theta(x) \quad .
\end{aligned}$$

Following this, we can write,

$$\log p_\theta(x) - D_{KL} [q_\phi(z|x)||p_\theta(z|x)] = \mathbb{E}_{z \sim q_\phi} [\log p_\theta(x|z)] - D_{KL} [q_\phi(z|x)||p_\theta(z)] \quad . \tag{3.2}$$

This allows a lower bound on $\log p_\theta(x)$ (since $D_{KL} [q_\phi(z|x)||p_\theta(z|x)] \geq 0$), which is known as the **Evidence Lower Bound (ELBO)**,

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi} [\log p_\theta(x|z)] - D_{KL} [q_\phi(z|x)||p_\theta(z)] \quad .$$

Also from Equation 3.2, we can see that if we can optimise the right hand side then we are optimising the log evidence alongside the KL-Divergence between the inferred posterior $q_\phi(z|x)$ and the true posterior $p_\theta(z|x)$. For a sufficiently complex q_ϕ it should be possible to optimise this to 0.

In order to optimise the right hand side we need to define a prior for z . In particular, it would be useful to choose one which makes the divergence against $q_\phi(z|x)$ easily calculable. For this reason it is usual to choose $p_\phi(z) = \mathcal{N}(z|0, \text{diag}(1))$, in which case, we have a simple, k -dimensional Gaussian-Gaussian divergence, with closed form solution,

$$\begin{aligned}
D_{KL} [\mathcal{N}(\mu_0, \Sigma_0)||\mathcal{N}(\mu_1, \Sigma_1)] &= \frac{1}{2} \left(\text{trace} \left(\Sigma_1^{-1} \Sigma_0 \right) + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) \right. \\
&\quad \left. - k + \log(\det \Sigma_1) - \log(\det \Sigma_0) \right)
\end{aligned}$$

Simplifying further since our prior has 0 mean and covariance matrix equal to the identity and assuming $q_\phi(z|x)$ has diagonal covariance,

$$D_{KL} [\mathcal{N}(\mu_0, \text{diag}([\sigma_i^2])||\mathcal{N}(0, \text{diag}(1)))] = -\frac{1}{2} \sum_{i=1}^k \left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2 \right)$$

Alongside this we also need to be able to compute $\mathbb{E}_q [\log p_\theta(x|z)]$. We could get a good estimate of this by repeatedly sampling z from q_ϕ and the computing $P(X|z)$ however this would be expensive. We can see from Fig. 3.1 that this would require multiple passes through a (computationally expensive) decoder network. Instead it is

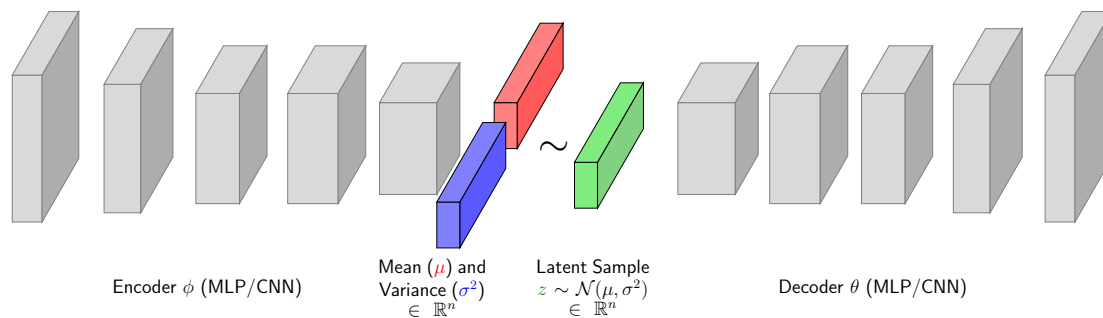


FIGURE 3.1: Simple VAE schematic.

usual to simply approximate this with a single sample from q_ϕ , assuming that it gives a decent estimate.

Even if we decide to approximate the expectation, we still have the problem that we would need gradient for it. Standard automatic differentiation programs cannot compute gradient through a random sampling operation. To avoid this limitation, Kingma and Welling [2014] propose the “reparametrisation trick” which allows good gradient estimation from the single Monte-Carlo sample².

The reparametrisation trick simply samples a stochastic auxiliary variable ϵ rather than sampling directly from $q_\phi(z|x) = \mathcal{N}(z|\mu(x), \sigma(x))$. This allows us to sample $z \sim q_\phi(z|x)$ by $z = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$.

Generally we either assume pixels are distributed normally or Bernoulli, which allows us to compute $\log p_\theta(x|z)$ by the mean squared error (Gaussian, with unit variance) or binary cross entropy (Bernoulli). As such, by reparametrization and our KL divergence estimate, we can compute and optimise all terms in the ELBO, allowing us to train a VAE model using auto-grad methods.

3.2 Disentanglement as a Concept

Disentanglement (particularly in VAEs) has been difficult to define, hence the efforts by Higgins et al. [2018] to provide a mathematical basis, which we will discuss later in the chapter. In this section we will provide some intuition on what to expect from disentangled representations, and introduce models which aim to produce them. We will then discuss some standard approaches to measure them.

The concept of disentangled representations was known before the introduction of VAEs by Kingma and Welling [2014]. Generally, the initial introduction is accredited to Bengio et al. [2013] for the following high level description of what to expect from disentangled representations.

²You could define other gradient estimators which do not require reparametrisation, however they can be of higher variance and require manual implementation in auto-grad methods.

“Different explanatory factors of the data tend to change independently of each other in the input distribution, and only a few at a time tend to change when one considers a sequence of consecutive real-world inputs . . . [we desire to] leverage the data itself, using vast quantities of unlabelled examples, to learn representations that separate the various explanatory sources.” (Bengio et al. [2013])

Generally, this quotation and disentanglement is interpreted to mean that individual latent units are sensitive to changes in individual generative factors, and not the others. We will introduce generative factors more rigorously in the next section. Disentanglement can be intuitively described in terms an inverse graphics problem, and we briefly consider this as introduced by Kulkarni et al. [2015]. In computer graphics, renders often map from scene descriptions encoded into a dense vector to an image of the scene. Generally the encoded scene will separate the notions of object location, pose, lighting etc. This allows fine grained control over the scene. The problem of inverse graphics is then how to infer from a (possibly) rendered image, the graphics code that generated the scene.

In Fig. 3.2, we provide an example of some factors of variation (i.e. generative factors) for the MNIST dataset of handwritten digits [LeCun et al., 1998]. In this problem, each digit can be fairly well defined by the digit type, angle, thickness and stroke width. These are good examples of conditionally independent factors, which we will discuss in the next section. You can also imagine that there exist very individual styles of writing digits which might not be fully explained by the factors shown in the figure, and these would be examples of dependent factors (again see next section). It is generally expected that disentangled representations will encode the conditionally independent factors into independent latent dimensions, but not dependent ones, which can be encoded in any way into dimensions which weren't occupied by independent factors.

3.2.1 Generative Factors

In the quotation, Bengio et al. [2013] talk about the explanatory factors of the data. In many modern works, these are referred to as the *generative factors*. The term generative factors derives from the assumption our dataset X is the result of a generative process depending on parameters θ^* and variable z , as described in Section 3.1.

In particular, Higgins et al. [2017] assume that the random variable z can be split into two parts, the conditionally independent v and conditionally dependent w . They then write the generative process (which they call the simulator process ‘Sim’) as,

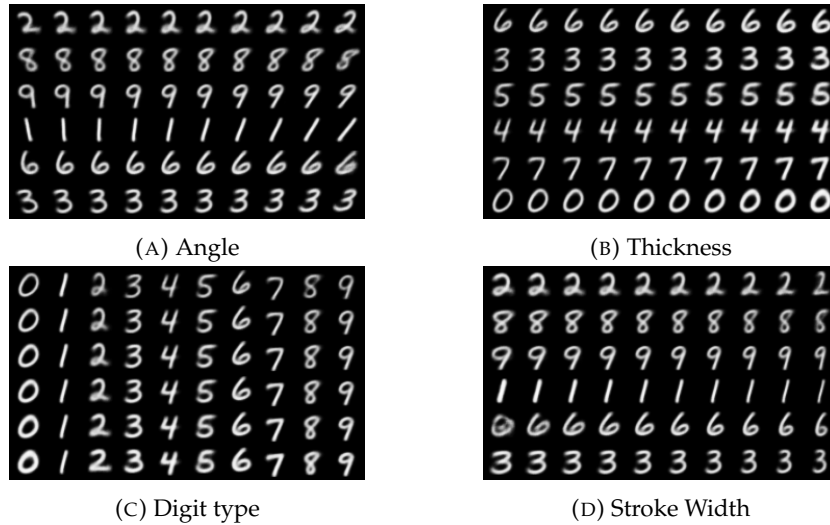


FIGURE 3.2: Example factors of variation as captured by Dupont [2018] on the MNIST dataset of handwritten digits. Each subfigure traverses a different latent unit and shows a different factor of variation. The ability to learn to separate these factors into different latent units is generally known as disentanglement. Figure by Dupont [2018].

$$p_{\theta^*}(x|v, w) = \text{Sim}(v, w) \quad , \quad (3.3)$$

Whilst the total set of generative factors is still given by v and w , for disentanglement, we will generally only consider factors v since the dependent factors w cannot be separated from each other. Whilst there is no generally accepted rigorous definition of disentanglement in terms of the VAE model (hence the work by Higgins et al. [2018]), if we assume that v and w exist, an acceptable (but loose) definition would be that the VAE latent space encodes v_i into independent dimensions.

In general, we say that for each image in the dataset, there exists a set of variables that completely defines the image, and the simulator process allows us to sample an image given these factors. Furthermore, there are a subset of these factors that are conditionally independent, and we will generally think of these as concepts which we can separate in the data. For example, the rotation of an object is independent of its location in the image, so the rotation and position may be examples of two of these conditionally independent factors. From this point onwards, whenever we refer to generative factors, we will mean the conditionally independent factors, not including the dependent factors.

If we consider 3D modelling software, then the simulation process is precisely the rendering function and the generative factors are the set of numbers that describe the objects in the scene, their positions, textures etc. Whilst for a complex real world scene, the set of such numbers might be very large or unknown, for synthetic data, we can be certain that they exist. Indeed, the datasets we will explore in this work were generated in such a fashion, and the generative factors values for each image are

available. It will often be useful to associate the factors v_i with the dataset concept to which they relate, e.g. v_1 may be describe rotation whilst v_2 might describe translation in the x direction. We will often talk about concepts such as rotation and translation as being the generative factors, when in reality, the factors are actually the underlying simulator variables v_i which relate to these concepts in the simulation process.

We can now see that the quote suggests that a disentangled representation space should have the property that concepts such as position, shape, etc. are separated into independent subspaces. In the particular case of a VAE model, for a disentangled latent space, we would expect each of these generative factors to be encoded into an independent latent dimension or latent subspace. As such, if we define a series of points in the latent space that only differ in the value of a single latent dimension, we would expect the output images to vary by just a single factor (for example, just the orientation of an object).

Following the description by Bengio et al. [2013], a number of works attempted to learn disentangled representations in semi-supervised [Cheung et al., 2014, Kingma et al., 2014, Kulkarni et al., 2015] and fully unsupervised [Whitney et al., 2016] settings.

3.2.2 Models for Disentanglement

We will now introduce a number of models for disentanglement. In all cases, the ultimate goal will be to learn a generative model with a latent code which is disentangled as described by the previous section. Specifically, the generative factors of the dataset should be separated in the latent code. We begin with InfoGAN, one of the first successful unsupervised methods, before outlining a set of models which will be used for evation in later chapters.

InfoGAN InfoGAN [Chen et al., 2016] aims to encourage a GAN [Goodfellow et al., 2014] to adhere to a latent code z alongside the usual noise variable η by maximising the mutual information $I(z; G(\eta, x))$ between z and the generator distribution G . By maximising this information and sampling the latent code dimensions independently, they hope the model will learn to associate different generative factors with independent latent dimensions.

To train the InfoGAN model, they first write the information,

$$I(z; G(\eta, z)) = H(z) - H(z|G(\eta, z)) = \mathbb{E}_{x \sim G(\eta, z)} \left[\mathbb{E}_{z' \sim p(z|x)} [\log p(z'|x)] \right] + H(z) \quad .$$

Since sampling from $p(z|x)$ would usually be intractable, they take advantage of the same trick we saw in the previous section, and compute a variational lower bound on

the mutual information. An inference distribution $q(\eta|x)$ is introduced alongside a prior $p(z)$ which is usually set to be uniform. First rewriting the expectation term,

$$\begin{aligned} \mathbb{E}_{x \sim G(\eta, z)} \left[\mathbb{E}_{z' \sim p(z|x)} [\log p(z'|x)] \right] &= \mathbb{E}_{x \sim G(\eta, z)} \left[\mathbb{E}_{z' \sim p(z|x)} \left[\log \left(\frac{p(z'|x)}{q(z'|x)} q(z'|x) \right) \right] \right] \\ &= \mathbb{E}_{x \sim G(\eta, z)} \left[\mathbb{E}_{z' \sim p(z|x)} \left[\log \frac{p(z'|x)}{q(z'|x)} + \log q(z'|x) \right] \right] \\ &= \mathbb{E}_{x \sim G(\eta, z)} \left[D_{KL} (p(z'|x) || q(z'|x)) + \mathbb{E}_{z' \sim p(z|x)} \log q(z'|x) \right] \\ &\geq \mathbb{E}_{x \sim G(\eta, z)} \mathbb{E}_{z' \sim p(z|x)} \log q(z'|x) \quad . \end{aligned}$$

Chen et al. [2016] now state and prove the following lemma that allows them to avoid sampling from $p(z|x)$. We present a slightly expanded proof³ for clarity.

Lemma 3.1. *For random variables X, Y and a function $f(x, y)$ with suitable regularity conditions then*

$$\mathbb{E}_{x \sim X, y \sim Y|x} [f(x, y)] = \mathbb{E}_{x \sim X, y \sim Y|x, x' \sim X|y} [f(x', y)]$$

Proof.

$$\begin{aligned} \mathbb{E}_{x \sim X, y \sim Y|x} [f(x, y)] &= \int_x P(x) \int_y P(y|x) f(x, y) dy dx \\ &= \int_x \int_y P(x, y) f(x, y) dy dx \\ &= \int_x \int_y P(x, y) f(x, y) \int_{x'} P(x'|y) dx' dy dx \\ &= \int_x \int_y P(x|y) P(y) f(x, y) \int_{x'} P(x'|y) dx' dy dx \\ &= \int_x \int_y P(x|y) f(x, y) \int_{x'} P(x', y) dx' dy dx \\ &= \int_{x'} \int_y P(x'|y) f(x', y) \int_x P(x, y) dx dy dx' \\ &= \int_x P(x) \int_y P(y|x) \int_{x'} P(x'|y) f(x', y) dx' dy dx \\ &= \mathbb{E}_{x \sim X, y \sim Y|x, x' \sim X|y} [f(x', y)] \end{aligned}$$

□

³By: <https://stats.stackexchange.com/questions/250838/integral-identity-of-lemma-contained-in-infogan-paper>

Given this lemma and the previous inequality, there is an optimisable lower bound of the information given by,

$$\begin{aligned} L(G, Q) &= \mathbb{E}_{z \sim p(z)} \mathbb{E}_{x \sim G(\eta, z)} \left[\log q(z|x) \right] + H(z) \\ &= \mathbb{E}_{x \sim G(\eta, z)} \left[\mathbb{E}_{z' \sim p(z|x)} [\log q(z'|x)] \right] + H(z) \quad (\text{by lemma}) \\ &\leq I(z; G(\eta, z)) \quad . \end{aligned}$$

We can now maximise $L(G, Q)$ rather than maximising the information $I(z; G(\eta, z))$ directly. Architecturally speaking, this process is now simple. For fake data, we can sample a latent z from some defined prior $p(z)$ (via the reparametrisation trick), pass this through the generator with sampled noise η and using a small linear layer on top of the discriminator, output the parameters for q . The minimax game for the InfoGAN becomes,

$$\min_{G, q} \max_D V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L(G, Q)$$

where D is the GAN discriminator and $V(D, G)$ is the standard GAN minimax objective.

This simple architecture is enough for disentanglement to emerge in the latent codes. In the paper, they experimented with manually chosen distributions for each code, *i.e.* for MNIST they chose one categorical (digit) and two continuous (style variations). Interestingly however, some of the more interesting results occurred when less care was put into choice of distribution, for example the classic CelebA dataset where they chose just 10 categorical codes (of size 10). Here they successfully extracted factors such as azimuth, elevation, lighting and face width.

β -VAE The β -VAE [Higgins et al. \[2017\]](#) was one of the first models to show consistent disentanglement using the VAE framework. β -VAE provided three additional contributions toward disentanglement that contributed to its success: first they cast a new light on the standard VAE framework based on an information bottleneck in the latent code, second they introduce a repeatable metric for approximating disentanglement (see Section 3.2.4) and third they introduce the dSprites dataset (see Section 3.2.3) a ubiquitous synthetic dataset for disentanglement research.

Similar to InfoGAN, the β -VAE aims to separate generative factors in a latent code. It differs from InfoGAN in that it uses the VAE framework and doesn't attempt to maximise the information between the latent code and the generated image. β -VAE instead aims to encourage independence through imposing a capacity constraint (modulated by parameter β) on the latent code. For large β , the code will be

encouraged to be independent if there are some underlying generative factors which are independent, the assumption being that encoding the generative factor values is the most efficient means of conveying information about the data.

Assume we have the simulating process set out in Section 3.2.1, with simulator ‘Sim’, dataset X , conditionally independent factors v and conditionally dependent factors w . The β -VAE framework aims to train an unsupervised generative model to learn the joint of the data points x and latents z such that $p(x|z)$ approximates $p(x|v, w)$. Recall from Section 3.1, a standard objective is to maximise the likelihood of the data x marginalised over the latents z ,

$$\max_{\theta} \mathbb{E}_{z \sim p_{\theta}(z)} p_{\theta}(x|z) \quad .$$

Similarly to the standard VAE, we prefer to approximate $p_{\theta}(z|x)$ with an inference distribution $q_{\phi}(z|x)$, since accurately estimating p would be hard. A secondary consideration for the β -VAE framework is that we also want $q_{\phi}(z|x)$ to capture the generative factors v in a conditionally independent way - which is consistent with our notion of disentanglement. They allow w to be captured by q in any manner, as long as it is in separate dimensions to those which capture v .

The important conceptual idea that this framework introduces compared to the standard VAE framework is that, rather than aiming to maximise $p_{\theta}(x)$, they maximise the marginal of $p_{\theta}(x|z)$ over z under an information bottleneck constraint on z . This constraint is chosen to be matching q_{ϕ} to a prior $p(z)$ which has the capacity constraint alongside the dimension-wise statistical independence which is desired. The constrained problem becomes,

$$\max_{\phi, \theta} \mathbb{E}_{x \sim X} \left[\mathbb{E}_{q_{\phi}(z|x)} \log p_{\theta}(x|z) \right] \text{ subject to } D_{KL}(q_{\phi}(z|x) || p(z)) < \epsilon \quad ,$$

which under KKT conditions gives the Lagrangian (with multiplier β),

$$\mathcal{L} = \mathbb{E}_{q_{\phi}(z|x)} \log p_{\theta}(x|z) - \beta \left(D_{KL}(q_{\phi}(z|x) || p(z)) - \epsilon \right) \quad . \quad (3.4)$$

The prior is then set to $\mathcal{N}(0, \text{diag}(1))$, and we arrive back at the original VAE objective, with the divergence term reduced by some constant ϵ and scaled by the multiplier β . In implementation, the multiplier is generally chosen manually and used to control the relative importance of the two objectives. The original paper examined this purely based on the fact that higher β would lead to a higher penalty for not matching the prior. Larger values should result in more disentangled results, since the prior is dimension-wise independent.

The objective was further analysed by the follow-up work by [Burgess et al. \[2018\]](#). By looking at $q_{\phi}(z|x)$ as a set of information channels that noisily transmit information

about input x , they can view the KL-Divergence as an upper bound on the amount of information that can be transmitted through the channel per data sample.

$$I(x, z) = D_{KL}(p(x, z) || p(x)p(z))$$

Since the posterior and prior are factorised Gaussians, we can interpret the latent dimensions as Gaussian white noise channels, with information capacity ϵ .

In the case of $\epsilon = 0$, this constraint encourages maximising the overlap of $q(z|x_1)$ and $q(z|x_2)$ for any two images x_1 and x_2 . Burgess et al. [2018] explain that this is due to the KL being minimised when $q(z|x_i) = \mathcal{N}(0, 1)$, and increased by non-zero posterior means or low posterior variances. Thus the KL pressure encourages similar means, which increases the overlap. Furthermore, good discriminability (and thus data log likelihood), $q(z|x_i)$ requires low variance, so overlapping these posteriors will lead to a broadening of the aggregate posterior allowing a lower KL divergence. The overlap is important since it encourages nearby (in pixel space) data points to be nearby in the latent space. When a sample z from $q(z|x_1)$ is more likely under $q(z|x_2)$, then the log likelihood cost (mean squared error between x_1 and reconstruction from sampled z) is high if x_1 and x_2 are visually distinct and low if they are visually similar. The result is that the distance between points in data space is preserved through the mapping to latent space, a concept termed as data locality.

Given this framework, the β -VAE with an appropriately chosen β outperformed Info-GAN in their experiments. Furthermore, the Info-GAN framework was sensitive to how the latent code was defined (i.e. the number of categorical vs continuous variables), whereas the β -VAE did not require such choices.

Controlled Capacity Increase VAE Burgess et al. [2018] provide the conceptual leap that explains how the information framework can lead to disentangling in the latent space. They find that starting with low (zero) capacity latent channels and gradually increasing this capacity over time leads to better disentanglement.

Assuming we start with a very strong zero capacity bottleneck ($\beta \gg 1, \epsilon = 0$), we can expect the image reconstruction loss to be high, and the KL-divergence loss to become small (since $\beta \gg 1$, and the latent space can transmit very little information). In the information framework, reducing the image reconstruction loss requires transmitting a non-zero amount of information through the latent channels. Burgess et al. [2018] propose gradually increasing the information capacity of the latent channels (increase ϵ) over training. As the capacity is increased, more information can be passed through the latent channels, and the image reconstruction can be improved.

Burgess et al. [2018] argue that the first concepts to be encoded will be those that most significantly increase the data log-likelihood. Borrowing the same example as Burgess

et al. [2018], for dSprites, this would likely be the position of the object, since log likelihood will vanish if the reconstruction has position off by even a few pixels. They then contend that as the capacity is further increased, new factors (for example, scale) would be encoded into independent latent dimensions and that this happens due to the same loss pressure that leads to data locality.

This analysis lead Burgess et al. [2018] to propose the Controlled Capacity Increase VAE (cc-VAE). This model begins training with low capacity latent channels ($\beta \gg 1, \epsilon = 0$) and as training progresses, they gradually allow the capacity to increase (increase ϵ). In the early stages of training (low capacity), the model will learn factors that give the best reconstruction gains (e.g. position). As capacity is increased, other factors such as scale or rotation are encoded. As discussed, they hope that gradually introducing factors to the encoding, rather than all at once, will result in better disentangling performance due to the log-likelihood penalty of encoding two factors into the same latent.

InfoVAE If one observes poor sample quality from a VAE it is understandable that you would attribute this to poor approximation of the true parameters of the generative process, or poor approximation of the data distribution. This could arise from a lack of complexity in the decoder θ . However, when using extremely high capacity decoders, it was found [Bowman et al., 2015] that the model often collapses to be independent of z ($p_\theta(x|z) = p(x)$). In this case, the latent distribution collapses to the prior, $q_\phi(z|x) = p(z)$, the capacity is 0 and the information $I(x; z)$ is also zero.

The goal of InfoVAE was to allow the maximisation of the information term $I(x; z)$ in the VAE framework, so that this collapse could not occur, even with high capacity decoders.

Since generally it is agreed that the aim of VAEs is to learning meaningful representations of data and not just produce high quality samples, having 0 information about the input is undesirable. As such Zhao et al. [2017] modified the ELBO to additionally maximise the information $I_q(x; z)$. However, rather than optimising the standard ELBO form, they consider an equivalent form, reweighted with α and λ ,

$$\mathcal{L}_{\text{InfoVAE}} = -\lambda D_{KL}(q_\phi(z)||p(z)) - E_{q_\phi(z)}[D_{KL}(q_\phi(x|z)||p_\theta(x|z))] + \alpha I_q(x; z) \quad .$$

This can then be rewritten to be a weighted form of the standard ELBO,

$$\begin{aligned} \mathcal{L}_{\text{InfoVAE}} &= \mathbb{E}_{p(x)} \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) \\ &\quad - (1 - \alpha) \mathbb{E}_{p(x)} D(q_\phi(z|x)||p(z)) \\ &\quad - (\alpha + \lambda - 1) D(q_\phi(z)||p(z)) \quad . \end{aligned}$$

Notice that this now explicitly optimises the aggregate posterior to be like the prior. Furthermore, notice that it regresses to the β -VAE objective when $D = D_{KL}$ and $\alpha = 1 - \lambda$, with $\beta = (1 - \alpha)$. When $\beta > 1$, $\alpha < 0$ and the objective penalises the information $I_q(x; z)$.

Zhao et al. [2017] consider a number of different divergences $D(q_\phi(z)||p(z))$ which are easier to compute than the KL divergence. Any implementation in our work will use the MMD variant, which we denote MMD-VAE. The MMD variant was found to result in better log-likelihood in their tests compared to the standard VAE, being close to those reported by the extremely high capacity PixelRNN, whilst retaining dependency on z .

Whilst not directly targetted at improving disentanglement performance, we will use the InfoVAE as one of our baselines for experiments later in this work. It is an interesting comparison since it allows for better image reconstructions whilst maintaining the disentangling properties of the VAE framework.

BetaTcVAE One of the greatest strides in terms of understanding and disentangling performance came from Chen et al. [2018] with the BetaTcVAE model. They propose to improve disentanglement performance by structuring the objective to minimise the total correlation between latent dimensions. To achieve this, they provide a simple decomposition (assuming a factorised $p(z)$) of the relative entropy to yield 3 terms with distinct properties.

$$\begin{aligned}
\mathbb{E}_{p(x)} D_{KL}(q(z|x)||p(z)) &= \\
&= \mathbb{E}_{p(x)} \mathbb{E}_{q(z|x)} \log \frac{q(z|x)}{p(z)} \quad \text{Expand KL} \\
&= \mathbb{E}_{p(x)} \mathbb{E}_{q(z|x)} \left[\log q(z|x) - \log p(z) + \log q(z) - \log q(z) + \log \prod_j q(z_j) - \log \prod_j q(z_j) \right] \\
&= \mathbb{E}_{q(z,x)} \log \frac{q(z|x)}{q(z)} + \mathbb{E}_{q(z)} \log \frac{q(z)}{\prod_j q(z_j)} + \mathbb{E}_{q(z)} \sum_j \log \frac{q(z_j)}{p(z_j)} \quad \textcircled{A} \\
&= \mathbb{E}_{q(z,x)} \log \frac{q(z|x)p(x)}{q(z)p(x)} + \mathbb{E}_{q(z)} \log \frac{q(z)}{\prod_j q(z_j)} + \sum_j \mathbb{E}_{q(z)} \log \frac{q(z_j)}{p(z_j)} \quad \textcircled{B} \\
&= \mathbb{E}_{q(z,x)} \log \frac{q(z|x)p(x)}{q(z)p(x)} + \mathbb{E}_{q(z)} \log \frac{q(z)}{\prod_j q(z_j)} + \sum_j \mathbb{E}_{q(z_j)q(z_{\setminus j}|z_j)} \log \frac{q(z_j)}{p(z_j)} \quad \textcircled{C} \\
&= \mathbb{E}_{q(z,x)} \log \frac{q(z|x)p(x)}{q(z)p(x)} + \mathbb{E}_{q(z)} \log \frac{q(z)}{\prod_j q(z_j)} + \sum_j \mathbb{E}_{q(z_j)} \log \frac{q(z_j)}{p(z_j)} \\
&= \underbrace{D_{KL}(q(z, x)||q(z)p(x))}_{\text{I. Index-Code MI}} + \underbrace{D_{KL}(q(z)||\prod_j q(z_j))}_{\text{II. Total Correlation}} + \underbrace{\sum_j D_{KL}(q(z_j)||p(z_j))}_{\text{III. Dimension-wise KL}} \quad (3.5)
\end{aligned}$$

Chen et al. [2018] propose the total correlation term (II.) is the vital component for disentanglement and the reason that β -VAE produces disentanglement. Penalising this term encourages the model to learn statistically independent factors, i.e. disentangling them. However, since evaluating $q(z)$ would normally require computing $\mathbb{E}_x q(z|x)$ (i.e. over the whole dataset), Chen et al. [2018] prefer a Monte-Carlo estimation which can be evaluated with mini-batch sampling, which they call Minibatch Weighted Sampling (MWS).

Let $\mathcal{B}_M = \{x_1, \dots, x_M\}$ be a minibatch sampled from $p(x)$, then $p(\mathcal{B}_M) = (1/N)^M$. Let $r(\mathcal{B}_M|x)$ be the probability of a minibatch given one of the samples is fixed to be x , then $r(\mathcal{B}_M|x) = (1/M)^{N-1}$. Note that $r(\mathcal{B}_M|x) = 0$ for minibatches which do not contain x and so $r < p$ in general. The expectation can then be estimated:

$$\begin{aligned} \mathbb{E}_{q(z)} \log q(z) &= \mathbb{E}_{q(z,x)} \log \mathbb{E}_{x' \sim p(x)} q(z|x') \quad \text{Expanding the expectation} \\ &= \mathbb{E}_{q(z,x)} \log \mathbb{E}_{p(\mathcal{B}_M)} \frac{1}{M} \sum_{m=1}^M q(z|x_m) \\ &\geq \mathbb{E}_{q(z,x)} \log \mathbb{E}_{r(\mathcal{B}_M|x)} \frac{p(\mathcal{B}_M)}{r(\mathcal{B}_M|x)} \frac{1}{M} \sum_{m=1}^M q(z|x_m) \quad \text{Since } r < p \\ &= \mathbb{E}_{q(z,x)} \log \mathbb{E}_{r(\mathcal{B}_M|x)} \frac{1}{MN} \sum_{m=1}^M q(z|x_m) \quad \text{Expanding } p, r \end{aligned}$$

This allows for approximating each of the terms in equation 3.5 through minibatch sampling, since we can approximate $\mathbb{E}_{q(z)} \log q(z)$ by,

$$\mathbb{E}_{q(z)} \log q(z) \approx \frac{1}{M} \sum_{i=1}^M \left[\log \sum_{j=1}^M q(z(x_i)|x_j) - \log(MN) \right]$$

where $z(x_i)$ is a sample from $q(z|x_i)$.

Ⓐ Grouping and $\mathbb{E}_{q(z|x)} \mathbb{E}_{p(x)} = \mathbb{E}_{q(z,x)}$ Ⓑ Linearity of expectation Ⓒ Break out dependent var

The alternate (and simpler) method given by Chen et al. [2018] was a process they termed stratified minibatch sampling, which computes $q(z)$ through the following:

$$\begin{aligned}
q(z) &= \mathbb{E}_{p(x)} q(z|x) \\
&= \mathbb{E}_{p(\mathcal{B}_M)} \frac{1}{M} \sum_{m=1}^M q(z|x_m) \\
&= \mathbb{P}(x^* \in \mathcal{B}_M) \mathbb{E} \left[\frac{1}{M} \sum_{m=1}^M q(z|x_m) \mid x^* \in \mathcal{B}_M \right] \\
&\quad + \mathbb{P}(x^* \notin \mathcal{B}_M) \mathbb{E} \left[\frac{1}{M} \sum_{m=1}^M q(z|x_m) \mid x^* \notin \mathcal{B}_M \right] \\
&= \frac{M}{N} \mathbb{E} \left[\frac{1}{M} \sum_{m=1}^M q(z|x_m) \mid x^* \in \mathcal{B}_M \right] + \frac{N-M}{N} \mathbb{E} \left[\frac{1}{M} \sum_{m=1}^M q(z|x_m) \mid x^* \notin \mathcal{B}_M \right]
\end{aligned}$$

By isolating x^* from the first term and x_M from the second, there is an estimator,

$$f(z, x^*, B_{M+1} \setminus \{x^*\}) = \frac{1}{N} q(z|x^*) + \frac{1}{M} \sum_{m=1}^{M-1} q(z|x_m) + \frac{N-M}{NM} q(z|x_M)$$

with the following estimates which become exact when $M = N$,

$$\begin{aligned}
q(z) &\approx \mathbb{E} f(z, x^*, B_{M+1} \setminus \{x^*\}) \\
\mathbb{E}_{q(z,n)} \log q(z) &\approx \frac{1}{M+1} \sum_{i=1}^{M+1} \log f(z_i, x_i, B_{M+1} \setminus \{x_i\})
\end{aligned}$$

Using either MWS or MSS, the three terms I, II and III can be computed since they allow evaluation of $q(z)$ in some efficient form. The BetaTcVAE loss function then weights each term individually,

$$\mathcal{L} = \mathbb{E}_{p(x)q(z|x)} \log p(x|z) - \alpha I(z; x) + \beta D_{KL}(q(z) \parallel \prod_j q(z_j)) + \gamma \sum_j D_{KL}(q(z_j) \parallel p(z_j))$$

rather than weight each by a single β as is the case in β -VAE.

The resulting model shows consistently strong disentanglement (both visually and empirically) and is generally a good choice for comparing disentanglement performance.

3.2.3 Datasets for Disentanglement

There have been many datasets created for the purpose of studying disentanglement and disentangled representations. Since it is useful to have access to the generative

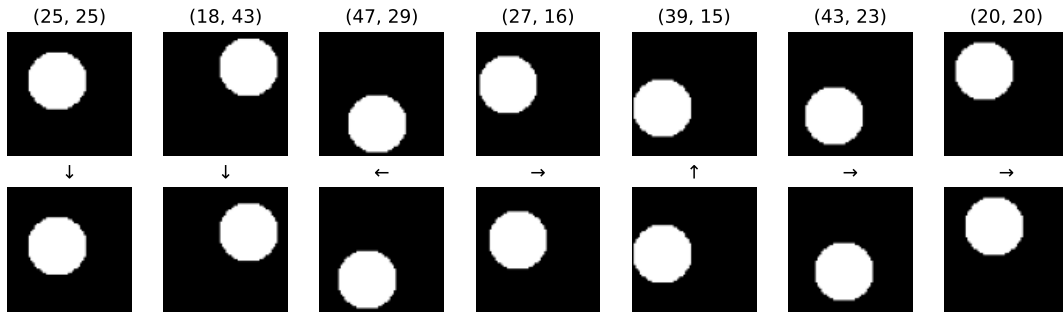


FIGURE 3.3: A selection of Flatland images with the associated generative factors (x, y) (top row) and an example of an action applied to the top row with results shown in the bottom row.

factors, these datasets are generally synthetic. In this work we will mostly use the FlatLand [Caselles-Dupré et al., 2018] and dSprites [Matthey et al., 2017] datasets, however we will also use the Norb dataset [LeCun et al., 2004]. We will now briefly describe these datasets and give their generative factors. As stated earlier, these factors will be defined by the concepts which they relate to (e.g. rotation etc) instead of just some dimensions of the dataset simulator variables v .

FlatLand The Flatland dataset was introduced by Caselles-Dupré et al. [2018] as a simple environment for reinforcement learning. It consists of a circular agent navigating a 2D world which may contain obstacles. For our work, we will use the simplified version of flatland (as used by Caselles-Dupré et al. [2019]) which does not contain obstacles.

Specifically, the Flatland dataset has a black canvas of size 64×64 pixels, with a white circular agent with radius 15 pixels. The agent cannot move into the boundary of the canvas (cannot be occluded) however it can take any other x - y position. For any given image from the dataset, the state is completely defined by the x - y position of the agent. As such, the generative factors are exactly the set of possible x - y positions.

$$\text{Generative Factors} = \{(x, y) | x, y \in [15, 49]\}$$

In the reinforcement learning framework, there must be actions that act on the environment. For Flatland, these actions are moving the agent in one of four directions (up, down, left, right). Whilst it can be selected freely, the step size (number of pixels moved by the agent) is set to 5 for Flatland in this work and Caselles-Dupré et al. [2019]. In Fig. 3.3 we show example Flatland images and actions.

dSprites The dSprites dataset [Matthey et al., 2017] is the most common dataset used for disentanglement research which use VAEs. It was introduced by [Higgins

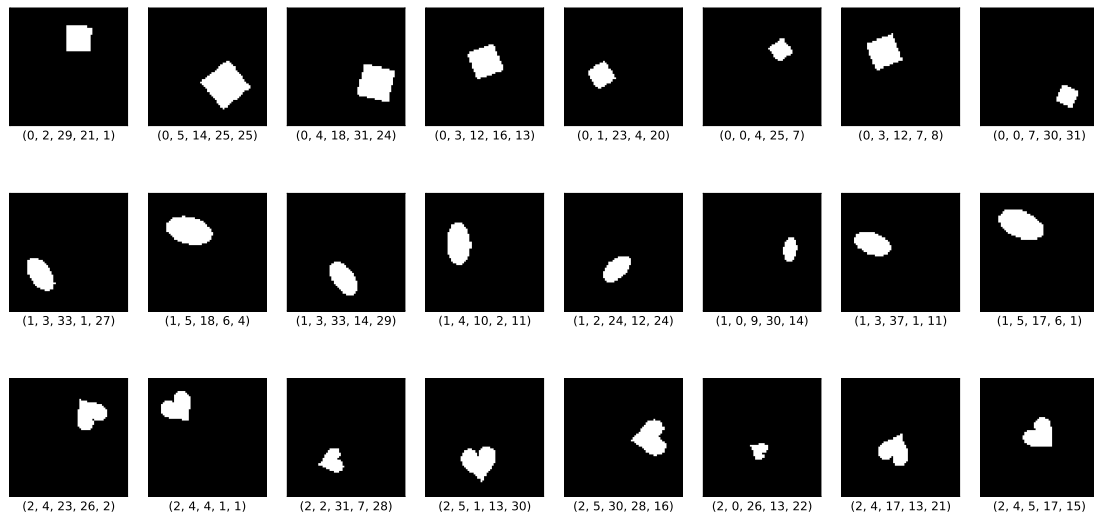


FIGURE 3.4: A selection of dSprites images with their associated generative factors (shape, scale, orientation, x , y).

et al., 2017] (under the name 2D shapes) as a means to allow evaluation of their disentanglement metric (BetaVAE Metric, see Section 3.2.4), which requires access to the generative factors.

$$\text{Generative Factors} = \{(\text{shape}, \text{scale}, \text{rotation}, x, y)\}$$

The dSprites dataset consists of a black canvas (64×64 pixels) with one of 3 white shapes present, a heart, square, or oval. The shapes are placed on a grid with 32 x positions and 32 y positions. Furthermore, they are rotated by one of 40 rotational angles (between 0 and 2π), and are scaled by one of 6 scale values. Each image can be defined exactly by these values (shape, scale, rotation, x , y), and as such, these form the generative factors for the dataset. In Fig. 3.4 we show example dSprites images.

Norb The norb dataset was introduced by LeCun et al. [2004] for the purpose of recognising 3D objects from their shape. It consists of real world images of 50 different toys under varying lighting conditions (6 variants), from 9 different elevations and from 18 different azimuth angles. Since again, each image can be described by the toy, lighting, elevation and azimuth values, these form the generative factors of the dataset.

$$\text{Generative Factors} = \{(\text{azimuth}, \text{toy}, \text{elevation}, \text{lighting})\}$$

In Fig. 3.5 we show example Norb images.

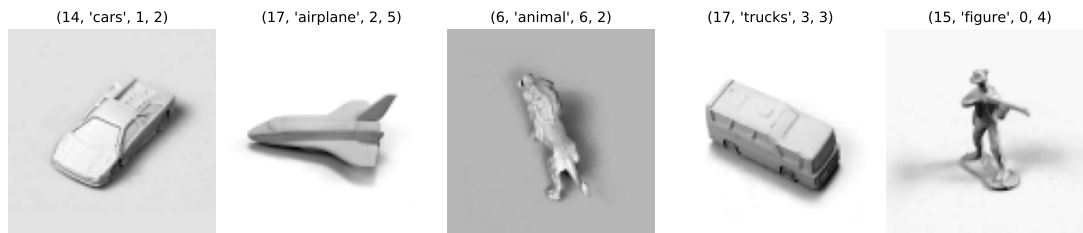


FIGURE 3.5: A selection of Norb images with their associated generative factors (azimuth, toy, elevation, lighting)

3.2.4 Measuring Disentanglement

Given the wealth of models aiming to produce disentangled representations, there is a requirement for methods to compare results. Many of the works proposing the previously discussed models also proposed new metrics to measure disentanglement based on differing interpretations of what disentanglement is or in what context it should be seen. We will now briefly list and discuss the main metrics and comparison methods that have been used. Most of this work will be concerned with learning linear disentangled representations, which are a form of disentanglement to which some of the metrics we discuss may or may not be able to measure appropriately. As such, it will also be useful to discuss here each metrics relevance to linear disentanglement, which will be introduced properly in Section 3.3.

Throughout these metrics, we will often require labels of the generative factors. We will consider the conditionally independent factors v from Section 3.2.2 (β -VAE), with individual factor k (e.g. shape or rotation angle) denoted by v_k . For the VAE latent space, we will denote the j 'th dimension by z_j .

Latent Space Visualisations The simplest way to gauge disentanglement is through visualisations of latent walks. By traversing a single latent dimension and keeping all others fixed, we can observe changes in the output that are solely due to that latent. If we perform this separately for each dimension, we can get an overview of how data is encoded into the space. If we see the generative factors encoded independently into different latents then we can say the space is disentangled. If we see dimensions with variation in multiple generative factors then we can say it is (likely) entangled. For an example, consider Fig. 3.6a which shows disentangled latent traversals. For the two factors (x and y position), we see each encoded into a single dimension with no overlap between them. For comparison, Fig. 3.6b shows entangled traversals, with dimensions (rows) encoding parts of both factors and multiple rows encoding the same factor.

Latent traversals have the advantage of being simple to perform and allow good intuition on whether a space is disentangled. They do, however, have a major

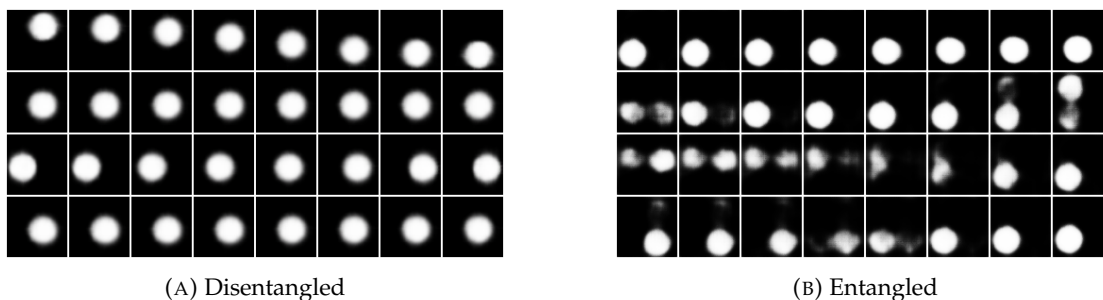


FIGURE 3.6: Example Disentangled and Entangled latent traversals for a simple two factor problem (x and y position) using 4 latent dimensions (rows).

drawback which is that they do not allow quantification of *how* disentangled the space is. So whilst latent walks can be used to demonstrate that a model can learn disentanglement, they cannot be used (effectively) to compare models. Furthermore, they are representative of only a single instance of the model being visualised, not a general property of the model.

Latent visualisations will continue to be a useful visualisation for linear disentangled representations. We will often traverse the orbit of actions on the latent space, rather than individual latent dimensions as has been described here.

BetaVAE Metric The BetaVAE metric [Higgins et al., 2017] reports the classification accuracy of a (low capacity) linear classifier on the task of predicting the latent dimension index in which a specific generative factor is encoded. The metric relies on a simulating process (‘Sim’) for the data, as described in Section 3.1, and also requires that we have labels for a subset of the independent factors $v \in \mathbb{R}^K$, although the conditionally dependent factors (w) need have no labels. We then

Each element of the dataset used to compute this metric is generated in the following way:

1. Choose dimension index $i \sim \text{Uniform}(0, K)$
2. For a batch of L samples:
 - Sample two sets (a and b) of factors $v_{a,l}, v_{b,l}$ where the i 'th dimension of each is the same and l refers to the number in the batch.
 - Sample two sets of factors $w_{a,l}, w_{b,l}$ randomly
 - Sample images $x_{a,l} \sim \text{Sim}(v_{a,l}, w_{a,l}), x_{b,l} \sim \text{Sim}(v_{b,l}, w_{b,l})$ from the simulating process
 - Infer the latent codes $z_{a,l} = \mu(x_a), z_{b,l} = \mu(x_{b,l})$ using the VAE encoder: $q(z|x) \sim N(\mu(x), \sigma(x))$
 - Compute $z_l = |z_{a,l} - z_{b,l}|$

3. Return tuple $(z = \frac{1}{L} \sum z_l, i)$ where the classifier has to predict the label i

The metric is then given as the classification accuracy on this dataset, i.e. how accurately the low capacity classifier can predict the label i using z . This has the obvious downsides that it will depend on the classifier chosen, the hyperparameters used to train the classifier, etc. In this work we use the scikit-learn multinomial logistic regression classifier with newton-cg solver and no penalty (other settings default). A link to the specific implementation is given at the end of this section.

The intuition for this metric is as follows, if the VAE latent space is disentangled and we fix index i , then z should have some dimension (k) which is close to 0, since the factor v_i is encoded in the latent dimension with index k . We can see this by considering the k 'th dimension of $z_{a,l}$, denoted $[z_{a,l}]_k$. Since factor index i is fixed, then we expect $[z_{a,l}]_k$ to be very close to $[z_{b,l}]_k$ and thus $[z_{a,l} - z_{b,l}]_k$ should be close to 0. Since the other dimensions of $v_{a,l}, v_{b,l}$ are chosen randomly, $[z_l]_{j \neq k} = [z_{a,l} - z_{b,l}]_{j \neq k}$ will not always be close to 0. As such, in a disentangled space, $[z_l]_k$ should be close to 0 and $[z_l]_j$ should be greater than 0. Even a low capacity classifier should be able to map from this to a unique index for k .

However, if the VAE latent space is entangled, even when fixing index i , the values $[z_{a,l}]_k$ and $[z_{b,l}]_k$ need not be similar. As such, there is no way to determine which index has been fixed by looking at z_l .

Mutual Information Gap (MIG) The mutual information gap [Chen et al., 2018] is motivated by the idea that (classical) disentangled representations encode individual generative factors into individual latent dimensions. Thus for disentangled representations, the difference in information about a single factor encoded into the first and second most informative dimensions will be large. Ideally it will all be encoded into a single dimension. However for entangled representations, the information about any given factor will be encoded across many or all dimensions, resulting in this difference being small.

Thus for each generative factor v_k , Chen et al. [2018] compute its mutual information $I(z_j, v_k)$ with each latent dimension z_j . The information gap is given as the average normalised (by the entropy $H(v_k) = -\mathbb{E}_{p(v_k)} \log p(v_k)$) difference between the first and second most informative latent dimensions. The MIG score, is then given as,

$$\frac{1}{K} \sum_{k=1}^K \frac{1}{H(v_k)} \left(I(z_{j_k}, v_k) - \max_{j \neq j_k} I(z_j, v_k) \right) \quad \text{where } j_k = \arg \max_j I(z_j, v_k) \quad . \quad (3.6)$$

The MIG metric can capture linear disentanglement where all acting symmetry groups have only 1D irreducible representations, however groups with higher dimensional

representations (e.g. cyclic groups) result in a low information gap and poor MIG scores. This is due to the information about the symmetry (which generally can be directly related to a single generative factor) being encoded across as many latent dimensions as the dimensionality of its irreducible representations. We will extend this to more adequately express linear disentangled representations with the Factor Leakage metrics.

DCI Informativeness, Disentanglement and Completeness The three DCI metrics proposed by Eastwood and Williams [2018] work from a common set of regressors, which are required to produce an importance matrix R_{ij} over the latent dimensions z_i for predicting the factor v_j . The regressors take the form of the Scikit-learn Gradient boosting classifier with default settings trained to predict the factor v_j from the whole code z . The importance matrix is then computed using the feature importances reported by Scikit-learn. Using the importance matrix, the three metrics can be defined.

Disentanglement measures the degree to which latent dimensions capture information about at most a single generative factor. The disentanglement score D_i for latent dimension z_i is given by,

$$D_i = \sum_j \rho_i (1 - H(P_i)) \quad \text{where } \rho_i = \frac{\sum_j R_{ij}}{\sum_{ij} R_{ij}}, \quad P_{ij} = \frac{R_{ij}}{\sum_k R_{ik}} . \quad (3.7)$$

The entropy is computed by $H(P_i) = -\sum_{k=0}^{K-1} P_{ik} \log_k P_{ik}$, and is high when latent dimension z_i is important to predicting many factors v_j , thus resulting in a low D_i . Again, ideal disentangled representations have one latent dimension important for one factor, in which case the entropy would be low and D_i high.

DCI disentanglement is suitable for all linear disentangled representations since they all enforce single latent dimensions to be relevant to single symmetry groups. The exception to this would be cases where symmetries do not correspond exactly with what is considered a generative factor. For example, orientation in 3D might feasibly (albeit perhaps incorrectly) be split into a 3 generative factors, one for rotations about each Cartesian axis. In this case, the entropy of any latent dimension concerned with orientation/rotation would be non-zero since it has non-zero probability over 3 factors. This can be overcome by choices of generative factors that align with the symmetry groups acting on the data or visa versa.

Completeness measures the degree to which generative factors are captured by a single latent dimension. The completeness score C_j is given by,

$$C_j = 1 - H(P_j) \quad (3.8)$$

The entropy here is given by $H(P_{\cdot j}) = -\sum_{d=0}^{D-1} P_{dj} \log_D P_{dj}$. In this case, if a single latent dimension contributes to the prediction of factor v_j then the score will be 1, however if many do then it will be close to 0.

For linear disentangled representations, it is often the case that irreducible representations are of dimension greater than 1. Thus linear disentangled representations will often be considered over-complete if the generative factors do not align with the symmetry groups acting on the data.

Informativeness is simply the amount of information the representation captures about the underlying generative factors. To measure this, [Eastwood and Williams \[2018\]](#) propose training a classifier to directly predict the generative factor values for a given latent representation. So, assuming $x \sim \text{Sim}(v, w)$ and $z = \mu(x)$ is encoded using the VAE encoder $q(z|x) \sim N(\mu(x), \sigma(x))$, then the classifier is tasked with predicting v from z . The informativeness is then reported as the classification accuracy subtracted from 1.

The informativeness continues to provide insight to linear disentangled representations, although it only measures information content and not disentanglement performance, so we will not make use of this metric.

Modularity and Explicitness [Ridgeway and Mozer \[2018\]](#) define two scores, the modularity which measures the extent to which latent dimensions encode information about a single factor, and explicitness which measures the extent to which factors are represented/retrievable in/from the code. To define the modularity, they first define a template vector,

$$t_{iv} = \begin{cases} \theta_i & \text{if } v = \arg \max_k (I(z_i, v_k)) \\ 0 & \text{else,} \end{cases} \quad (3.9)$$

where $\theta_i = \max_k (I(z_i, v_k))$ and $I(z_i, v_k)$ is the mutual information between latent dimension i and generative factor v_k . Thus θ_i represents the information between latent dimension z_i and the generative factor v_k with which it has the most mutual information. The modularity is then defined,

$$\text{Mod}_i = 1 - \frac{\sum_k (I(z_i, v_k) - t_{ik})^2}{\theta_i^2 (N - 1)}, \quad (3.10)$$

where the quotient represents the deviation from the ideal case.

To measure the explicitness, [Ridgeway and Mozer \[2018\]](#) fit a one-versus-rest logistic-regression classifiers which classify the factors v_k from the latent code z . The explicitness metric is then reported as the mean of the ROC-AUC scores of the classifiers for each factor v_k . For this ROC analysis, we use Scikit-learn multiclass

implementation which allows us to compute ROC-AUC scores from multiclass classifiers which predict each dimension of factor v_k (See end of section).

Similar to previous metrics, the modularity is suitable for linear disentangled representations since it measures the extent latent dimensions talk about single factors (assuming factors equate to symmetries). The explicitness is also suitable since the whole code is used as input to the classifiers.

SAP The SAP score [Kumar et al., 2017] aimed to better correlate metric score with qualitative measurements (inspection of latent traversals). They begin by obtaining the score matrix R_{ik} which represents the ability to predict factor v_k using only latent dimension i . For categorical factors, this is computed as a classification accuracy from a linear SVM. For continuous factors, it is computed as the R^2 score of a linear regression.

The SAP score is then reported as the mean difference between the top two scores for each generative factor. A high SAP score is obtained when each factor is encoded into a single latent dimension, as is desired by classical disentanglement. A low SAP score is obtained when the opposite is true.

$$SAP = \frac{1}{N} \sum_k \left(\max_i R_{ik} - \max_{i \neq i_k} R_{ik} \right) \quad (3.11)$$

Whilst this is not immediately a good metric for linear disentanglement, the authors suggest grouping the latents based on correlation and getting the score matrix at group level. This would allow grouping based on the expected dimensions of the irreducible representations for linear disentanglement and may offer a good metric in that case.

Our Implementations The implementations of these metrics which we used for all our experiments is hosted online at <https://github.com/MattPainter01/UnsupervisedActionEstimation/tree/master/metrics>.

3.2.5 Common Assumptions

Locatello et al. [2018] present one of the largest empirical studies of disentanglement, in terms of number of models trained, and provide context for many baseline models and datasets. Through training over 10000 models on baseline datasets and evaluating a number of disentanglement metrics, they aimed to address questions such as:

- Do Disentanglement metrics agree?

- How important is model selection compared to hyper-parameter choice or randomness?
- Can we identify disentangled representations without knowing generative factors?
- Are disentangled representations beneficial?

In general, they find that the majority of disentanglement metrics correlate with each other, albeit to varying degrees across different datasets. There are some metrics which do not follow this trend, in particular the modularity score which seemed to correlate strongly with solely the BetaVAE metric. The SAP score also did not strongly correlate with other metrics.

Their next findings were more sombre. Model selection was shown to only explain 37% of the variance in disentanglement scores, with the rest being attributed to hyperparameters (22%) and randomness (41%) (i.e. choice of random seed). They also found no successful means to distinguish disentangled representations from entangled ones without supervision. It was found, however, that transferring good hyperparameters from datasets with known generative factors onto other datasets, is somewhat beneficial. This does not guarantee a disentangled model however, only providing a set of hyperparameters which might increase the probability.

For the final question, [Locatello et al. \[2018\]](#) consider the data efficiency of disentangled representations, since it was generally assumed disentanglement would correlate with efficiency. They in fact find no correlation, a finding that was consistent across all models trained, refuting this commonly held assumption. Whilst they are careful to state they have not considered all forms of disentanglement (i.e. supervised/semi-supervised) or other benefits of disentanglement (fairness, interpretability), this is still an impactful finding.

3.3 Towards a Definition of Disentangled Representations

This section will introduce the SBDRL framework which defines linear disentangled representations, which will be the focus of most of the rest of this work. SBDRL considers a slightly different problem setting to the standard VAE, and is based on the notion of symmetries of the data, defined by symmetry groups which act on it. We will introduce the SBDRL framework through a classic deep learning example which will be used throughout the later chapters. Recall from 3.2.3 that dSprites is a synthetic dataset which was generated by taking 3 shapes and rendering images for each whilst varying the following factors: x position, y position, scale and orientation/rotation. Further recall that since these factors generate the dataset, we call them generative factors.

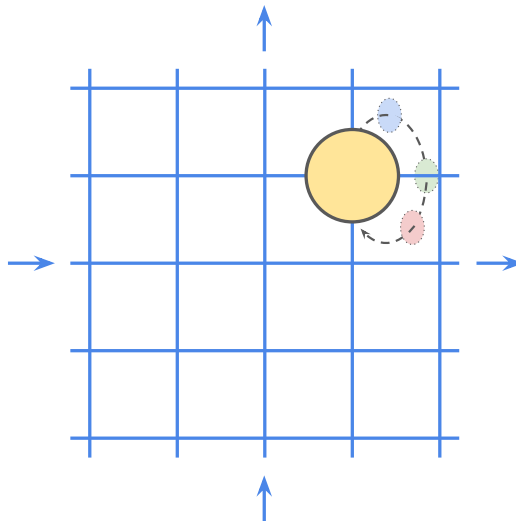


FIGURE 3.7: Visualisation of the Grid World explored by Higgins et al. [2018] consisting of a circular agent moving on a grid with periodic boundary conditions alongside a circular hue axis.

SBDRL Problem Setting Setting aside the VAE framework for the moment, the SBDRL problem statement is introduced by Higgins et al. [2018] through a reinforcement learning setting. In particular, we have an environment which provides states \mathcal{W} , and a set of actions that can be applied in each state. We will refer to \mathcal{W} as the set of world states. Higgins et al. [2018] then define the actions on this environment to correspond to its symmetries.

The example given by Higgins et al. [2018] to help visualise this scenario is a grid world with a single object, which can move around in four directions (up, down, left, right), and change colour by taking discrete steps on a circular hue axis (See Fig. 3.7). In this instance, each state in the world space is given by the triplet of (x position, y position, colour). If we define symmetries as any transformations which preserve the identity of the object, then we have 3 symmetries on this space: x translation, y translation and colour change. The effect of these transforms on the world space are then the actions in this environment.

Higgins et al. [2018] then define a generative process alongside this environment, which maps the world states to observations, given by $b : \mathcal{W} \rightarrow \mathcal{O}$. Whilst the observation space may take any form, we can assume for this work that it will be a space of images, akin to the space X from the VAE formulation. In order to study representations in this setting, Higgins et al. [2018] then introduce an inference process (representation function) $h : \mathcal{O} \rightarrow \mathcal{Z}$. The space \mathcal{Z} is then called the representation space (usually $\mathcal{Z} \subseteq \mathbb{R}^N$ for some N). The inference process can be any model which takes an observation as input and returns a representation. For the moment, we have

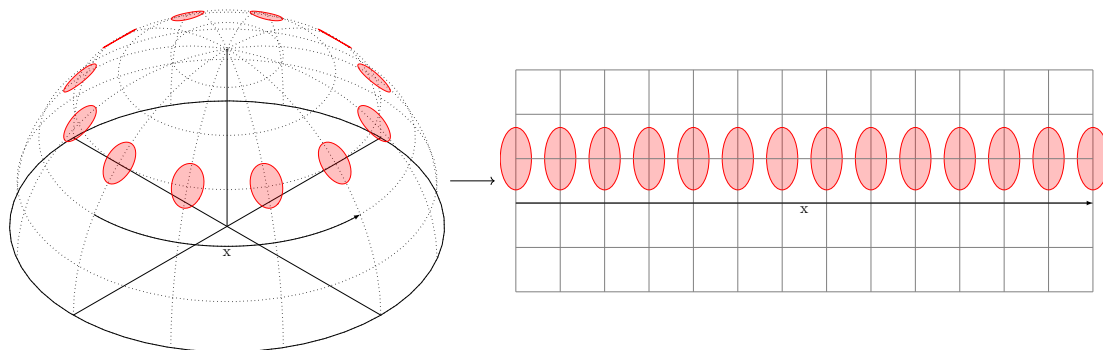


FIGURE 3.8: Visual example of wrapping on sphere. Unfolding the dome (left) as you would a world map (similar to a Mercator Projection) would result in a 2D grid (right), where we can see the object wrapping. Whilst this analogy holds for a single cyclic symmetry, if we were to include the 'y' axis, then a torus would be the correct geometric depiction.

no particular model in mind for learning the representation, it is simply a function which takes an observation/image and returns $z \in \mathcal{Z}$. SBDRL focuses on how this mapping must behave in order for it to be called linearly disentangled. We will discuss in Section 3.3.2 how we can apply VAEs in this setting.

How do Symmetry Groups Fit In Now the question becomes, how do we relate these symmetries to symmetry groups? Still considering the grid world of Fig. 3.7, let us consider just translation in the x axis. Each state is then defined by its x coordinate, $x \in \{0, 1, \dots, N_x\}$. Since we have the periodic boundary condition, when we continually increment the x coordinate by 1, once it reaches N_x , the next state coordinate would wrap back to 0. For the simple 1D case, we can analogue this to an object on a globe (See Fig. 3.8). This periodic relationship looks very similar to that of elements of a cyclic symmetry group, where $g^{n-1} = 1$. It is this relationship which motivates the initial discussion of Higgins et al. [2018].

If we think of each point on the grid in Fig. 3.8 as representing an image with the generative factor x (continuing to exclude the others for now), then we can consider the x translation symmetry. We know that if we move (for the case of Fig. 3.8) 13 steps in the x direction, then we end up in the same place. Thus consider the group C_{13} acting on the world space of x values as the relationship $g^N \circ x = x + N(\text{mod } 13)$. This group describes the x translational symmetry in our world. Naturally the same argument would hold for y translational symmetry, however in $C_N \times C_N$, the best geometric analogy would be the torus, rather than the sphere, since the Mercator projection depicted in Fig. 3.8 distorts space, particularly around the poles.

Considering dSprites for example, we have 32 values for each Cartesian coordinate, so the translational symmetries are described by C_{32} for both x and y . The group which reflects translation is then the product of these two, $G = C_{32} \times C_{32}$, with group action

$(g_x^n, g_y^m) \circ (x, y) = (x + n, y + m)$. For (2D) rotation, there exists a natural choice of symmetry group, which is the continuous rotation group $SO(2)$, however in dSprites the rotations are discrete, so it might make more sense to again consider a cyclic group (of order 40). The final symmetry in dSprites is scale. It is not clear what symmetry group best describes scale, but we can take the same trick we used in translation and define periodic boundary conditions, allowing us to consider it relative to another cyclic group C_6 . Taking these groups, we can consider the symmetry group acting on dSprites as $G = C_{32} \times C_{32} \times C_{40} \times C_6$, as we will see in Section 3.3.1.

Obviously, since we can decompose the symmetries of dSprites into a product of cyclic groups, this might be considered quite a simple problem. Indeed, as we saw in the previous section, disentanglement on dSprites is fairly simple to achieve to some degree. In the real world we are far more likely to require consideration of the full $SO(2)$ or $SO(3)$ groups for rotations in 2 or 3 dimensions, for example. Whilst the irreducibles of $SO(2)$ are relatively easy to determine, $SO(3)$ is non-abelian, and it's much harder to enumerate all irreducibles.

Disentangled Group Actions The previous paragraphs discussed groups acting on the world space of dSprites. For SBDRL, these groups will generally take the form of a product of groups, each of which describes a different symmetry and has a corresponding group action. We can then consider the representation of a VAE and how these actions change the space. In particular, we can look at whether these different symmetries are disentangled in the VAE representation space, i.e. do they effect latent subspaces that are independent of each other. We will now give the necessary definitions to explore this.

The most important definition from SBDRL is that of a disentangled group action.

Definition 3.2. An action $\circ : G \times X \rightarrow X$ of a group $G = G_1 \times G_2$ on a set X is a **disentangled group action** if there exists a decomposition $X = X_1 \times X_2$ and actions $\circ_i : G_i \times X_i \rightarrow X_i$ such that

$$(g_1, g_2) \circ (v_1, v_2) = (g_1 \circ_1 v_1, g_2 \circ_2 v_2)$$

Remark 3.3. This basically says that when we act on, for example, a world space \mathcal{W} with a subgroup in $G = G_1 \times G_2$, then only a subspace of the world space should be affected, and it should be independent of those affected by the other subgroup/s. Equivalently, if it acts on a representation \mathcal{Z} , then it should only affect a subspace independent of actions by the other subgroup/s.

We would like to move from the notion of a disentangled action on a set X to that of a *disentangled representation* $f : \mathcal{W} \rightarrow \mathcal{Z}$ (of the world space) on the representation space \mathcal{Z} . We define f as the composition of h and b , $f = h \circ b$, where only here does the \circ

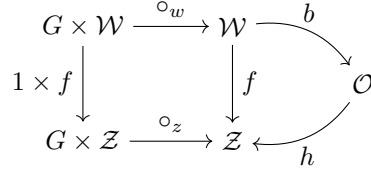


FIGURE 3.9: Function relationships for SBDRL

denote composition and not a group action. Recalling the problem setting for SBDRL in Fig. 3.10, we have the functional relationships given in Fig. 3.9. In general the generative process b will be predefined, and our models will only have access to the observation space \mathcal{O} , not the world space \mathcal{W} . As such we will often call h the representation rather than f , furthermore we will also call \mathcal{Z} a representation if it is clear from context that we are referring to a space and not a mapping.

Example 3.1. Recalling the symmetry group acting on d Sprites, $G = C_{32} \times C_{32} \times C_{40} \times C_6$, an example of a disentangled action of subgroup C_{32} on world space $\mathcal{W} = \{(x, y, s, \theta)\}$ would be if $g \circ (x, y, s, \theta) = ((x + 1) \bmod 32, y, s, \theta)$. The world subspace affected by this action is the space $\{(x, y, s, \theta) \mid x \in \{0, \dots, 31\}\}$, for any given y, s, θ . If we had the ideal f , mapping to the generative factor labels, then g is equivariant and this would form a disentangled representation.

We can now give a definition of disentanglement with respect to some symmetries acting on the data given by symmetry group G .

Definition 3.4. Given a group action $\circ_w : G \times \mathcal{W} \rightarrow \mathcal{W}$, a representation \mathcal{Z} is a **disentangled representation** with respect to G if there exists an disentangled group action $\circ_z : G \times \mathcal{Z} \rightarrow \mathcal{Z}$ such that,

$$g \circ_z f(w) = f(g \circ_w w) \quad \forall g \in G, w \in \mathcal{W} \quad ,$$

i.e. f is an equivariant map.

Remark 3.5. This equivariance ensures that the symmetry structure of \mathcal{Z} reflects that of \mathcal{W} . So whilst it is not assumed that the action \circ_w is disentangled, it's likely that we could not find a disentangled action \circ_z if \circ_w was not also disentangled.

Remark 3.6. A representation \mathcal{Z} may be disentangled with respect to more than one set of symmetries acting on the data (*i.e.* more than one symmetry group). In this case, there would exist an action corresponding to each of these symmetry groups.

For us, we can create a dataset of transitions for which each transition corresponds to action by a specific symmetry. In other words, the observations in any given transition differ by one action in the world space \circ_w . If it is then possible to find a disentangled action on the latent space then the representation can be considered disentangled with respect to the symmetry group used to generate the transitions.

Definition 3.7. The previous definition can be written in a longer, but more useful and intuitive form:

A representation \mathcal{Z} is *disentangled* with respect to $G = G_1 \times \cdots \times G_n$ if:

1. There exists action $\circ : G \times \mathcal{Z} \rightarrow \mathcal{Z}$,
2. The map $f : \mathcal{W} \rightarrow \mathcal{Z}$ is equivariant between actions on \mathcal{W} and \mathcal{Z} , and
3. There is a decomposition $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_n$ or $\mathcal{Z} = \mathcal{Z}_1 \oplus \cdots \oplus \mathcal{Z}_n$ such that each \mathcal{Z}_i is fixed by action \circ_j of all $G_{j,j \neq i}$ and affected only by G_i .

Remark 3.8. This will be important when considering metrics for linear disentangled representations in later chapters. In particular, part 3, subgroups acting on *independent* subspaces.

Linear Disentangled Representations Since our work will be concerned with representations learnt by a variational autoencoder with latent spaces forming real vector spaces, it is a good idea to consider actions which preserve their linear structure. There is a large body of work concerning linear transformations on vector spaces which revolves around group representations. A linearly disentangled representation can now be defined through the following definitions.

Definition 3.9. A group representation $\rho : G \rightarrow GL(V)$ is **linearly disentangled** with respect to group decomposition $G = G_1 \times \cdots \times G_n$ if there exists a decomposition $V = V_1 \oplus \cdots \oplus V_n$ and representations $\rho_i : G_i \rightarrow GL(V_i), i \in \{1, \dots, n\}$ such that $\rho = \rho_1 \oplus \cdots \oplus \rho_n$, that is,

$$\rho(g_1, \dots, g_n)(v_1, \dots, v_n) = (\rho_1(g_1)v_1, \dots, \rho_n(g_n)v_n) \quad .$$

Remark 3.10. This definition can be seen as the equivalent of Definition 3.4 in the specific case where the action is defined by a group representation, e.g. $\circ_g = \rho(g)$. We can think of this as a restricted case where the action is linear.

Remark 3.11. Recall from Chapter 2 (remark 2.39) where we stated two relations (without proof) that the irreducible representations of $G = G_1 \times \cdots \times G_n$ are the representations $\rho_1 \otimes \cdots \otimes \rho_n$ where ρ_i is an irreducible representation of G_i . Combining this with definition 2.34, and the assumption that V decomposes into $V_1 \times \cdots \times V_n$, then we would expect the representations to look like

$$\rho = \underbrace{(\rho_{11} \otimes \cdots \otimes \rho_{1n})}_{\text{Irreducible of } G} \oplus \dots \oplus \underbrace{(\rho_{n1} \otimes \cdots \otimes \rho_{nn})}_{\text{For each } V_i}$$

Comparing this with the definition, we can see that to be linearly disentangled requires ρ_{ij} to be irreducible, where only one of the ρ_{ij} are non-trivial for each i . In this

case,

$$\rho = \rho_1 \oplus \cdots \oplus \rho_n \quad ,$$

where ρ_1, \dots, ρ_n are the non-trivial irreducible representations.

Definition 3.4 defined a disentangled representation with respect to some symmetry group G , but the action \circ_z was not necessarily linear. We can now define a linear version of this definition through group representations, which are by definition linear.

Definition 3.12. A representation $f = h \circ b$ is a **linear disentangled representation** with respect to $G = G_1 \times \cdots \times G_n$ if there exists a linearly disentangled group representation ρ and the related action $\circ_z : (g, v) \mapsto \rho(g)(v)$ such that f is equivariant with respect to \circ_z, \circ_w .

Remark 3.13. We will see models in later chapters which can explicitly learn such ρ .

Example 3.2. If we consider the group $G = C_4 \times SO(2)$, then we know the (real) irreducible representations (Chapter 2, example 2.5) are both rotation matrices. We find that action by the group element g^k (for some $k \in \{0, 1, 2, 3\}$) and rotation angle θ will correspond to action,

$$\rho(g^k, \theta) = \rho_1(g^k) \oplus \rho_2(\theta) = \begin{pmatrix} \cos\left(\frac{2\pi k}{4}\right) & -\sin\left(\frac{2\pi k}{4}\right) & 0 & 0 \\ \sin\left(\frac{2\pi k}{4}\right) & \cos\left(\frac{2\pi k}{4}\right) & 0 & 0 \\ 0 & 0 & \cos(\theta) & \sin(\theta) \\ 0 & 0 & -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

on the representation space \mathcal{Z} .

Disentanglement with respect to symmetries This section has introduced a definition of linear disentanglement with respect to a particular decomposition of a symmetry group acting on the data. Assuming we know which symmetries are acting, and we know their irreducible representations, then this allows us to know the structure of a linearly disentangled representation with respect to that particular symmetry group. It might be that there are multiple valid symmetries for dataset, in which case a given representation may be linearly disentangled with respect to one (or more) but not others. This is a downside of the SBDRL framework, since it requires that we know (or can know) what the acting symmetries are.

It is important that the linear form of disentanglement defined (Definition 3.9) in this chapter lets us understand the structure through irreducible representations. This is not the case for the definition of disentanglement given in Definition 3.4. It is for this reason that this work will almost exclusively focus on *linear* symmetry based disentanglement.

3.3.1 Symmetries on Disentanglement Datasets

For the rest of this work, we will often need to consider symmetries acting on the disentanglement datasets defined in Section 3.2.3. Since we know the generative factors for these datasets, we can associate one symmetry with each generative factor. Furthermore, since the generative factors are discrete, the simplest way to define symmetries on these datasets is by defining group actions for each of them and determining which symmetry group they would correspond to.

FlatLand As mentioned in Section 3.2.3, FlatLand has been explored under the reinforcement learning framework, which already assigns actions to the dataset. Specifically, the actions were defined to be translating the agent by ± 5 pixels in one of the x or y axes. Since the agent can only take positions in the range $[15, 49]$, and it is warped to the opposing side when reaching a boundary, this results in the symmetry structure

$$C_7 \times C_7 \quad . \quad (3.12)$$

This is since we have $(49 - 15)/5 = 6.8 \approx 7$ possible steps in each of the x and y axis, when taking 5 pixel steps.

dSprites Whilst dSprites does not have a standard set of actions under the reinforcement learning framework, we can easily define some using the generative factors. The simplest action we can define for each factor is an action which increments the value of that factor by 1 (or some step size) and wraps back to 0 when applied to the last value of that factor. This then corresponds to a cyclic symmetry where the action is by the generator of that cyclic group.

For example, action on the x position factor by such a generator g_x , would take the form: $g_x \circ (\text{scale}, \text{rotation}, x, y) = (\text{scale}, \text{rotation}, x + 1(\text{mod } 32), y)$. If we define actions of this form for each of the generative factors (with the modulo based on the number of each factor), we get the symmetry structure,

$$C_6 \times C_{40} \times C_{32} \times C_{32} \quad . \quad (3.13)$$

Note that we do not consider the shape factor in the symmetry structure. In general, we think of symmetries as mappings that preserve the identity of an object, and not mappings that change the identity. For dSprites, the identity of the object can be thought of as the shape, and as such we don't consider changing the shape factor as a valid symmetry. This is however a design choice, and it could be that others would consider changing the shape as a valid symmetry.

Norb Similar to dSprites, the Norb dataset does not have a standard set of actions defined on it by previous works. We can use a similar approach to dSprites and define actions on this dataset based on the generative factor values. If we do this, we get the symmetry structure,

$$C_{18} \times C_9 \times C_6 \quad (3.14)$$

Similar to the dSprites dataset, we again do not consider changing the toy as a valid symmetry. The reasoning behind this is more evident in the Norb dataset, where each toy is in fact a different real world object. Saying that changing the toy is a symmetry of the toy is a particularly strange way of framing the problem.

Transition Datasets On each of the datasets we have just listed, we have an alternate version which consists of transitions rather than just states. The transitions in each case are simply a triplet consisting of a state, an action and the state resulting from applying the action to the first state. We can then create a dataset of transitions which is the set of all possible transitions on all states using all actions. For the datasets we explore, we will just be considering actions by the cyclic generators and their inverses. which reduces the number of transitions we have in each dataset.

3.3.2 VAEs as the representation function

In the previous section, we introduced the notion of a representation as simply a function that maps from an observation space (generally images) to a representation space. The generative process in SBDRL is determined by the function b , and the representation space \mathcal{Z} is not necessarily the same space as \mathcal{W} . For the VAE however, it is assumed that the dataset is generated by a generative process which depends on some latent space, which is also the space to which the VAE encoder maps. This allows us to frame the problem as attempting to optimise the VAE parameters θ towards the true generative process parameters θ^* .

As noted by [Quessard et al. \[2020\]](#), it is difficult to immediately relate the SBDRL and VAE frameworks since SBDRL assumes a deterministic simulator as opposed to the prior based generative process in the VAE. There has been recent work [Yang et al. \[2021\]](#) that aims to join the two frameworks by considering actions as permutations on the observation space. For our work, we will assume that the VAE can be applied to the dataset of observations, and not attempt to define a concrete relationship between them.

Even without a concrete relationship between these spaces we can still use the SBDRL results to understand how the VAE representation needs to look in order to be linearly

SBDRL	\mathcal{W}	\mathcal{Z}	\mathcal{O}	b	h
VAE	\mathcal{Z}	\mathcal{Z}	X	<i>Sim</i>	$q_\theta(z x)$

TABLE 3.1: Loose relationships between SBDRL spaces and spaces in the VAE framework.

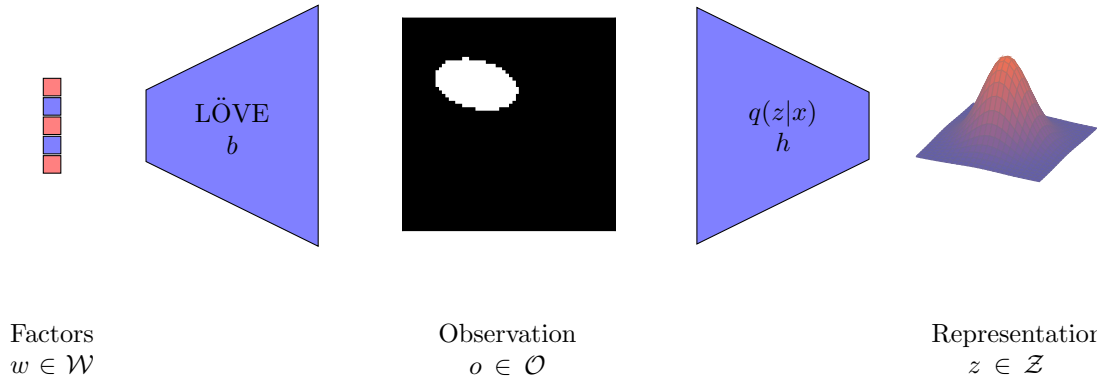


FIGURE 3.10: Visual schematic of the SBDRL problem setting for a VAE on dSprites

disentangled and how this compares to standard disentanglement. It might be useful to keep in mind the loose associations given in Table 3.1 whilst thinking about this problem.

For the specific example of dSprites, we can define the world space as equal to the set of generative factors (See Section 3.2.3),

$$\mathcal{W} = \{(s, \theta, x, y) \mid s \in \{0, \dots, 5\}, \theta \in \{0, \dots, 39\}, x, y \in \{0, \dots, 31\}, \}$$

Note that we do not consider the shape to be a generative factor. This is arguable in dSprites, since it is not clear what constitutes an ‘object’. In the real world, objects tend not to spontaneously change shape, but this is entirely possible in the world of dSprites.

For the dSprites problem, we then have that the SBDRL simulator function b is simply the code written in the LÖVE framework of Lua, which takes one such vector, say $w = (5, 18, 6, 4)$ and returns an image/observation $b(w) = o$. This is essentially a simple rendering function which draws a shape at the position defined in w with the desired rotation and scale. This is diagrammed in Fig. 3.10. Given a VAE model, the inference function h is given by $q_\theta(z|o)$, the VAE encoder. The representation space \mathcal{Z} is then given by the VAE latent space.

3.3.3 Linear Symmetry Based vs Classical Disentanglement

The general consensus for a representation to be classically disentangled is that it must encode individual generative factors into individual (and independent) latent

dimensions. Under this definition, it cannot generally be said that symmetry based disentangled representations form a subset of classically disentangled ones, or visa versa. However, in the restricted case where all the symmetries acting on the data have 1 dimensional representations, then they will be functionally equivalent. In all other cases, linear disentangled representations will encode generative factors across more than 1 latent dimension. As we saw in Section 3.2.4, the idea that factors must be encoded into single latent dimensions is a core assumption of some disentanglement metrics. As such, such linear disentangled representations would often be considered entangled under these metrics.

We can see this difference if we consider specific examples of representations for the flatland problem. On Flatland, we know the two generative factors are the x and y positions of the agent in the image. These can be within the closed intervals $[15, 49]$. We could define a classically disentangled 2D representation of this problem to encode these variables linearly in the range $[-1, 1]$, given by

$$z_{\text{Classical}}(x, y) = \left(\frac{x - 32}{17}, \frac{y - 32}{17} \right) . \quad (3.15)$$

Recall from Section 3.3.1 that we can define cyclic symmetries on Flatland by $G = C_7 \times C_7$. In this case, we can use Definition 3.9 and Remark 3.11, to see that a linearly disentangled representation can be given by,

$$z_{\text{Sym}}(x, y) = \begin{pmatrix} \cos\left(\frac{2\pi n_x}{7}\right) & -\sin\left(\frac{2\pi n_x}{7}\right) & 0 & 0 \\ \sin\left(\frac{2\pi n_x}{7}\right) & \cos\left(\frac{2\pi n_x}{7}\right) & 0 & 0 \\ 0 & 0 & \cos\left(\frac{2\pi n_y}{7}\right) & -\sin\left(\frac{2\pi n_y}{7}\right) \\ 0 & 0 & \sin\left(\frac{2\pi n_y}{7}\right) & \cos\left(\frac{2\pi n_y}{7}\right) \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad (3.16)$$

where $n_x = \frac{x-15}{5}$, $n_y = \frac{y-15}{5}$. This representation encodes x into a circle in the first two dimensions and y into a circle in the last two dimensions. The point $(x, y) = (15, 15)$ is represented by the point $(1, 0, 1, 0)$. This form of z_{Sym} comes from the fact that the irreducible representation of the cyclic group C_7 is a rotation matrix of order 7, so the direct product is given by the form in Remark 3.11, and the point $(1, 0, 1, 0)$ was chosen arbitrarily. In this specific case, we have $V = V_1 \oplus V_2$ where $|V_1| = |V_2| = 2$ where V_1 is the space spanned by $\{(1, 0, 0, 0), (0, 1, 0, 0)\}$ and V_2 is the space spanned by $\{(0, 0, 1, 0), (0, 0, 0, 1)\}$.

We can see that there is significant difference between the classical and symmetry based representations, not least being that the classical representation can be defined on \mathbb{R}^2 and the symmetry based representation must be defined on \mathbb{R}^4 . Furthermore, each of the Cartesian dimensions are encoded across two latent dimensions in the symmetry based representation and across one in the classical representation.

The real difference between symmetry based and classically disentangled representations is that through SBDRL, we have a stronger basis for what the representation should look like.

In the example we just gave, the standard disentangled representation could take any form where the factors are encoded into separate dimensions. There are many transformations or changes that can be made which would result in a disentangled representation, for example, scaling, rotation or dimension permutation. These transformations can be made asymmetrically for either of the factors also. Furthermore, there is no requirement that the states be encoded linearly, so consecutive x positions may not be equally separated in the latent space.

For symmetry based disentanglement (assuming the cyclic symmetry), there are only two ambiguities, the point $(1, 0, 1, 0)$ (i.e. the representation can be rotated) and the order of $V_1 \oplus V_2$ (i.e. the x and y factor subspaces can be swapped). The result is that we have a much better idea of what the structure of the representation will be before we even train a model.

Chapter 4

Learning Linear Disentangled Representations

This chapter will cover methods to learn linear disentangled representations and how we might quantify them. We will aim to answer the following questions:

- Can we learn linearly disentangled representations?
- How do we measure them?
- Are they learnt by standard VAE models?
- Can we learn them without action labels?

To answer the first question, we first perform a two part feasibility study in Section 4.1. In the first part (Section 4.1.1), we show that we can learn a disentangled representation in the supervised setting - where we learn an encoder that maps from observations to generative factors, and a decoder that can map from observations to generative factors. Since linear disentanglement requires equivariance between actions in world space and latent space, in the second part (Section 4.1.2), we show that we can learn to approximate the effect of world space actions in latent space. We show this both using an arbitrary neural network, and also by learning a mapping to space where the action is forced to be equivariant. Having shown that each part of the problem is feasible in isolation, we then move on detailing ForwardVAE [Caselles-Dupré et al., 2019] (Section 4.2) before introducing our proposed model for linear disentanglement, GroupVAE [Painter et al., 2020a] (Section 4.3).

For the second question, we introduce in Section 4.4, a number of our proposed metrics for measuring different aspects of linear disentanglement. We introduce the Independence score, SVD overlap and relative latent error for this purpose, and

expand the mutual information gap metric to better deal with linear independence with the Factor leakage metrics.

To address the third question, we perform an empirical study of the representation spaces for a number of standard VAE models in Section 4.5. We attempt to explicitly optimise representations of actions on the latent space such that they are equivariant to those actions on the world space. By determining if this is possible, and the extent to which the actions are equivariant, we can determine whether the representations spaces are linearly disentangled.

To answer the final question, we explore (in Section 4.6) GroupVAE, our proposed model to learn linear disentanglement without access to labelled actions/transitions.

4.1 Actions on VAE Subspaces

This section will act as a feasibility study to demonstrate that we can learn to approximate group actions on VAE subspaces. In particular, that we can learn them such that are equivariant to those on the world space, as is required by the definition of linear disentanglement from Definition 3.7. This will allow us to answer in part the first question posed in the previous section, “Can we learn linearly disentangled representations?”. To do this, we break the problem down into idealised parts, where if we can learn all parts of the problem effectively, then it suggests that we might be able to learn it as whole. First we show that we can learn an idealised encoder, then an idealised decoder, before finally showing that we can approximate actions on VAE spaces. Finally, we show that when we know the symmetry structure, we can learn a mapping from the VAE latent space to a linearly disentangled one with this symmetry structure.

4.1.1 The Idealised Model

We consider the idealised model where each component is learnt independently and with ideal inputs. We shall learn first a disentangled classifier, then a disentangled simulator and finally we examine two means to directly learn actions.

Classifying Disentangled Factors A model which learns to represent images by their exact generative factors would be the perfect model with regards to classical disentanglement. This also extends to linear disentanglement since a linearly disentangled representation could likely be obtained with minimal computation, certainly if the symmetry structure was known. We now demonstrate that a simple

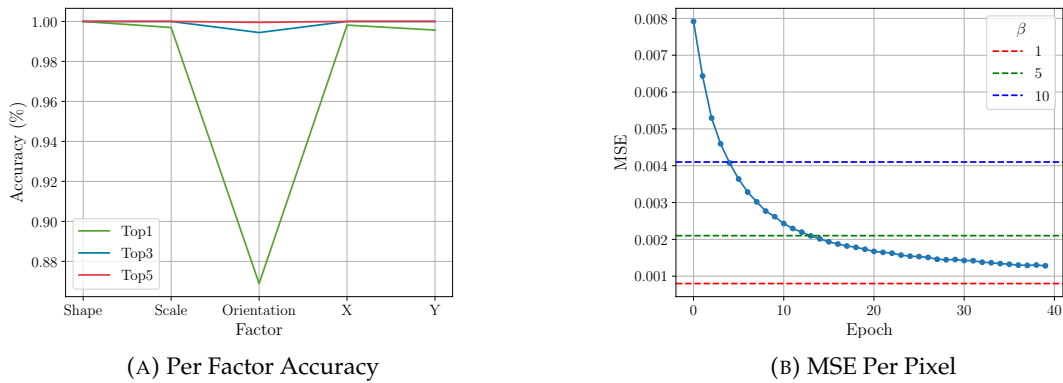


FIGURE 4.1: Idealised inference (A) and simulator (B) results. (A) We can classify dSprites generative factors directly from pixels to a high degree of accuracy. (B) We can learn to simulate dSprites images from the generative factors.

MLP can learn to classify dSprites generative factors (shape, scale, orientation, x , y) directly from pixels, whilst under supervision.

To do this, we take an 3-layer MLP with 1200 hidden units and train it for 40 epochs using back propagation to classify from dSprites pixels (64×64 px) the generative factor values for that image (see Section 3.2.3). We train using the Adam optimiser [Kingma and Ba, 2014], batch size of 64, learning rate of 1×10^{-4} and a cross entropy loss. We then report the per class classification accuracy using an independent (from training) validation set comprised of 10% of the total dataset.

From Fig. 4.1a, we find that individual factors can be classified to a very high accuracy (99%) with the sole exception being orientation which proved comparatively difficult (88%). This could be due to small visual differences between sequential orientation observations, however for dSprites the visual difference for orientation is approximately 0.0103 per pixel. This is lower than that for translations (approx 0.0150) but higher than that for scale (approx 0.00897). If we normalise by the number of possible values each factor can take, we find a relationship similar to our prediction scores ([27.5206, 0.001495, 0.00026, 0.0005, 0.0005]). Since these numbers don't quite align with Fig. 4.1a (scale and translation are equally predictable but do not vary equally), it is likely there are additional factors at play such as biases inherent in the model choice.

We know that models are unlikely to ever learn the exact generative factor values as their representations. Generally VAE representations are continuous (with exceptions [Maddison et al., 2016]) and generative factors are discrete. However, this shouldn't prevent them learning representations which hold similar structure to the generative factors. An example of such structure could be encoding generative factors into independent subspaces as required by classical and linear disentanglement. Another example could be preserving the relations between states in each factor, i.e.

an object rotated by 2° is encoded close to that of the original object, but an object rotated by 50° is encoded far away. This kind of structure is naturally present in linear disentangled representations when the symmetry structure reflects the visual structure, which occurs when subgroups talk about the same concepts as the generative factors. Furthermore, this kind of structure is present in standard (disentangled) VAEs also, since variational sampling leads to smoothness with respect to output observations. A good understanding of this is provided by Burgess et al. [2018].

Learning a Disentangled Simulator Recall from Section 3.3 that SBDRL has two mappings, a generative/simulator process b and an inference process h . The previous paragraph can be seen as attempting to learn the inverse simulator b^{-1} as the inference process. We now attempt to learn the simulator process (in a supervised manner) for dSprites. In the VAE setting this can be considered as learning the decoder when given idealised latent encodings (i.e. the generative factors).

To do this, we train a 4 layer MLP (1200, 1200, 1200, N_{pixels} units) to generate the observation corresponding to the generative factors over a period of 40 epochs. We optimise using the Adam optimiser, batch size of 128, learning rate of 1×10^{-4} and a Binary cross entropy loss. We can then report the validation mean squared error (MSE) per pixel (Fig. 4.1b) for a validation set comprising of 10% of the total data. For reference we also provide results for trained β -VAEs with $\beta = 1, 5$ and 10 on the same validation data.

We can see that the simulator generates images with low error suggesting that the decoding task of a VAE is simple compared to encoding, at least in this setting. Obviously these results should not be surprising given the wealth of works applying VAEs to disentangle dSprites. It is still useful to consider idealised cases before attempting to solve the problem as a whole, not least to understand which aspects of the problem may be difficult.

4.1.2 Approximating world space actions

Given that we can learn idealised versions of the simulator and inference processes, the question remains as to whether we can learn to approximate actions on pretrained VAE subspaces. This section will form the second part of the feasibility study, where we explore learning to approximate world space actions on the latent space.

Learning Actions on VAE Subspaces Recall from 3.1 that a VAE has encoder parameters ϕ and decoder parameters θ . Setting aside the task of learning (group) representations for now, we look instead at the easier task of supervised modelling

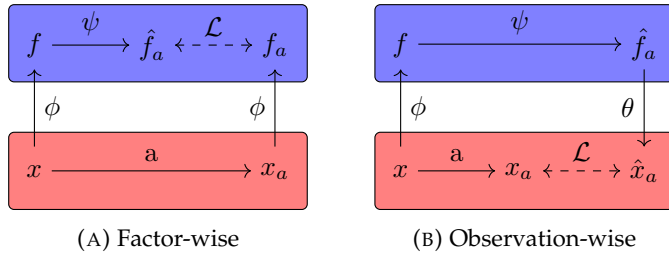


FIGURE 4.2: Comparison of unsupervised and supervised methods for learning actions a on pretrained VAE subspaces, defined by encoder ϕ , decoder θ and action estimator ψ . They differ in where the loss (\mathcal{L}) is computed. The loss is between two quantities joined by the dashed line. When supervised, the loss is computed in the latent space (blue) and when unsupervised it is in the image/input space (red).

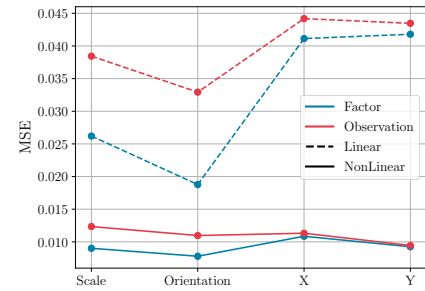


FIGURE 4.3: Post Action Observation reconstruction error for each dSprites factor using the factor-wise and observation-wise models.

where the action label is provided to the model. We will consider the case where the group acting on dSprites has subgroups corresponding to generative factors and actions (by cyclic generators) which increase the appropriate factor by exactly 1. As such the symmetry structure is $G = C_3 \times C_6 \times C_{40} \times C_{32} \times C_{32}$.

We consider two idealised methods to learn actions, the factor-wise and observation-wise settings. Schematic diagrams of the learning process for both models are given in Fig. 4.2.

The factor-wise model trains an MLP with parameters ψ to predict post action factors with an MSE loss between predicted and true post action factors. The observation-wise model uses the same idea but decodes predicted factors to post action observations, which then has a loss signal provided by the binary cross entropy between predicted and true post action observations, avoiding knowledge of the exact post action factors.

In both cases, we use the same pretrained VAE, a CC-VAE with capacity 25 annealed over 100000 iterations and 10 latent dimensions. This has the backbone of a β -VAE with the architecture as defined in section B.2, which follows the CelebA [Liu et al., 2015] experiments of the original paper [Higgins et al., 2017]. The CC-VAE was trained on dSprites for 100 epochs using a binary cross entropy loss, a batch size of 128, a learning rate of 1×10^{-4} and the Adam optimiser. We can then report the MSE between the reconstructed image and the true image when the action changes only a single factor. This MSE is given on a 10% validation set, independent of the training set.

For both factor-wise and observation-wise settings, we train a linear (linear map) and non-linear (3 layer, 1 hidden) MLP to determine the complexity of the problem. The linear model was just a single dense layer between input and output. The non-linear

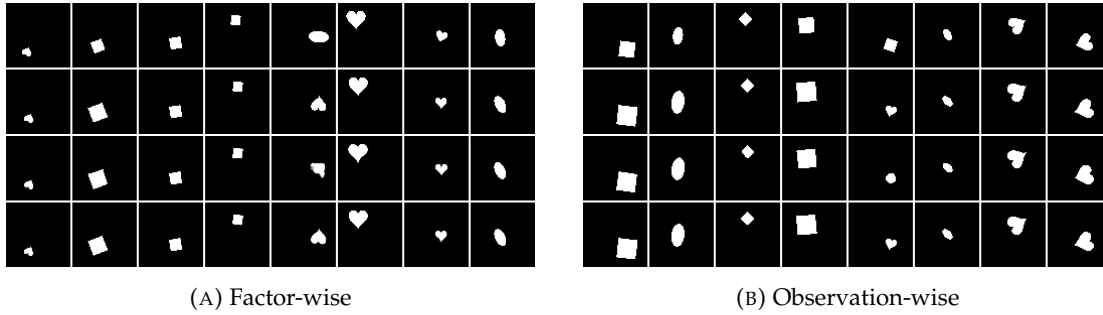


FIGURE 4.4: Some examples of an input image (top row), the observation after applying an action in world space (second row), the reconstruction after applying the learnt action (third row) and the true reconstruction of the second image (bottom row). The models learn to apply most actions fairly well, and can't learn to change the shape. Generated with CCVAE and capacity 25. Note that actions are generally visually small (change only a few pixels). Actions are most noticeable in scale or shape change.

model was a 3 layer MLP with 64 hidden units. In all cases, the MLPs were trained with a batch size of 256, the Adam optimiser with learning rate 1×10^{-3} and 100 epochs. The factor-wise model was trained using a binary cross entropy loss and the observation wise model was trained using a mean squared error loss. All results are evaluated on a validation set of 10% of the whole dataset.

In Fig. 4.3, we give the expected MSE for linear and non-linear prediction models on individual factors. We can see that the linear models fail to learn the action in both cases, whereas the non-linear models can predict the action to a good degree. From the example reconstructions in Fig. 4.4, we can see that both models perform adequately, generally reconstructing the post-action observation decently.

Learning actions on VAE subspaces is obviously possible, and it seems likely that fully end to end models could learn similar structures. However, neither of these models necessarily learn actions in independent subspaces since this is decided by the underlying (pretrained) VAE. We will perform a similar experiment to search for linear structures in backbone VAE models in Section 4.5.

Learning known structure The previous experiments have shown that we can learn actions on VAE subspaces. However they did not allow us to extract symmetry structure easily and the actions were free to depend on the latent code. For this problem and our defined (cyclic) symmetries, we know the irreducible representations (rotation matrices) which should be learnt in a linearly disentangled space.

We now take the representation of a VAE and instead of attempting to directly optimise a neural network to approximate the actions, we learn a mapping to an intermediate space where the actions are cyclic. Since the previous experiment showed that a linear model could not learn actions, we only consider non linear models for this task. We use a small MLP, mapping to an intermediate space where we

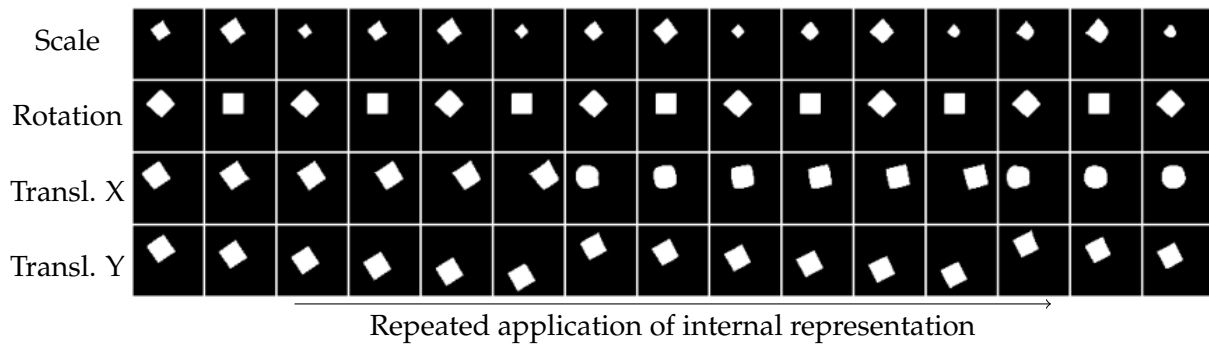


FIGURE 4.5: Visualisation/Orbit of learnt actions (Scale, Rotation, X translation and Y translation) on dSprites. Taking the same initial image, each row shows a different internal representation / action as it is repeatedly applied. There is some degradation as actions are repeated (see x translation in the third row) although most actions are of good quality.

apply the known irreducible representation for the observed action. We can then apply a (learnt) inverse map to return to the VAE latent space, where the image is decoded and compared to the post action image. The goal is for the intermediate space to be linearly disentangled with respect to the defined symmetries, whilst the VAE latent space need not be. We know that an intermediate space exists for which actions are linear (at least locally), since this is the final layer of the non-linear model in the previous experiment. The difference here is that we want it to obey the same structure as the symmetries (cyclic), in addition to being linear.

For this experiment, we use the same underlying VAE (CC-VAE) as the previous experiment, trained under the same procedure. The MLP which maps from the VAE latent space to the intermediate space has 3 layers with (128, 128, N_{out}) units. The inverse model, which maps from the intermediate space back into the VAE latent space was an MLP with the same structure but mirrored. These were trained over 300 epochs with the adam optimiser, using a learning rate of 10^{-4} , batch size of 256 and a binary cross entropy loss. MSE results are reported on a 10% validation set which was independent of the training set.

In Fig. 4.5, we demonstrate the actions learnt on this task via latent traversals. We see good quality action traversals, albeit with some degradation as they are composed. In Fig. 4.6a we show the MSE of a single action for each factor, where translation errors are very low and the scale and orientation errors are slightly larger. Note that these errors are generally smaller than those from the previous experiments, and despite long training times, the model was still converging, so these errors could be improved. In Fig. 4.6b we quantify the degradation by comparing the result obtained when taking N steps of an action to the image obtained when applying the action in world space. The error scales roughly linearly in all cases and with similar rate of increase. We can see a saw tooth type shape in the Scale factor, with a period of roughly 3 steps,

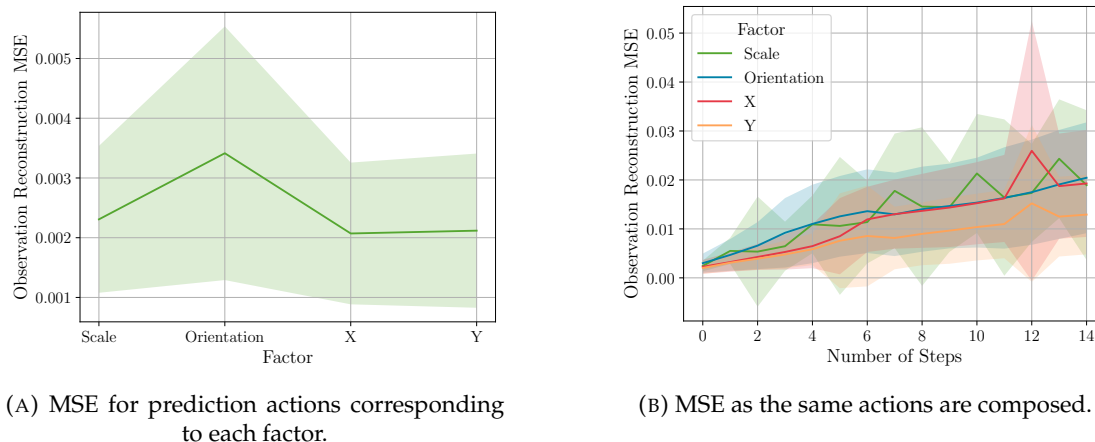


FIGURE 4.6: Evaluating a model trained to learn cyclic structure on pre-existing latent spaces. (left) MSE for a single action. We don't see the degradation from Fig. 4.5, suggesting that locally, the action estimation is good. (right) We see that the error gradually increases as the actions are composed. Solid lines indicate mean performance with 1 standard deviation given as the shaded area.

which correlates with the number of scale factors. The error increases when the shape cycles from large to small, and gradually decreases as the shape gets larger, before cycling again.

Learning a mapping to this known structure was surprisingly difficult, taking a large number of iterations to train to a acceptable degree. This emphasises that the task of transforming a standard disentangled representation is not as simple as it may seem, even if we are effectively just parametrising a circle for cyclic symmetries. We have found that on FlatLand, a simpler dataset than dSprites, we can learn the model in under 5 epochs, however this has only two factors and no visual variation other than translation. Its possible a larger model may allow for much faster dSprites convergence, however this would still imply that the task is more complicated than it seems. This result will be useful to remember in later sections where we train end to end models to learn linear disentangled representations, obviously this is not a simple task.

4.2 ForwardVAE

The previous section aimed to learn linear disentangled representations from a predefined VAE representation. Caselles-Dupré et al. [2019] introduced ForwardVAE as the first model to learn linear disentangled representations jointly with a VAE model. In this section, we will detail the ForwardVAE model as an example of how to learn linear disentangled representations end-to-end.

The intuition behind ForwardVAE was that by observing action transitions (pre and post action observation pairs), we can learn internal matrix representations to approximate actions in the latent space. Since these are linear maps, encouraging them to approximate actions forces the latent space to be structured in a way that the actions can be represented linearly. As such, we are encouraging equivariance between the actions on the world and latent spaces, whilst also encouraging them to be linear, which is required by the definition of a linearly disentangled representation (Definition 3.12). We can think of the internal representations of ForwardVAE to be approximations of the representations $\rho(g)$ in Definition 3.9.

In order to train ForwardVAE, at each training step, it is provided an observation x associated with the world state w and an action a . ForwardVAE stores a number of internal representations $\{A_i\}$ each corresponding to one of the possible actions that may be applied. Caselles-Dupré et al. [2019] enforce A_i to take the form:

$$A_i = \begin{pmatrix} & & & & \\ & \mathbf{U}_i^\delta & 0 & 0 & \\ & & 0 & 0 & \\ & 0 & 0 & & \\ & 0 & 0 & & \mathbf{L}_i^{1-\delta} \end{pmatrix}, \quad (4.1)$$

where $\delta \in \{0, 1\}$ and U, L are learnt, i.e. A_i has the form of an identity matrix with either an upper or lower (2D) block being learnt. This structure reflects the expected linearly disentangled structure of FlatLand, $C_{13} \times C_{13}$, which has block diagonal irreducibles (recall Remark 3.11) in the form of rotation matrices. The block diagonal structure is what forces the representation to be disentangled with respect to the symmetry groups acting on the data, since the actions by different symmetry groups are mapped to different blocks.

ForwardVAE is optimised by first passing x through the VAE encoder, resulting in latent code z . The internal representation A_a associated with action a is then applied to the latent code, resulting in an estimate of the post action latent $\hat{z}_a = A_a z$. This can then be compared to the true post action latent code z_a obtained by encoding the post action image x_a , i.e. the observation corresponding to world state $w_a = a \circ w$.

The objective can then be formed,

$$\mathcal{L}_{\text{ForwardVAE}}(x, z_a) = (x - \hat{x})^2 + \text{KL} + (z_a - \hat{z}_a)^2, \quad (4.2)$$

where KL is the usual VAE KL divergence loss. This process is diagrammed in Fig. 4.7a. This is an action supervised process, since selection of the internal representation A_a is dependent on knowledge of which a was applied. Note that our model GroupVAE will be introduced to avoid this restriction.

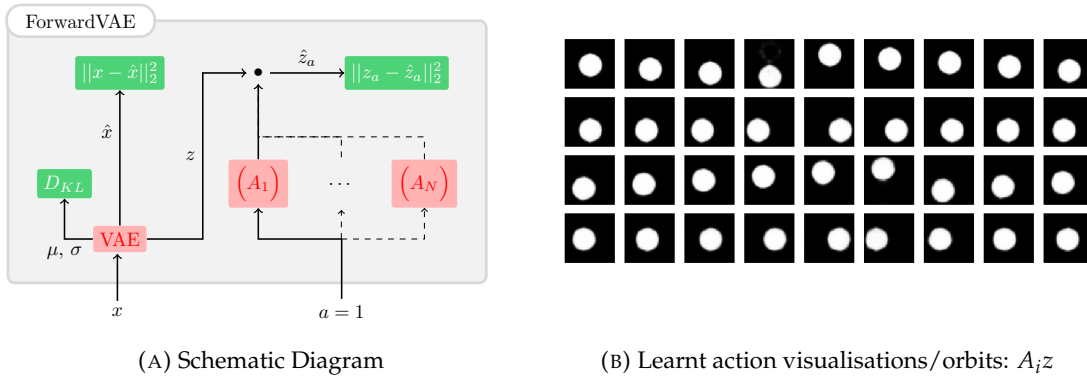


FIGURE 4.7: ForwardVAE

The training procedure introduced by Caselles-Dupré et al. [2019] successfully resulted in learning linear disentangled representations on FlatLand. This can be seen through visualisations of the learnt internal representations (A_i) in Fig. 4.7b, where each representation (row) clearly corresponds to an independent action (down, left, up, right). Since there is no mixing of actions in the visualisation, it must be that the representations corresponding to horizontal translations have the opposing δ value to those corresponding to the vertical ones. In other words, the two component groups of the FlatLand symmetry are encoded into independent latent subplanes, and thus are linearly disentangled. This was further demonstrated by Caselles-Dupré et al. [2019] through explicitly examining the internal representations A_i and comparing them to the known irreducible representations. It was found that the average difference between an internal representation A_i and the known irreducible representation of C_7 (rotation matrix with rotation angle $\alpha = 0.9$) was of the order 10^{-4} .

4.2.1 Learning Observed Actions

ForwardVAE learns to represent actions that it observes in the data, a concept which has some prior work in the reinforcement learning space. Rybkin et al. [2018] learn a (composable) mapping from pre to post-action latents, conditioned on observing the action. Edwards et al. [2019] learn both a forward dynamics model to predict post action states (given state and action) and a distribution over actions given the initial state. Contrary to ForwardVAE and our work, both methods allow arbitrarily non-linear actions (parametrised by neural networks) which makes them unsuitable for learning linear disentangled representations. Furthermore they differ significantly in implementation. Thomas et al. [2017] utilise an autoencoder with a ‘disentanglement’ objective to encourage actions which correspond to changes in independent latent dimensions. They use a similar reward signal to that we will use in Section 4.3, however encourage no latent structure other than that single latents should vary with single actions. Furthermore, they require action labels, which will be

unnecessary for our work. Choi et al. [2018] learn to predict actions between two frames in an (action) supervised manner with a focus on localising the agents in the image. Whilst these methods all learn actions, they do not learn them with regards to a particular symmetry structure and thus will not consistently form linear disentangled representations.

4.3 Unsupervised Action Estimation

ForwardVAE explicitly requires the action label at each step in order to choose the correct internal representation to apply, and thus it cannot learn linear disentangled representations in their absence. We propose jointly learning to estimate the observed action alongside the latent representation mapping. This will enable learning of linear disentangled representations when action labels are not known, extending the domain of application. For our proposed solution, GroupVAE [Painter et al., 2020a], we will introduce two variants. First we describe the overall structure of GroupVAE without the specifics that lead to the two variants.

GroupVAE GroupVAE will follow a similar overall structure to ForwardVAE. We have a backbone VAE for encoding images to a latent space and decoding back into observation space. There is an associated VAE loss term in the objective \mathcal{L}_{VAE} which comprises of the standard reconstruction (likelihood) loss ($\mathcal{L}_{\text{recon}}$) and the KL divergence term (KL). Alongside this, we have a number of learnable internal representations (A_i) which serve a similar purpose as in ForwardVAE. Importantly, in GroupVAE, the number of internal representations is not necessarily equal to the number of true actions, since this would require knowledge of the action labels. GroupVAE also has a set of parameters (ψ), in the form of a small (convolutional) neural network, which allow us to infer from the image pair (x, x_a) a distribution $p(A_i|x, x_a)$ over the internal representations, noting that a is unknown. The input to this network will be the concatenation of the image pair and the output will be a distribution¹ which is akin to a policy in RL terms. Whilst the structure of this network will be determined by the complexity of the problem (how hard is it to determine what action occurred), for all our experiments, it will take the form described in appendix Section B.6.

Our aim is optimise the distribution p to associate each action a with a particular internal representation A_i , and simultaneously optimise A_i to be an irreducible representation of a . When A_i is a good approximation of the irreducible representation, we get a good estimate of the post action latent z_a by $\hat{z}_a = A_i z$. The particular way we use the distribution p to associate actions and internal

¹In reality a probability mass function

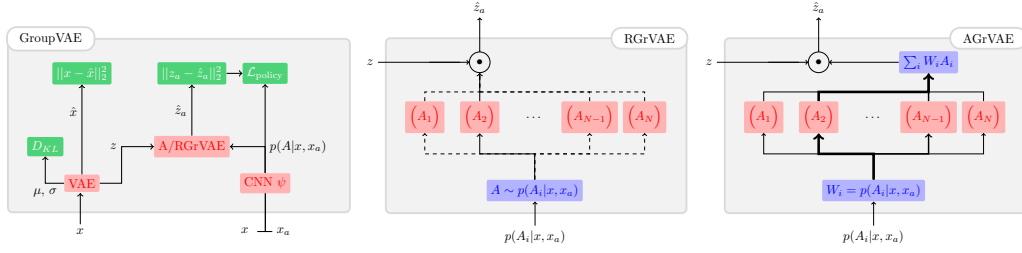


FIGURE 4.8: Schematic diagram for the GroupVAE variants. Dashed lines denote possible paths dependent on policy. Weighted lines represent the strength of attention.
■ - Learnable module. ■ - Loss. ■ - Operation without parameters.

representations is the differentiator between the two GroupVAE variants AGrVAE and RGrVAE. Having chosen a variant, we can compute the post action latent objective $\mathcal{L}_{\text{Latent}}$. The GroupVAE objective is a function of the input observation x and the post action latent z_a for some action a . It is given by,

$$\mathcal{L}_{\text{GrVAE}}(x, z_a) = \underbrace{(x - \hat{x})^2}_{\mathcal{L}_{\text{Recon}}} + \underbrace{\text{KL} + \gamma (z_a - \hat{z}_a)^2}_{\mathcal{L}_{\text{Latent}}} + \mathcal{L}_{\text{Policy}} \quad , \quad (4.3)$$

where $\mathcal{L}_{\text{Policy}}$ is only used for RGrVAE. Fig. 4.8 provides a schematic of the general GroupVAE structure where the variant decision making is encapsulated by the module labelled A/RGrVAE.

Action Selection We consider two possible methods for action selection, given the distribution over internal representations $p(A|x, x_a)$. The first variant is known as Reinforced GroupVAE (RGrVAE) and uses policy gradients to optimise ψ and select an internal representation. The second is known as Attentional GroupVAE (AGrVAE) and uses an attention mechanism for the same purpose.

Reinforcement - Policy Gradients One of the core algorithm families in reinforcement learning is Policy Gradient methods. Using a policy gradient method, we can directly sample from distribution $A_i \sim p(A|x, x_a)$, and we have the GroupVAE post action latent estimation

$$\hat{z}_a = A_i z \quad .$$

Since we are optimising the parameters ψ which generate p , we need to be able to compute gradient through the Categorical sampling. In the REINFORCE setting [Williams, 1992], the policy gradient where action A_i receives reward $R(A_i, s)$ is given by,

$$\mathcal{L}_{\text{policy}} = \begin{cases} -\log(p(A_i|\psi, s)) \cdot R(A_i, s) & \text{if } R(A_i, s) > 0 \\ -\log(1 - p(A_i|\psi, s)) \cdot |R(A_i)| & \text{if } R(A_i, s) < 0 \end{cases} \quad . \quad (4.4)$$

The intuition being that actions with high reward will be encouraged and negative rewards ensure bad actions are discouraged (by encouraging all others). Since this loss is defined solely by the policy distribution p , and not the sampled action A_i , the gradient is free to pass backwards through the parameters ψ . The choice of reward is important in this setting since it drives the whole process. We could envisage a number of different reward functions for this problem, such as:

$$R_{\text{recon}}(I_1, I_2, \hat{I}_2) = \sum(I_2 - \hat{I}_2)^2 - \sum(I_2 - I_1)^2$$

$$R_{\text{binary}}(z_1, z_2, \hat{z}_2) = \begin{cases} 1, & \text{if } \sum(z_2 - \hat{z}_2)^2 < \sum(z_2 - z_1)^2 \\ -1, & \text{otherwise} \end{cases}$$

However, we will choose the reward to be,

$$R_{\text{latent}}(z_1, z_2, \hat{z}_2) = \sum(z_2 - \hat{z}_2)^2 - \sum(z_2 - z_1)^2 \quad . \quad (4.5)$$

This reward encourages actions that result in predictions of the post action latent which are close to the true values. This has a couple of advantages. First, it is very similar to the latent loss used in ForwardVAE. Second it only requires a pass through the encoder, and not the decoder, as is required by R_{recon} . Third, it provides stronger signal the better our predictions become, unlike R_{binary} which provides a constant signal and could result in multiple choices being rewarding for each action.

The final consideration of policy gradient methods is an exploration strategy. It is possible for the policy network to collapse and only allocate density to one of the possible selections, thus only giving gradient to this representation. To avoid this, exploration methods are used. For RGrVAE, we will consider entropy weighting (used by methods such as soft-actor critic [Haarnoja et al., 2018]) and random action selection. We will briefly explore these methods in Section 4.6. A schematic of the reinforcement method is given in Fig. 4.8 (mid).

Attentional Mechanisms The term attention in machine learning can refer to a few different concepts. The particular form we are interested in is a simple concept which was popularised in language models and essentially weights each possible choice by how “important” it is.

For GroupVAE, these choices are the possible internal representations A_i , and the attentional mechanism simply takes the distribution $p(A_i|x, x_a)$ and applies it linearly

to the choices. Thus the post action latent estimation is given by,

$$\hat{z}_a = \sum_i p(A_i|x, x_a) A_i z \quad . \quad (4.6)$$

A schematic for the attentional mechanism is given in Fig. 4.8 (right).

Attention or Reinforcement? Given that we have two methods that perform the same function, we now discuss some advantages and disadvantages. We will empirically compare them in Section 4.6, but for now we focus on the mechanics of each method and how they might affect learning.

The major value of a reinforcement mechanism is that we select and apply only a single choice from $\{A_i\}$, which is not possible in the attentional mechanism. This allows us to find exact representations of actions, which we can compare to known irreducibles as a performance metric. The downside of reinforcement methods is that they are generally harder to train and slower to converge and run.

The advantage of the attentional mechanism is that the gradients are consistent throughout $\{A_i\}$, coming directly from reconstruction performance of z_2 . This will generally lead to more stable training, whereas reinforcement learning is known to be relatively unstable and difficult to tune. Attention also ensures that all actions are updated, which is not guaranteed by the reinforcement method, since the policy can freely ignore individual actions. The downside of attention is that applying a linear combination of $\{A_i\}$ means that no particular A_i need correspond to any particular action. Whilst this could be mediated by finding the linear combination chosen for each action, there is no guarantee that this will be constant across the latent space. This may prevent us from comparing to known irreducibles which limits the usefulness of the attentional variant.

Finally, there are runtime and memory differences between the two methods. Whilst attention on paper requires more operations to apply each action (each internal representation is multiplied), they are extremely optimised on modern hardware. Reinforcement however has the additional overhead of REINFORCE, which is exacerbated when memory intensive methods such as regret minimisation are used. We will briefly explore runtime and memory requirements at the end of this section.

Internal Representations GroupVAE, like ForwardVAE, houses a number of internal representations, which we have denoted $\{A_i\}$. In the ideal case, each internal representation A_i will represent some true action a and will take the form of an irreducible representation of that action. For example, if action a is by cyclic generator $g \in C_N$, then the internal representation which learns to represent a will take the form

of a rotation matrix of degree N . Naturally, for GroupVAE to be able to optimise A_i towards (for this example) a rotation matrix, A_i must be able to take the right form. A_i cannot be, for example, a matrix of size 1. ForwardVAE chooses these representations to be 2D matrices without any additional structure. Quessard et al. [2020] have similar representations (in a different context) which they restricted to be rotation matrices.

For GroupVAE, we can use either of these approaches and we will see in later experiments that either can be effective. To represent rotation matrices, we can learn a cyclic angle α for each representation, and upon applying the action we can compute,

$$\rho(g) = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} .$$

We will often talk about the internal representations as being related to a particular symmetry group. The rotation matrices would be related to C_N , or $SO(2)$. Similarly, the generic 2D matrices used by ForwardVAE would be related to $GL(2)$. For standard disentanglement, which encodes factors into a single dimension, we could consider an Integer group $(\mathbb{Z}, +)$ with presentation $\langle t \rangle$, 1D representation $\rho(t) = (\tau) \in \mathbb{R}$, and learnable τ . Another possible group with simple irreducibles is D_N , it is generated by a cycle and reflections, the reflection elements can be represented by,

$$\rho(\sigma_x) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \rho(\sigma_y) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

Change of Basis Thus far, we have assumed that the actions of dimension greater than one should be encoded into sequential latent dimensions, an assumption also made by other models in this space ([Caselles-Dupré et al., 2019]). This is not required however, since we can learn a change of basis alongside each internal representation, which can be applied to latents before computing the post action latent. A true change of basis would have columns forming a basis of the latent space (or a real vector space of the same dimension), but restricting a learnable matrix to have this property is difficult. We could imagine normalising columns and adding additional losses such as full rank losses which might help, but will likely be expensive (could involve computing SVDs). Instead we choose to simply initialise a 2D ‘change of basis’ matrix $B \in \mathbb{R}^{N \times N}$ to the identity and assume that the process of optimising representation A under the change of basis $z_a = B^{-1}ABz$ learns an appropriate mapping. This does leave open the possibility for the change of basis to map into lower rank spaces, however in this case the inverse is not computable and the optimisation will fail. In practice, we tend to include an additional loss on the determinant such that it is close to 1, and consequently, encourage invertibility.

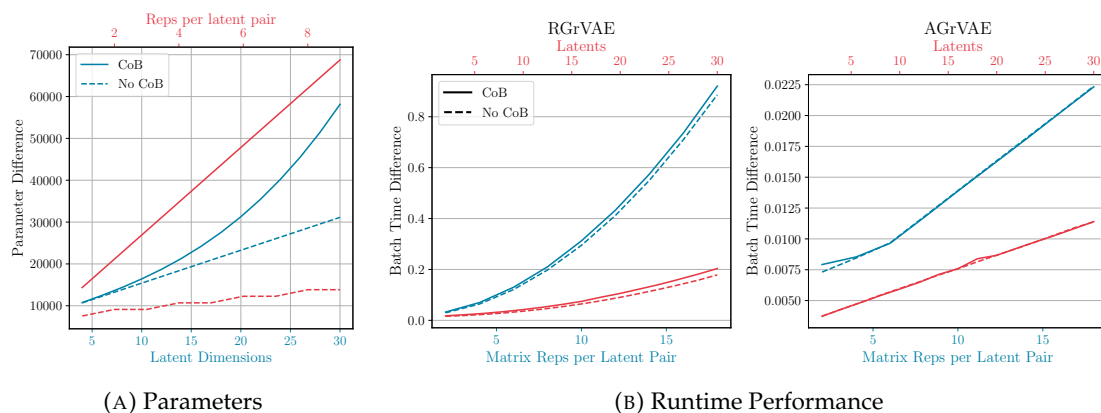


FIGURE 4.9: Parameter count and runtime charts for RGrVAE on FlatLand, both given in excess of those of the underlying VAE. Solid lines denote GroupVAE with a change of basis matrix also learnt. Dashed lines denote GroupVAE without a change of basis. When not varying the latent dimension number, we used 8 latent dimensions, and in all experiments used cyclic internal representations. Batch times were estimated on FlatLand with a batch size of 128 over 10 batches.

Assuming we can learn such a mapping, a change of basis may be useful if the latent space has existing structure which encodes latents into non-sequential dimensions. This focuses the learning task less on rearranging the latent space and more towards learning the actions and finding the spaces in which they are encoded. Whilst the change of basis does add parameters and runtime, it also offers flexibility which may be worth the additional complexity.

Properties GroupVAE introduces parameters ψ and A_i on top of the standard VAE. In Fig. 4.9a we chart this parameter count as we vary the number of latent dimensions or the number of internal representations. Note that RGrVAE and AGrVAE have the same number of parameters, however introducing a change of basis will increase the number of parameters in proportion to the size and number of internal representations. We plot this with and without a change of basis matrix. There is roughly a linear increase in parameters without a change of basis with respect to both latents and representations. When a change of basis is used, we see linear increase with representations, and polynomial increase with latent dimensions. Typically however we are using a low number of latent dimensions, so this should not be overly restrictive.

Fig. 4.9b compares the runtime performance of RGrVAE (left) and AGrVAE (right). Here we can see that the runtime increases polynomially for RGrVAE and linearly for AGrVAE both with and without a change of basis matrix. AGrVAE is an order of magnitude faster than RGrVAE per batch, which is mostly due to the methods used by AGrVAE being commonplace in deep learning and consequently have highly optimised implementations in the underlying frameworks. It should be possible to significantly improve the runtime speed of RGrVAE with efficient kernels and fused

operations. Note that the change of basis matrix has very little impact on the runtime performance.

4.4 How do we measure LDR

Given the definition of linear disentanglement and the fact that we can learn these representations through models such as ForwardVAE, it would be useful to quantify the degree to which representations are linearly disentangled. In this section we will propose metrics which aim to measure linear disentanglement and examine their drawbacks and merits. This section will aim to answer the second of our questions posed at the beginning of the chapter, “How do we measure LDRs?”.

Independence Score The first of our proposed metrics is the *Independence Score*. This score is based on part 3 of definition 3.7, which states:

3. There is a decomposition $\mathcal{Z} = \mathcal{Z}_1 \times \dots \times \mathcal{Z}_s$ such that \mathcal{Z}_i is fixed by the action of all $G_j, j \neq i$ and affected only by G_i .

This says that each of the subgroups G_i must act on an independent latent subspace \mathcal{Z}_i . We define the independence score explicitly to measure the extent to which a representation satisfies this part of the definition. It measures explicitly how independent latent subspaces which are affected by actions of G_i are from all the other subgroups, i.e. do they share latent dimensions. For actions of $G = G_1 \times \dots \times G_s$ on the latent code z and the latent code after applying action a , denoted z_a , we define the score as:

$$\text{Independence Score} = 1 - \mathbb{E}_z \frac{2}{s(s-1)} \sum_{i,j \neq i} \max_{a \in G_i, b \in G_j} \left(\left| \frac{z - z_a}{\|z - z_a\|_2} \cdot \frac{z - z_b}{\|z - z_b\|_2} \right| \right) \quad (4.7)$$

We can break down individual parts of the definition to provide insight,

- $z - z_a$: Taking this difference means that for actions which only interact with a few latent dimensions, all dimensions other than these in the difference will be 0. This means the dot product value depends only on the dimensions that have been changed by the actions, and isn't dependent on the number of latent dimensions.
- $\frac{z - z_a}{\|z - z_a\|_2}$: Unit vector in the direction of the action at particular latent z

- $\left| \frac{z-z_a}{\|z-z_a\|_2} \cdot \frac{z-z_b}{\|z-z_b\|_2} \right|$: Projection of ‘direction’ of action a onto direction of action b . This will be 1 if they are the same action and 0 if they are perpendicular.
- $\max_{a \in G_i, b \in G_j}(\dots)$: We are interested in the worse case. If most of the actions in a subgroup are independent of actions in another subgroup but 1 action is shared between them then we shouldn’t consider the representation independent at that z .
- $\frac{2}{s(s-1)} \sum(\dots)$: Average by the number of possible pairs (G_i, G_j) .
- $\mathbb{E}_z[\dots]$: We care about the expected independence, not necessarily the independence at any particular part of the latent space.
- $1 - (\dots)$: We would like independent representations to score 1 and dependent ones to score 0.

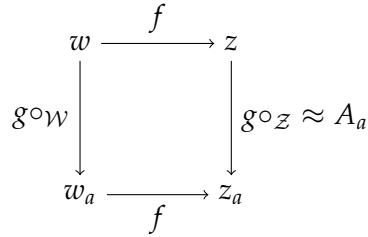
In general we will measure the independence score by optimising a set $\{A_i\}$ of predefined (but learnable) representations for a_i . This will allow us to compute metrics such as symmetry approximations (how well do the learnt representations match the known ones) for models which do not have internal representations, i.e. non-symmetry based models. Furthermore, this will allow us to estimate the independence with respect to known symmetries, by restricting A_i to have particular structure. For example, for symmetries C_N , we will often restrict them to be cyclic by learning only a cyclic rotation angle, rather than a generic $SO(2)/GL(2)$ matrix. Since this is our general approach, we will denote the result ‘Independence’, ‘True Independence’ or ‘G-Independence’ (where G would be C_N when the acting symmetries are cyclic). When estimating directly from the dataset (without learning representations A_i), we will denote the result as ‘Direct Independence’. Note that often a dataset does not have transitions from a given state for every action, usually only a subset of such actions. In this case the Independence as written in Eqn. 4.7 is hard to estimate since it relies on comparing local actions. We can achieve an estimate of the Direct Independence by averaging over all actions we have observations for, which will offer us decent estimates for highly independent spaces, but degrades as the space is less independent. As such, we will use the G-Independence since it allows us to estimate the independence based on all actions for all observations.

Relative Latent Error Of the 3 parts of definition 3.7, part 1 is concerned with existence, which is assumed. Part 3 we can measure by the independence score. Part 2² says that the action on the latent space should be equivariant with that on the world space.

²Note that $h \circ b$ denotes a function composition here, and not a group action as it will continue to be elsewhere

2. The composition $f = h \circ b : \mathcal{W} \rightarrow \mathcal{Z}$ is equivariant with respect to the group actions on \mathcal{W} and \mathcal{Z} . i.e. $g \circ_{\mathcal{Z}} f(w) = f(g \circ_{\mathcal{W}} w) \forall w \in \mathcal{W}, g \in G$.

ForwardVAE encourages this by optimising for it, the actions will be equivariant if the latent space is structured cyclically (on FlatLand), which is the case of low latent reconstruction error.



Minimising $\|A_a z - z_a\|$ effectively minimises $\|A_a z - f(g \circ_{\mathcal{W}} w)\|$ since $A_a z$ approximates $g \circ_{\mathcal{Z}} f(w)$ and $z_a = f(g \circ_{\mathcal{W}} w)$.

As such we can estimate the extent to which part 2 holds by the error term. If our learnt action has the same effect as the world action, they are equivariant. Since no model is going to have an error of 0 and different models will encode into different spaces, we will consider the relative latent error - the latent error divided by the expected distance between latents.

$$\text{Relative Latent Error} = \mathbb{E}_{z,a} \frac{(z_a - \hat{z}_a)^2}{\frac{1}{N} \sum_{z,a} (z - z_a)} \quad (4.8)$$

SVD Overlap For an alternate measure to the independence, we can consider the singular values of the spaces of latent actions, i.e. the spaces spanned by vectors $z_a - z$ for each a . The number of high magnitude singular values tells us the number of active dimensions for that action. This allows us to find the degree of actions as well as determine the independence by comparing the singular vectors for different actions. To relate back to part 3 of the definition 3.7, we can consider the mean overlap between two action planes (assuming for demonstration that the actions have two dimensional irreducible representations).

$$\text{Action Vectors : } V_a = \{z - z_a \mid \forall z\}$$

$$\text{Singular Vectors : } \{A_{a,i}\} = \text{SVD}(V_a)$$

$$\text{Mean Overlap : } \frac{s(s-1)}{2} \sum_{a \in G_i, b \in G_j} \frac{1}{4} \sum_{k,l \in \{0,1\}} A_{i,k} \circ A_{j,l}$$

In its current form, the mean overlap is dependent on knowing the dimensionality of the irreducible representations of the symmetries acting on the data (we chose the

range of k, l based on this). For our work we will know the form of the irreducibles, however there may be situations when this is not possible. In this case we could simply consider all singular vectors ($k, l \in \{0, \dots, N\}$) and weight them by their singular values. This method would mean that vectors with low singular values would not contribute significantly to the overlap, and it would largely be determined by the significant (by singular value) vectors.

Factor Leakage We introduce the Factor Leakage as an extension to the Mutual Information Gap metric (Chapter 3, Section 3.2.4) to allow awareness of the number of dimensions the latent factor is encoded across. The MIG metric measures the difference in information between the first and second most informative latent dimensions for each factor. The Factor Leakage instead considers the whole information curve, so as to better capture factors which are encoded across latent subspaces rather than single dimensions. This is important for considering linear disentangled representations. We have i 'th latent dimension z_i , the k 'th generative factor v_k and the mutual information between the two $I(z_i, v_k)$. We assume that for each v_k , the order of latent dimensions z_i has been sorted by the mutual information. We can consider a number of variants,

$$\text{FL Mean}(z, v, d) = \frac{1}{N_v} \sum_{k=1}^{N_v} \sum_{i=d}^{N_z} I(z_i, v_k) \quad (4.9)$$

$$\text{FL Norm Mean}(z, v, d) = \frac{1}{N_v} \sum_{k=1}^{N_v} \sum_{i=d}^{N_z} \frac{I(z_i, v_k)}{\max_t I(z_t, v_k)} \quad (4.10)$$

$$FL(z, v) = \frac{1}{K} \sum_{d=1}^{N_z} \text{FL Norm Mean}(z, v, d) \quad (4.11)$$

In general, we will use the final variant (4.11) if not specified otherwise, and d is set to 2 when the Mean metrics are used individually. The parameter d allows us to ignore the information between the code and the d most informative latents. For classically disentangled representations, we would expect there to be 1 highly informative latent for each factor. For linearly disentangled representations, we can see more than 1 highly informative latent dimension. The parameter d then decides the sensitivity of the metric to these cases. For instance, for a classically disentangled representation, we would expect FL Mean to be large for $d = 1$ and small for $d = 2$, whereas, for a linearly disentangled representation with cyclic symmetry, we would expect it to be large for $d = 2$ and small for $d = 3$.

Comparison Note that the symmetry based metrics (i.e. not Factor Leakage) all require labelled observations and the ability to sample based on actions. This may seem overly restrictive compared to some standard disentanglement metrics which

can be evaluated without generative factor labels, however since linear disentanglement is fundamentally tied to the symmetry decomposition we believe that supervision can be the only way to measure it.

Since the independence and latent error metrics speak to different parts of the linear disentanglement definition, they are naturally complementary. The SVD metric however responds to similar structures as the independence. Whilst the SVD and independence scores can effectively serve the same purpose, the major decider is complexity. The independence is relatively simple to compute and does not require singular value decompositions or other analysis techniques. For this reason, we will mostly be considering the independence score when comparing models, however the SVD metric can be used for additional analysis. For example, it may be useful to consider the singular values when you can sample actions from a (component) group but do not know the dimensionality of its irreducibles. In this case, the singular values might allow us insight into the dimensionality.

The Factor Leakage, as an extension of the Mutual Information Gap is not a symmetry based metric. Whilst the Factor Leakage will respond to linear disentanglement differently to standard disentanglement, it may also be degenerate. A model which is not linearly disentangled may have the same Factor Leakage score to one that is. Given additional scores from symmetry based metrics (like independence), we may be able to separate these models, and similarly, the Factor Leakage may be able to distinguish models which have similar independences. This however may also be the case for the latent error metric. Similar to the SVD metric, the Factor Leakage will be most useful as an analysis tool and as a means to further justify if a model is linearly disentangled or not.

Empirical Comparison We can briefly compare the different metrics on FlatLand to see how they relate empirically for a number of baseline models (symmetry based or otherwise). In Fig. 4.10a we report the Pearson correlations between the symmetry metrics. Unsurprisingly, the Independence and SVD overlap correlate fairly (negatively) strongly, as they both measure the extent component groups act on independent subspaces. The Factor Leakage is also (negatively) correlated with the Independence, however the relative latent error is much less so. Considering the SVD, FL and Relative error metrics, we can see that the SVD overlap correlates lightly with the others whilst the latter two correlate very weakly.

The Pearson correlation is a measure of linear correlation, whereas the Kendall correlation is a rank correlation - are the resulting value orderings the same between two variables. In Fig. 4.10b we see similar results to the Pearson correlation. The Independence results in fairly similar rankings (albeit reversed) to all metrics barring the relative latent error. The Kendall correlations between the SVD, FL and relative

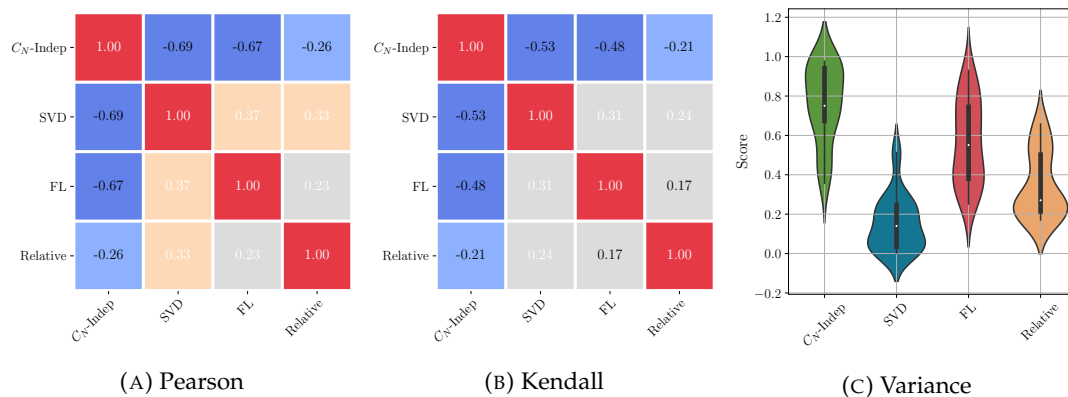


FIGURE 4.10: Comparison of Symmetry Metrics

error metrics are reduced compared to the Pearson correlations, showing that they measure fairly distinct representational properties.

Finally we show the general distributions of the metrics in Fig. 4.10c. We can see that the independence is weighted towards higher values, which is unsurprising since we are evaluating disentangling models (and symmetry based models, which will have high independence, as we will see). The SVD is weighted heavily towards low overlaps, for the same reasons. The Factor Leakage shows a much broader distribution, which is likely due to the range of different models encoding the data in different ways. Finally, the relative latent error is mostly weighted low with moderate density at higher errors.

In general, the G -Independence will be a good all rounder metric, since it should capture some symmetry structure whilst also reflecting the independence (in the sense of Eqn. 4.7) of the representation. As it correlates with the SVD and FL metrics to a moderate degree, it may be sufficient to consider only the Independence out of the three. The relative latent error does not correlate significantly with any of the other metrics, so we will try to consider it alongside the Independence when possible.

4.5 Which Spaces Admit LDR

In the previous sections we have defined ForwardVAE and proposed GroupVAE as models to learn linear disentangled representations. In this section we will show that these models can learn such representations and explore if standard VAE baselines also have this capability. We shall use the FlatLand problem due to its simplicity and few viable symmetries. This will allow us to search candidate spaces empirically for a number of the possible symmetries. This ability will allow us to explore the third question posed in the chapter introduction, “Are LDRs learnt by standard VAE models?”.

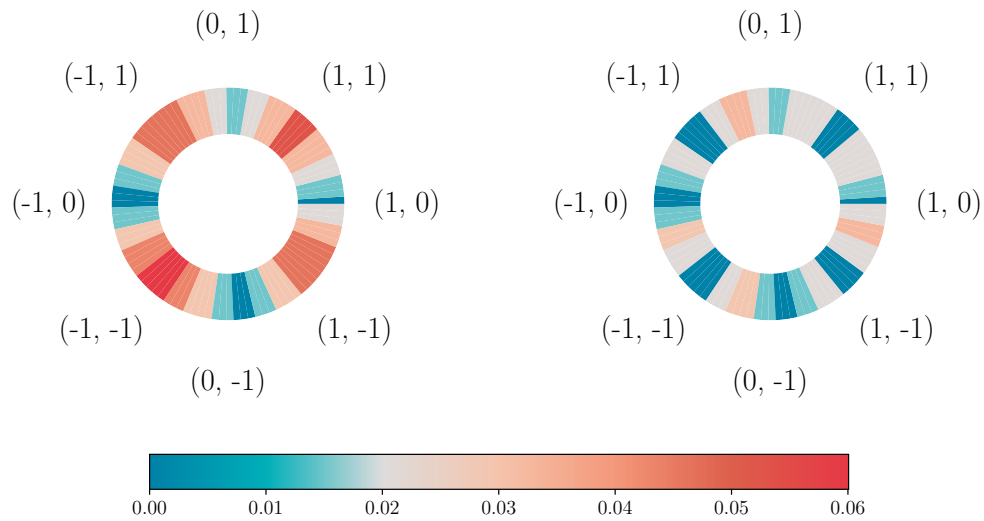


FIGURE 4.11: Minimum MSE for translations in each direction against axis aligned translations (left) and axis aligned as well as diagonal translations (right). We can see the minimum MSE decreases dramatically once we consider additional translation vectors. Step Size: 5

Possible Actions FlatLand is a grid world of size 64 with 32 possible agent states on each Cartesian axis. Even restricting to solely cyclic symmetries of order N acting on the data, there are many possible realisations of their actions since they can act at any angle to the Cartesian. For example, a perfectly valid symmetry structure could have actions which translate the agent by orthogonal vectors $(2, 1)$ and $(-1, 2)$. The first acts at approximately 26 deg from the Cartesian x-axis, however any such angle is equally valid, as long as the two vectors are orthogonal. Since we cannot consider all possible actions, we will restrict to actions at 0 or 45 degrees, i.e. corresponding to translation vectors $(1, 0)$ and $(0, 1)$ or $(1, 1)$ and $(-1, 1)$. We will call these two sets of actions the Normal and Diagonal bases respectively. Whilst there are a large family of other possible building block translations (e.g. $(2, 1)$, $(1, 3)$, etc.), we believe they are suitably close to one of our listed translations. Indeed, in Fig. 4.11, we show that the expected MSE against translations in different directions is much lower once we consider these additional vectors.

Method We would like to empirically search for linear disentangled representations in a latent space with respect to group structure $G = C_{13} \times C_{13}$, which act according to a pair of orthogonal translation vectors as we previously defined.

We first define a rotation matrix A_i for each action and their inverses, i.e. $\{A_1, A_2, A_3, A_4\}$ since we have two actions and two inverses. We then observe transitions (x, x_a) with latent codes (z, z_a) and optimise the matrix A_i corresponding to action a in order to best approximate the post action latent, $\mathcal{L} = \|z_a - A_i z\|$. Once converged, we can measure the independence of the representations A_i by replacing

z_a and z_b of Equation 4.7 with $A_a z$ and $A_b z$ respectively. We can also compute post action observation errors, post action latent reconstruction error and symmetry reconstruction errors. The latter being defined as the mean squared error between the expected rotation angle of 0.926° and that found in rotation matrices A_j .

The matrices are optimised for 30 epochs using a mean squared error loss, batch sizes of 1024, and the Adam optimiser with learning rate of 0.1. All results will be reported on a 10% validation split of the dataset.

Pretrained VAEs For this experiment we require a number of pretrained VAE models, the architecture for the backbone VAE in each case was the same for most instances. This architecture is defined in appendix section B.3. There are two exceptions to this, the first being the TC model which uses the architecture defined in section B.4. The second is the MMD-VAE which uses the architecture defined in section B.5. All VAEs were trained as outlined in Appendix Section C.1, with the only variation being the training length, where all models for this section were trained for 200 epochs.

Results We first consider reconstruction scores and the rotational symmetries discovered (or not so). In Fig. 4.12 we provide reconstruction errors for (relative) post action latents (Relative Latent) and observations (Recon), alongside the independence score and symmetry reconstruction (MSE between rotation matrix angle α of internal representation and known irreducible). We first note that ForwardVAE and RGrVAE in general achieve the best scores in each metric for the normal basis (blue) and show extremely low variance. Since they are designed to learn linear disentangled representations, this should be expected. We can also see that this is not the case for the diagonal axes, since they both encourage actions to be aligned to the Cartesian.

Of the baseline (non symmetry based) VAEs, we first note that the TcBetaVAE independence scores are very high under the Cartesian axes whilst the DIP-I and β -VAE score similar to ForwardVAE on all metrics bar latent MSE.

TcBetaVAE is known to learn strongly disentangled representations in the classical sense. Models that do this well should learn very independent representations. For this particular experiment, we are optimising 2D rotation matrices (with a change of basis) to best approximate actions. For perfect classical disentanglement, there is one active dimension per factor, and the remaining dimensions are inactive - the output does not depend on them. As such, approximating them with 2D rotation matrices would result in the representation acting on the active dimension and one inactive dimension - otherwise the output would vary in two factors, leading to worse action approximations. Since learning of each rotation matrix is independent, the inactive dimension chosen by each rotation matrix is completely arbitrary. This could lead to

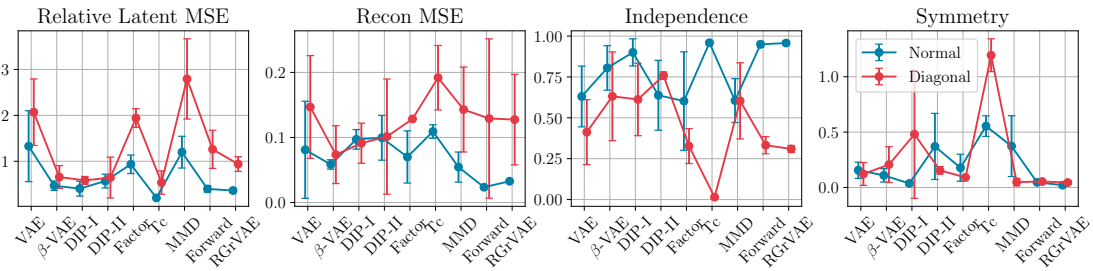


FIGURE 4.12: Metrics when attempting to find linear disentangled representations in VAE latent spaces. We look for actions which translate on the normal (Cartesian) and diagonal (Cartesian rotated 45 degrees) axes. In all subfigures, blue denotes results under the normal axes and red those under the diagonal axes. Solid lines indicate mean performance with 1 standard deviation given as error bars.

reduced independence scores if two representations by chance selected the same inactive dimension. The fact that TcBetaVAE scores consistently high independence scores suggests that this doesn't happen, despite there being only two inactive dimensions (and two active ones). This then suggests that TcBetaVAE encodes each factor *almost* completely in a single dimension (based on latent traversals) but they still depend a small amount on a second dimension, which the learnt rotation matrices select.

Since they perform relatively similarly, we shall discuss BetaVAE and DIP-I together. The first thing to note is that the variance (particularly on the independence) is much higher for baselines compared to symmetry based models. This suggests that if baselines learn linear disentangled representations, they are not consistent in doing so. Consider the reconstruction MSE, which is a good overall measure of action estimation quality. Compared to the symmetry models, none of the baseline models perform well at all. We will explore in a later section if individual instances of baseline models actually learn linear disentangled representations or not.

Returning to an overall comparison of baselines to symmetry models, in Fig. 4.13a we plot the mean component overlap (SVD metric). We see a similar picture to the previous results, ForwardVAE/GroupVAE achieves very low overlap (very independent) whilst the baselines show varying larger overlaps. DIP-I is the only baseline model that shows similar overlap to the symmetry based models. Note that whilst the previous experiment suggested that TcBetaVAE may be potentially linearly disentangled, we can see from the overlap that it is almost certainly not. This is reinforced by Fig. 4.13b which plots the expected magnitude of the singular values, and shows that TcBetaVAE has one large value and three small values. This says that it relies strongly on single dimensions to encode actions, and is not linearly disentangled with respect to cyclic groups, which would show two dimensions per action. This can be seen for the ForwardVAE and RGrVAE models, with a large gap between the magnitude of the first and second two singular values. Fig. 4.13b specifically

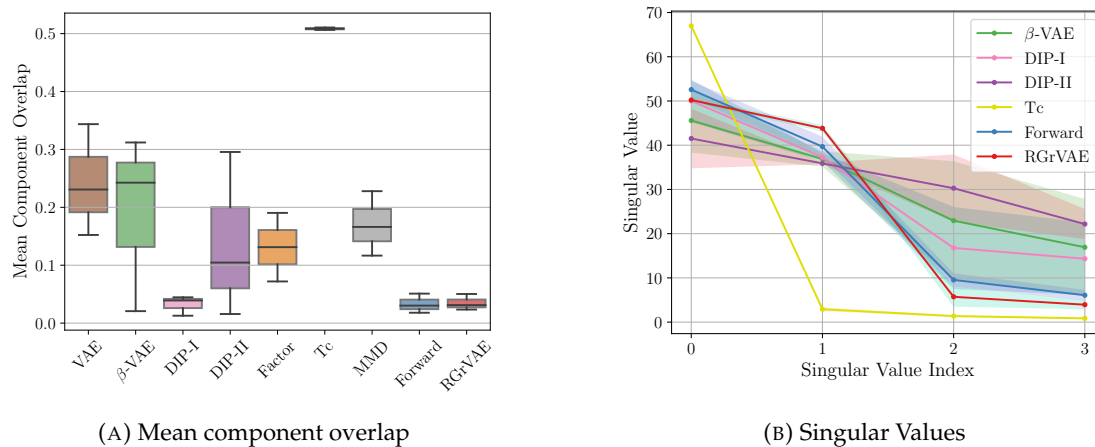


FIGURE 4.13: Singular value analysis for baselines and ForwardVAE on FlatLand. (B) Solid lines indicate mean performance with 1 standard deviation given as the shaded area.

highlights the baselines that are potentially linearly disentangled. We can see that DIP-I also shows a relatively large gap between the first and second pair of singular values. By comparison DIP-II and β -VAE show relatively linear decrease in magnitude.

The two methods we explored in this paragraph have highlighted DIP-I and β -VAE as showing possible linear disentanglement, with DIP-I looking more promising. From considering the variance in most of the baselines, we can conclude that, in general, baseline VAEs do not learn linear disentangled representations on FlatLand, at least with respect to cyclic symmetries. This does not rule out the possibility of them learning such representations occasionally, by luck, however.

Do baselines show Linear Disentanglement? Compared to ForwardVAE and RGrVAE, baselines show lower independence and reconstruction scores across both bases. This suggests that baselines do not consistently learn linear disentangled representations, and that there is no loss pressure towards them. However, we can see some evidence that they are occasionally found in the baselines despite this. In particular, the BetaVAE and DIP-I models show high independence in the standard basis alongside relatively low reconstruction errors. It is easier to see this in Fig. 4.14 (left) which scatters the independence and (symmetry) reconstruction scores against each other for all the data points, coloured by their model type. The cluster in the lower right (red circle), contains all the ForwardVAE and RGrVAE runs, which are linearly disentangled representations. There are a number of DIP-VAE runs in the cluster which are likely linearly disentangled. There is a second, looser, cluster with lower independence and worse reconstructions which seem to have some degree of linear disentanglement, perhaps they are not linear for all factors, or it is a low quality linearly disentangled representation.

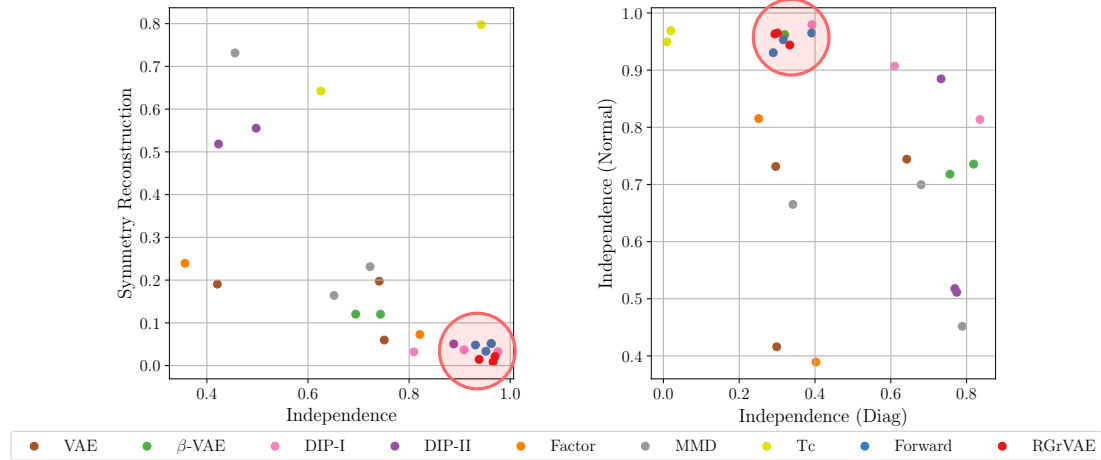


FIGURE 4.14: Scatter of individual runs of baseline and symmetry based VAEs for independence and symmetry metrics. The cluster of linear disentangled models is highlighted. (left) Scatter of independence (x), symmetry recon (y). (right) Scatter of independence (ni) (y), independence (di) (x)

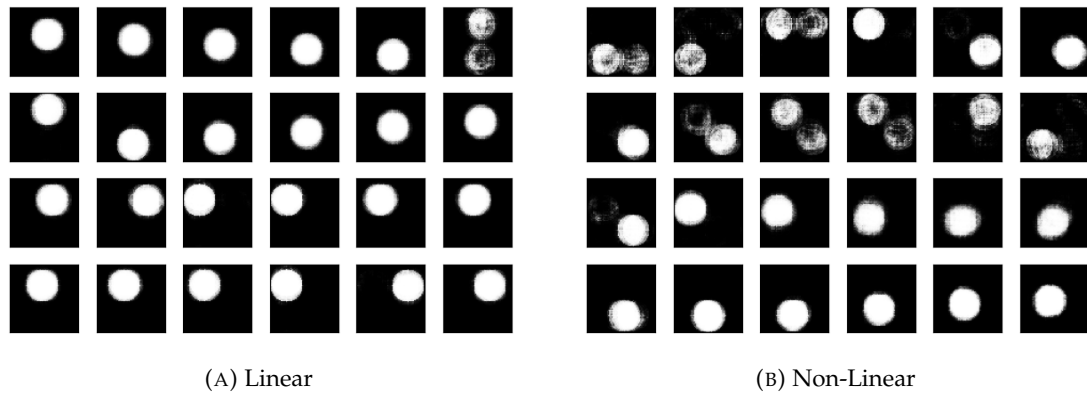


FIGURE 4.15: Learnt action orbits on two different Dip-VAE latent spaces. We can see that sometimes BetaVAE learns a linear representation, and other times it does not. Note that even in the non-linear case, the lower two actions appear to be somewhat linear.

To further demonstrate the ability of some baselines to learn linear disentangled representations, we provide examples from a DIP-I. On the left of Fig. 4.15 we see action traversals on a DIP-I run which happened to be linearly disentangled and on the right, a different (non linear disentangled) run. Notice that in the latter case, the lower two traversals seem like they are somewhat disentangled, however the top two are not. This explains why in the previous experiments we see baselines such as β -VAE and DIP-I sometimes achieving scores similar to the symmetry based models - they learn linearly disentangled representations some of the time. Even in the cases where they do not, they may have partial disentanglement as seen in the figure. Note however that we cannot confidently extend this to models other than DIP-I, DIP-II (possibly) and β -VAE, since these are the only models that showed similar scores to the symmetry based models.

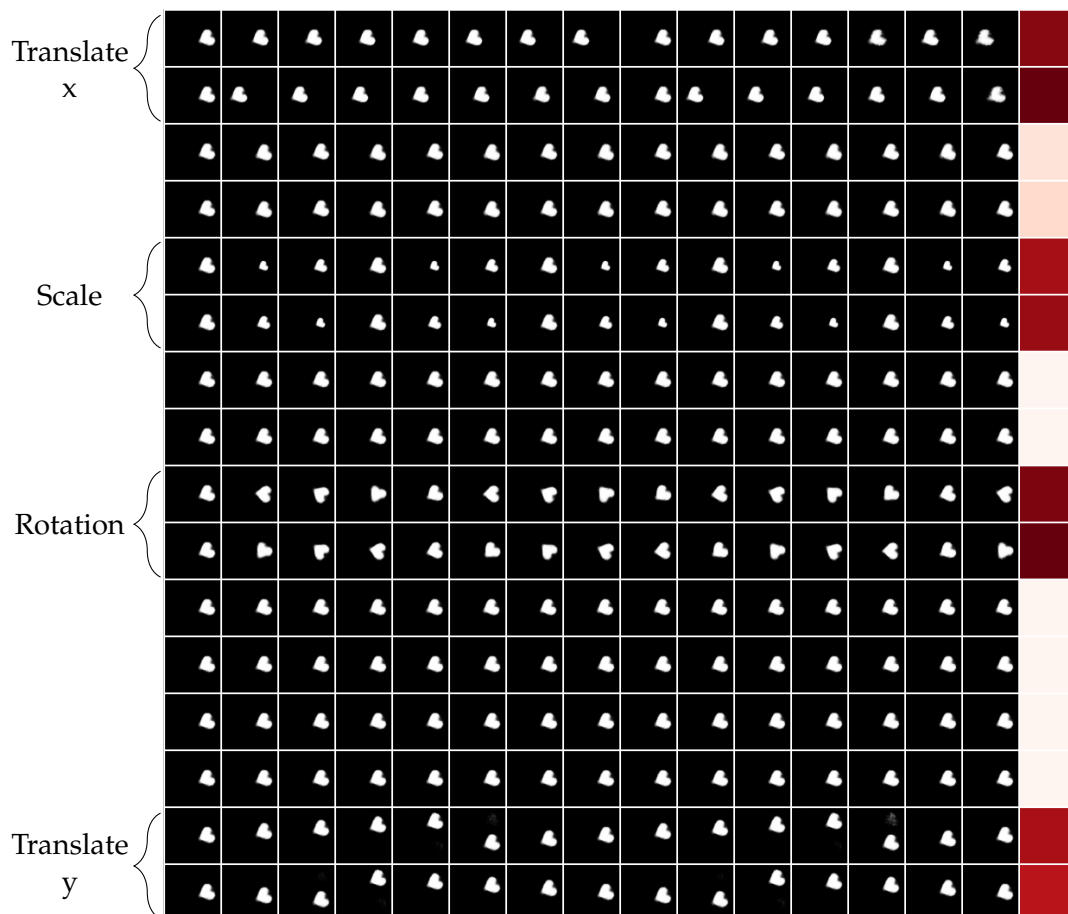


FIGURE 4.16: Action orbit traversals for all available actions of RGrVAE trained on dSprites and a heat-map over the actions selected over the dataset. We sampled the true actions from the dataset based on the following symmetry groups: Scale: C_3 , Rotation: C_{10} , Translation: C_8 . Thus the group acting on the data is: $G = C_3 \times C_{10} \times C_8 \times C_8$.

Extending to dSprites FlatLand being a simple, lightweight dataset allowed us to quickly train and evaluate lots of models, whilst searching for linear disentanglement. We will now perform reduced versions of these experiments on the more computationally demanding dSprites dataset, to see if the results match those on FlatLand. The training parameters for the dSprites RGrVAE models are given in Appendix Section C.2.

We present RGrVAE action traversals in Fig. 4.16, with a heat-map to highlight active dimensions. Without yet evaluating the representation through metrics, we can already see that this example is linearly disentangled. We can see high quality representations of all actions, although the rotation has learnt a lower order symmetry than is present in the data. This is likely due to internal rotational symmetry present in two of the three dSprites shapes making this symmetry ($\approx C_4$) a good representation for the action on most of the observations. It is possible that with additional training the proper rotation symmetry could be learnt.

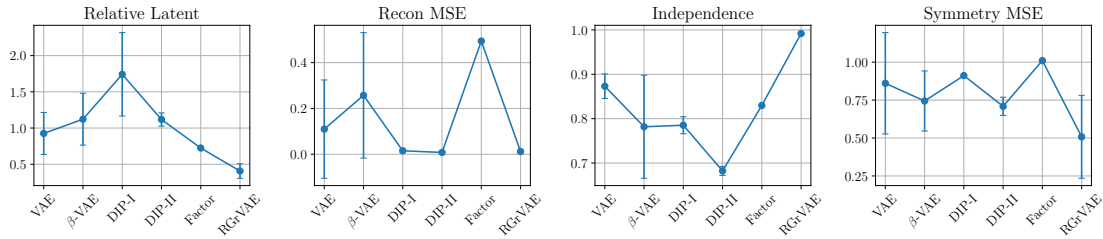


FIGURE 4.17: Reconstruction metrics for RGrVAE and baselines on dSprites. Solid lines indicate mean performance with 1 standard deviation given as error bars.

Since we cannot easily access diagonal actions, and the results of testing diagonal actions in Fig. 4.12 are generally worse than those on the standard bases, we will only evaluate symmetry metrics on the Cartesian for dSprites. Fig. 4.17 provides reconstruction, relative latent and symmetry MSEs. We find that, similar to the experiment on FlatLand, the reconstruction MSE of DIP-VAE (in this case both I and II) are similar to that of the symmetry based model. However, unlike in the FlatLand experiment, DIP-VAE performs significantly worse on all the other experiments, including the symmetry reconstruction, suggesting that this time the DIP-VAE did not learn a linear disentangled representation in any of the test runs. The same follows for β -VAE. Note that the reconstruction MSEs are worse across the board, even for RGrVAE, since dSprites is a significantly more complex task with more factors and different shapes to model. Despite this, RGrVAE continues to show very high independence, indicative of continuing to learn a linear disentangled representation, albeit with diminished quality due to task complexity. We believe that additional training time could alleviate this.

In Fig. 4.18a we repeat the singular value analysis of Fig. 4.13a, where we see a similar picture. RGrVAE shows much lower mean overlap than other models, although due to lower quality representations, the overlap is comparatively larger than on FlatLand. Fig. 4.18b then plots the (size sorted) singular value magnitudes, and shows that RGrVAE has two high value singular values with the rest significantly lower. Baseline models show large initial singular values with a gradual drop off or a steep initial drop off before levelling out. This suggests that none of the baselines have learnt linear disentangled representations in this test. This doesn't disprove the possibility, rather just that we did not observe them.

In general the dSprites experiments show the same results as the FlatLand experiments. Symmetry based models consistently learn linear disentangled representations, whilst baseline VAEs do not, but may by chance learn them on occasion. This may be rarer in more complex datasets such as dSprites, although we cannot say this definitively.

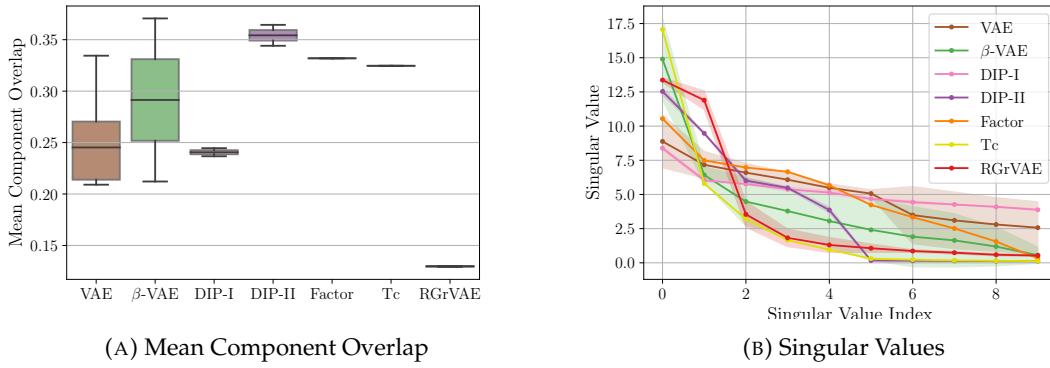


FIGURE 4.18: Singular value analysis for baselines and RGrVAE on dSprites. (B) Solid lines indicate mean performance with 1 standard deviation given as the shaded area.

4.6 Evaluating GroupVAE

Having defined models which have the ability to learn linear disentangled representations, and looked at whether standard VAE baselines learn the same representations, we now want to further explore the symmetry based models, and in particular our proposed GroupVAE. We will begin by determining which of the two GroupVAE variants is best for learning linear disentangled representations. We will then look at other aspects of GroupVAE, such as hyperparameters, to determine how they impact the learning. By exploring GroupVAE and its properties, we aim to answer the final question posed in the chapter introduction, “Can we learn LDRs without action labels”.

Metrics Throughout this section we will be comparing models using a number of reconstruction metrics, other than the symmetry based metrics outlined in Section 4.4. The first is the observation reconstruction metric (or ‘X’) which is simply the mean squared error between the input image and the VAE reconstruction of this image. The second reconstruction metric is the latent reconstruction. This is the mean squared error between the predicted post action latent and the true post action latent, this can also be given as the relative latent reconstruction error as described in Section 4.4. The final reconstruction metric is the symmetry reconstruction, which is given as the mean squared error between the rotation angle of the learnt internal representations and the rotation angle of the true irreducibles (rotation matrices for cyclic symmetries). All metrics are reported averaged over a validation set consisting of 10% of the data held out from training and given with errors over 5 trained model instances.

Attention or Reinforcement In Section 4.3, we gave a high level comparison of RGrVAE and AGrVAE. We now perform preliminary experiments on FlatLand for both variants to determine their effectiveness. For this, we train 5 repeats for both

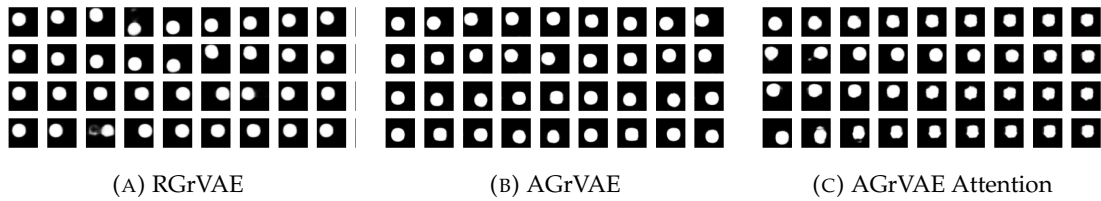


FIGURE 4.19: Action traversals for RGrVAE and AGrVAE. AGrVAE has both traversal of the internal representations directly and based on the attention mask generated once for each action.

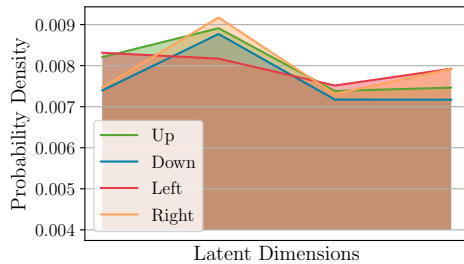


FIGURE 4.20: Distribution of attention in expectation over the latent space.

	Recon MSE	Relative Z MSE
RGrVAE	$0.022_{\pm 0.009}$	$0.150_{\pm 0.001}$
AGrVAE	$0.309_{\pm 0.113}$	$0.202_{\pm 0.079}$

TABLE 4.1: Comparison of RGrVAE and AGrVAE reconstruction metrics.

AGrVAE and RGrVAE on the FlatLand dataset. Both models use the backbone defined in appendix section B.3 and RGrVAE uses the action estimation network defined in appendix section B.6. Both GroupVAE variants used 4 internal representations, one for each observable action. The models were trained for 200 epochs, with the adam optimiser, learning rate 0.0001, batch size 128 and mean squared error loss. All results are reported on an validation split consisting of 10% of the total dataset.

We begin with Fig. 4.19, showing action traversals on FlatLand. RGrVAE shows very distinct actions with correct warping as the agent reaches the edges of the frame. AGrVAE however does not show any sign of the actions in the traversals - since the true actions are a combination of the internal representations. For AGrVAE, is not guaranteed that these combinations are constant for each action however, they may be different at different points in the latent space. In Fig. 4.19c, we use an attention mask generated once for each action to traverse. Since we still do not see good quality actions, it is likely that the correct attention for each action varies over the latent space - i.e. it has not learnt a linear disentangled representation.

To confirm this, we can look at the expected attention distribution for each action in expectation over the latent space, as shown in Fig. 4.20. There is no obvious structure in these distributions, they are almost uniform across all dimensions, which tells us that the action representations do vary across the latent space. We can be sure of this and that it is not just poor learning since the post action reconstructions (Table 4.1) show similar error to RGrVAE.

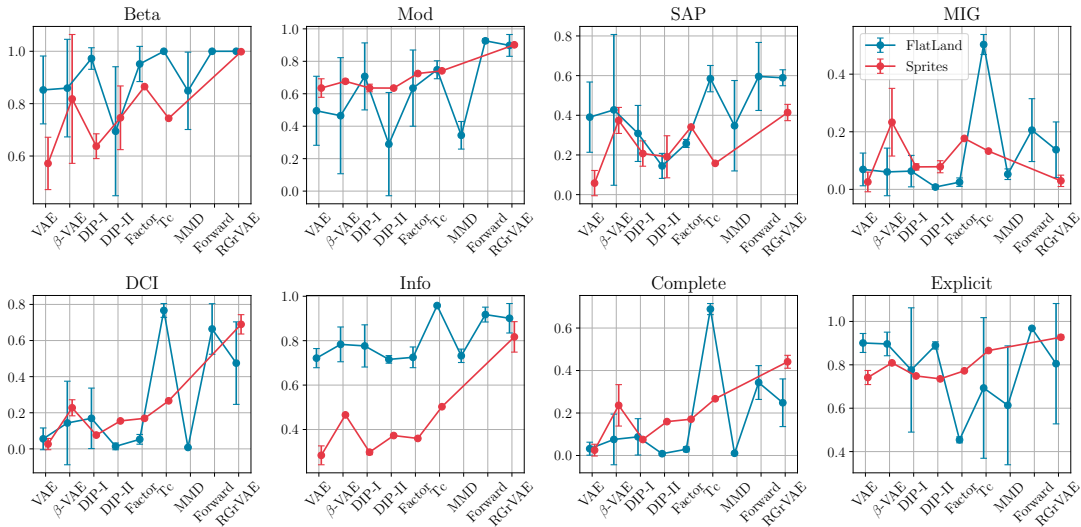


FIGURE 4.21: Disentanglement metrics for FlatLand (blue) and dSprites (red). Solid lines indicate mean performance with 1 standard deviation given as error bars.

Since AGrVAE did not learn linear disentangled representations effectively, we will only be considering RGrVAE for further experiments. Whilst we may be able to ‘fix’ AGrVAE with additional losses and structural changes, we prefer to stick to the conceptually simple RGrVAE, rather than tailor AGrVAE for each problem. Since the main disadvantage of RGrVAE is that it has slower convergence/runtime, this does not limit its scope, only the hardware requirements of future experiments.

Symmetry based models under disentanglement metrics Section 4.5 discussed how we can learn linear disentanglement metrics and searched for evidence of them in standard VAE baselines. We now would like to compare the symmetry based models to standard VAEs under a set of classical disentanglement metrics. We would also like to understand what role our independence, SVD and factor leakage metrics play.

For these experiments, all models were trained as outlined in Appendix Section C. We then evaluated disentanglement metrics on the resulting representations. This provided us with a set of metric scores for each model, resulting in variance in the scores which we represent with 1 standard deviation error bars.

Fig. 4.21 provides classical metric scores on FlatLand and dSprites for the baseline VAEs and symmetry based models. Overall, we see the same relationship between symmetry based and classical models on both datasets. There is a notable exception where the MIG metric shows the symmetry based models having a (relatively) large gap compared to most (not Tc) of the classical models on FlatLand, but not on dSprites. We would expect the symmetry based models to have very low MIG since the first two most informative latents should relate to a single factor on both tested datasets. On FlatLand, the symmetry based models tended to have larger mutual

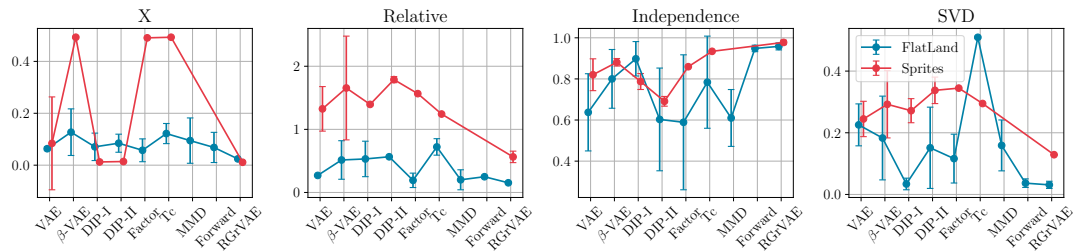


FIGURE 4.22: Symmetry Disentanglement metrics for FlatLand and dSprites. Solid lines indicate mean performance with 1 standard deviation given as error bars.

information scores than the classical models, with a small gap (relative to the mutual information) between the first two (sorted) latents and a very large gap for the next two. This may also be due to classical models (except Tc) learning low quality disentangled representations, which would explain the disparity between them and Tc, since the latter is known to nearly always learn well disentangled representations.

For the other metrics we see broadly similar relationships between symmetry based and classical models on both datasets. In general the symmetry based models get good metric scores on the BetaVAE metric, Modularity, DCI disentanglement, Informativeness and Completeness. The explicitness metric shows less distinction between symmetry and classical, although symmetry based models tend to get similar scores to the highest classical models. In most cases, Tc performs the best of the classical models, achieving higher metric scores than symmetry based models in a number of cases.

The linear disentangled representations appear to lend themselves to generally consistent and often distinct scores under the classical metrics, probably due to their highly structured nature, which we will see in Chapter 5. In particular, under the BetaVAE metric, symmetry based models consistently achieve 100% accuracy, a property that is only otherwise achieved by Tc.

Now considering symmetry based metrics (Fig. 4.22), we can see that the post action reconstruction errors (X, Relative) show symmetry based models performing best on both datasets. Note that in some cases on dSprites, we were unable to learn good estimates of the actions at all, in particular for BetaVAE, FactorVAE and Tc. The DIP-VAE models however were fairly amenable to learning linear approximations of the actions, as we can see from the low post action observation error. We see similar results for the Independence and SVD metrics. Symmetry based models are highly independent and show very little SVD overlap and in general the classical models are less independent and show more overlap. Interestingly, the DIP-VAE models do not show particularly strong independence or particularly low overlap, so despite occasionally seeing them learn linear disentangled representations in Section 4.5, we don't believe they have done so on dSprites in this instance.

For FlatLand, we again see the symmetry based models have lower post action reconstruction metrics and SVD overlap, with higher independence. Notice that the DIP-I model has very low SVD overlap, suggesting that these instances may have learnt somewhat linear disentangled representations, although the fairly poor post action observation and relative latent scores, suggest it would be a low quality representation. The only stand out result here is the SVD overlap of T_c , which is extremely large for this dataset. This is likely due to the SVD estimation learning 2D cyclic matrix representations, which are not a great fit for strongly classically disentangled models such as T_c . Since each factor will be encoded into a single ‘active’ dimension, each representation matrix will act on one ‘active’ dimension and one (basically random) ‘inactive’ dimension. If the inactive dimensions chosen are the same for multiple actions then the overlap will be high, despite the model being strongly disentangled.

4.6.1 Properties

Training Scheme All models in this section were trained according to the scheme set out in Appendix Section C.

Temporal Consistency One means to determine the quality of learnt action representations is to observe the errors introduced by composition. Over a number of steps we can randomly select an action, apply the corresponding internal representation and compare the decoded result to the true observation/latent after applying the same action in world space. The better the representation and action estimation, the less error will be introduced over time.

Fig. 4.23a presents the reconstruction (observation) error on FlatLand for RGrVAE and ForwardVAE. Despite RGrVAE being (action) unsupervised, the initial errors are comparable to those of ForwardVAE. As we increase the number of steps, RGrVAE tends to perform slightly worse, although is more consistent. We see a similar picture in the relative latent error (Fig. 4.23b), where RGrVAE initially performs well and gradually accumulates errors. Note that the errors accumulation is decreasing with respect to the observation (Fig. 4.23a), but are increasing linearly when considering relative latent error. Also note that RGrVAE has a steeper linear error compare to ForwardVAE, so it will become increasingly worse than ForwardVAE.

Symmetry based models are effective at modelling short term action sequences, as shown by the low errors in reconstruction metrics. However as the number of steps is increased, any error in the internal action estimations is bound to accumulate, which is evident from both of these figures. This might be particularly relevant for cyclic symmetries, since there is error in both the rotation angle and the determinant, which

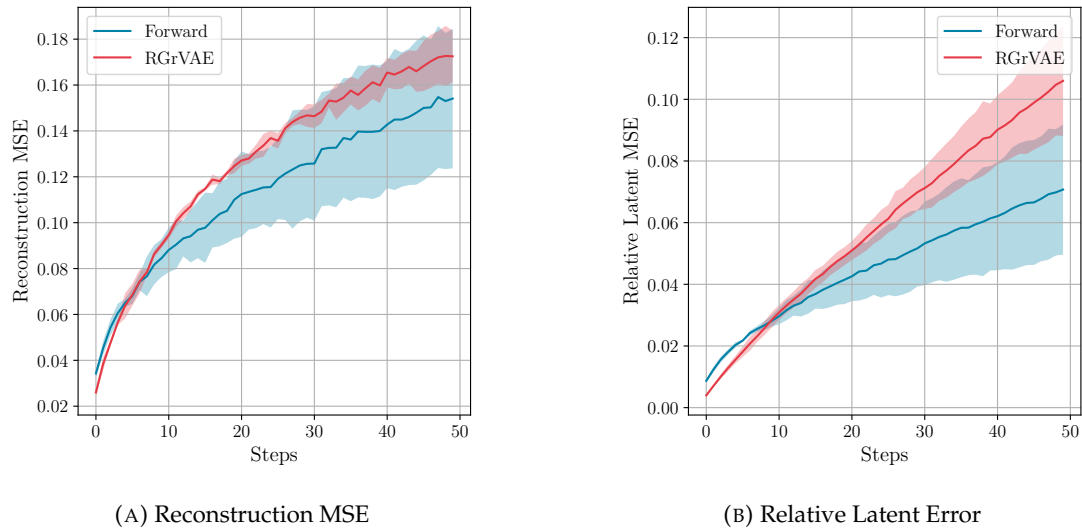


FIGURE 4.23: Comparing ForwardVAE and RGrVAE under temporal consistency. Solid lines indicate mean performance with 1 standard deviation given as the shaded area.

can lead to A^nz exploding or tending towards a constant in the limit. Nevertheless, for short term modelling and planning, symmetry based models may be a suitable method.

Long Action Sequences The previous paragraph considered the errors introduced as symmetry based models attempt to extrapolate over action composition. We will now consider learning by observing image transitions which are separated by application of more than a single action. To do this, we created a FlatLand dataset where transitions could consist of more than one action application, with the maximum number of actions defined by the ‘offset’ variable. To compensate for the increased number of actions that can be observed, we allowed RGrVAE to apply more than one action also. To do this, we first take a single action step in RGrVAE as usual, and decode the result. Using the decoded result and the post-transition image, we then use the RGrVAE action estimation network to infer another action to apply. We repeat this process for a fixed number of actions (10 for this experiment), and also include a identity internal representation in RGrVAE so that it can decide when to stop applying actions.

We can see from Fig. 4.24a that the policy continues to converge for a small number of steps, however the final independence shows a large variance. This is reflected in the post action observation reconstructions (Fig. 4.24a) which show gradually increasing error as the steps are increased. When taking a large number of steps, the reconstruction error is relatively low with low independence, showing that the model has learnt a good representation of the actions however it is not linearly disentangled with respect to $C_N \times C_N$.

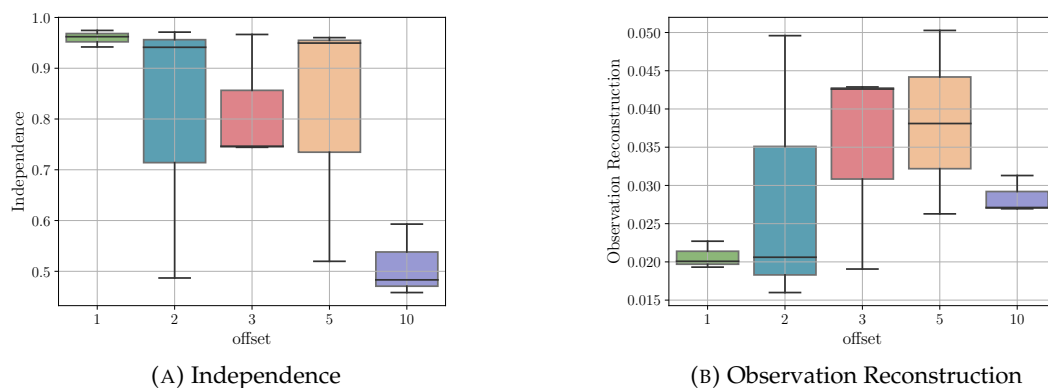


FIGURE 4.24: Long Action Sequences

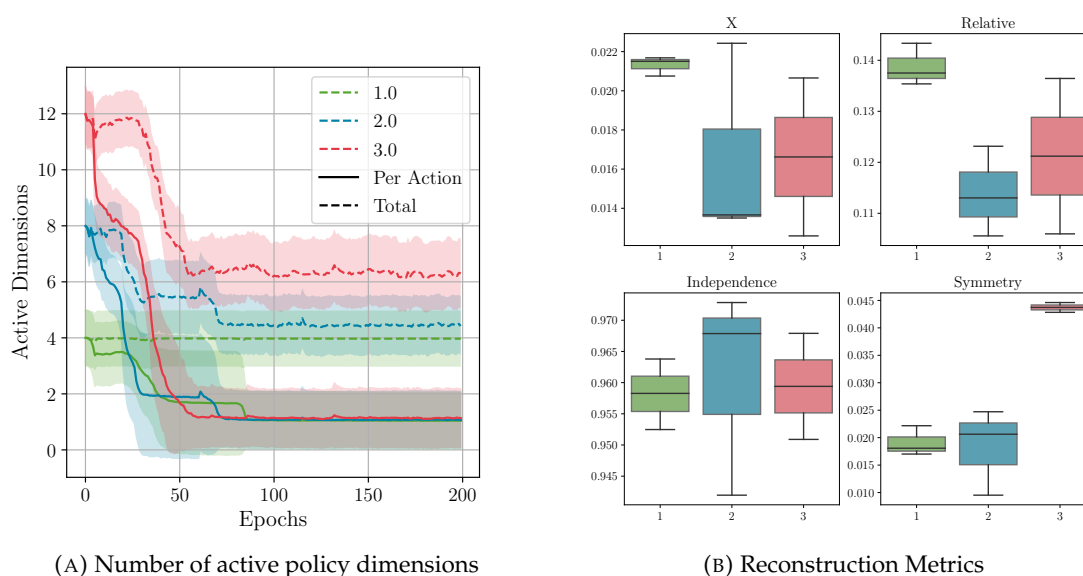


FIGURE 4.25: Over Representation when we have 1, 2 or 3 internal representations per true action. (A) 1 standard deviation is given as the shaded area.

Over Representation A problem unique to GroupVAE is that of choosing how many internal representations to define. If we define fewer internal representations than there are actions acting on the data, then we will never learn a good representation. However, if there are fewer actions than internal representations then we may learn multiple representations of the same action - possibly in different latent subspaces. Considering this, we should strive to define as many internal representations as actions, however since we are unlikely to know this number, it is preferable to ‘over represent’ each action by defining more internal representations. For this experiment, we simply train RGrVAE instances as outlined in Appendix Section C but multiply the number of internal representations by 1, 2 or 3.

To see how RGrVAE responds to over representation, we measure the number of active policy dimensions per action, estimated by $N \approx e^h$ where h is the mean entropy of the policy distribution per action. In Fig. 4.25a we consider the active dimensions as

we increase the number of representations for each action (colour), considering both the total number active and that per action.

In all cases, we can see that the total active policy dimensions (dashed) is slightly above the expected 4 for FlatLand, and the active policy dimensions per action is slightly above 1. This says that in all cases, the models were making occasional mistakes. Note however that the total number of active dimensions only increases significantly when the number of representations per action is equal to 3. As such, we feel it is reasonable to assume RGrVAE can deal with over representation, as long as you have some idea of the total number of actions. Only once the model becomes extremely over represented does it start failing. We note that even the worst model in this test still resulted in a good linear disentangled representation, it just had redundancy. It's possible that a stronger identity loss on the representations, would resolve this. Since the model still learns desirable representations, this does not appear to be a large problem.

We further demonstrate the quality of the representations in Fig. 4.25b where we plot representation and symmetry errors. Here we can see that even when over represented, the model still learns good estimates of post action observations (X), latents (Relative) and independence. The symmetry reconstruction is impacted, but not significantly. This does suggest that as we increase the over representation, the quality of the latent representation will decrease, however it appears to do so relatively slowly.

Robustness - Visual Noise Experiments so far have all been under the ideal conditions for each dataset. Since it is important to understand robustness of our models we now consider learning in noisy conditions - for a few different types of distractor. There will be two simple problems - Gaussian noise (mean 0, variance 0.01) and Salt+Pepper noise (per pixel probability 0.1) - and one complex problem, where the backgrounds of FlatLand images are replaced with images from CIFAR10. From studying these, we hope to determine if symmetry based models are overly reliant on the cleanliness of the data. Due to the complexity of learning linear disentangled representations, it is not currently feasible for us to train them on large scale real world datasets.

Fig. 4.26 provides independence scores for ForwardVAE and RGrVAE, alongside the symmetry reconstruction score and (for RGrVAE) an estimate of convergence time based on the independence of the policy. This estimate is given as the number of epochs required for the policy distribution to have an estimated independence score of (0.95).

From the independence (Fig. 4.26 - left), we can see that none of the distractors prevented learning. ForwardVAE showed lower independence for both the

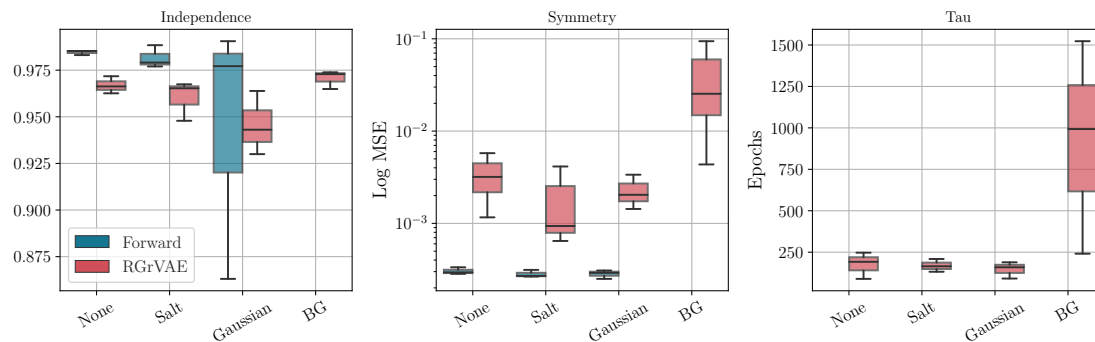


FIGURE 4.26: Metric scores for ForwardVAE and RGrVAE when trained on FlatLand under different distractors.

Salt+Pepper and Gaussian noises, but generally still being larger than 0.95. Note that we saw a single case of low independence, which was likely due to a failure in the independence estimation and not due to a poor representation, since the symmetry scores (estimated directly from the model) are very good. RGrVAE shows the same relationship, with the simple noises reducing the independence. Interestingly the complex background showed very similar independence to the standard (noiseless) case. This may be due to the significantly longer training time (due to significantly longer convergence time) refining the latent space independence.

Symmetry reconstruction scores (Fig. 4.26 - mid) for ForwardVAE are, as expected, very good. In all cases they are an order of magnitude better than RGrVAE, and very consistent across the simple noises. RGrVAE also appears to be unaffected by the simple noises, showing very similar errors in each case. Unsurprisingly, the complex noise resulted in worse estimation of the underlying symmetry. Since the problem is more complex, we expected the representations to suffer, although it is interesting that the independence was largely unaffected. In either case, the symmetry error was still small even in the worst cases (< 0.1), suggesting the representation is still of good quality.

Finally, we can see that (for RGrVAE) the convergence time (Fig. 4.26 - right) was only significantly slowed by the complex backgrounds. The simple noises are random in nature and it's likely there was never gradient toward learning spurious structures. For CIFAR10 images however, there is significant structure, much of it that might be shared between CIFAR10 and FlatLand images. Structures such as curves are obviously very useful in detecting the agent in FlatLand, and likely act as a strong distractor when occurring in CIFAR10 backgrounds.

4.6.2 Backbone VAEs

RGrVAE has the property that it can be combined with any standard VAE. We will briefly compare different VAEs as backbones for RGrVAE, and fine tuning on top of

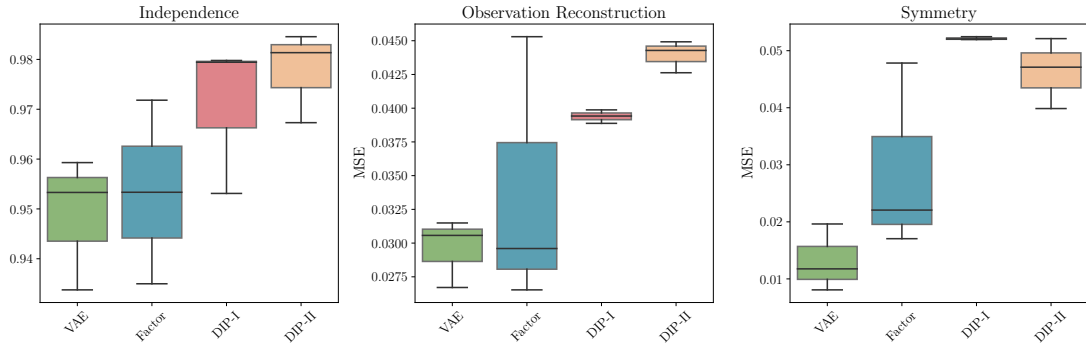


FIGURE 4.27: RGrVAE performance with different backbone VAE models. DIP-VAE backbones achieve slightly better independence, but slightly worse observation and symmetry errors. There is no consistent benefit.

pre-trained backbones to determine how they impact the representation.

Training Scheme All RGrVAE instances for this section were trained as outlined in Appendix Section C but with 400 epochs to allow for the more complex backbones to converge.

Backbones Fig. 4.27 reports independence, post action observation reconstruction and symmetry reconstruction for the different backbones. We can see that despite slightly higher independence scores on the DIP-VAE backbones, they result in larger observation and symmetry errors. The FactorVAE backbone shows similar independence to the standard VAE with better median post action observation reconstructions, but higher symmetry error. The FactorVAE model also generally lead to large variances in the score, whereas the default VAE and DIP-II model were more consistent. Given these observations, whilst the backbones only perform slightly worse than the standard VAE (which might be due to their increased complexity requiring additional training), there is no obvious benefit to them. As such, we should stick to the standard VAE which has fewer parameters and offers simplicity.

4.6.3 Policy Convergence

For RGrVAE to function correctly, the policy should converge to an injection between observed actions and internal representations. We will now explore policy convergence through the estimated independence score (the independence of the policy distribution).

Training Schemes For all the following experiments on policy convergence we will use the same training scheme (of Appendix Section C, barring the parameters under

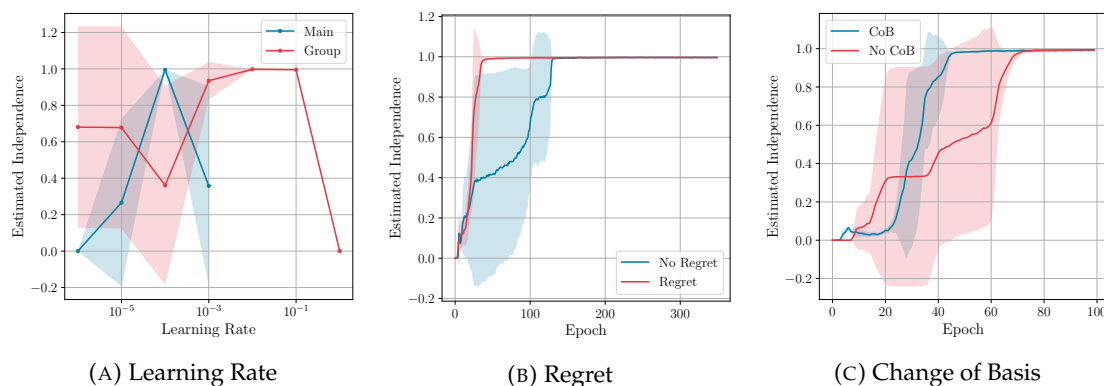


FIGURE 4.28: Policy convergence on FlatLand. RGrVAE is sensitive to learning rate choices, but less so to regret and change of basis. Solid lines indicate mean performance with 1 standard deviation given as the shaded area.

examination. The training time for each experiment is indicated by the x axis of the figures, since we found that different parameters required different amounts of training to converge adequately.

Learning Rates Choice of learning rate has been found to be vital for good policy convergence. In particular, the relationship between the internal representation learning rate ('Group' - Red) and the main learning rate of the VAE/policy network ('Main' - Blue). In Fig. 4.28a we see that the main learning rate is very sensitive, however the internal representation learning rate is less so. We found that the optimal choice for main learning rate was 10^{-4} (for a Group learning rate of 0.1). For the Group learning rate, (and Main learning rate 10^{-4}), any value between 10^{-3} and 10^{-1} results in consistent policy convergence. Anecdotally, we found that these learning rates transferred well to the dSprites dataset.

Regret Regret minimisation provides gradient to all possible actions and so should result in more consistent learning. The downside of regret minimisation is the vastly increased runtime, since it has to evaluate the reward for every action rather than just one. For this reason, we only use regret for the FlatLand experiments, but it is still interesting to determine its impact. From Fig. 4.28b we can see that, as expected, the use of regret leads to faster convergence with more consistency. Without regret, we see a large variance in the estimated independences although all models had converged within 150 epochs.

Change of Basis Similar to the regret, we only use a change of basis for the FlatLand experiments due to runtime considerations. The change of basis allows the latent space to encode actions into any plane, rather than sequential axis aligned planes. We might expect this to result in faster convergence, and from Fig. 4.28c, we can see that

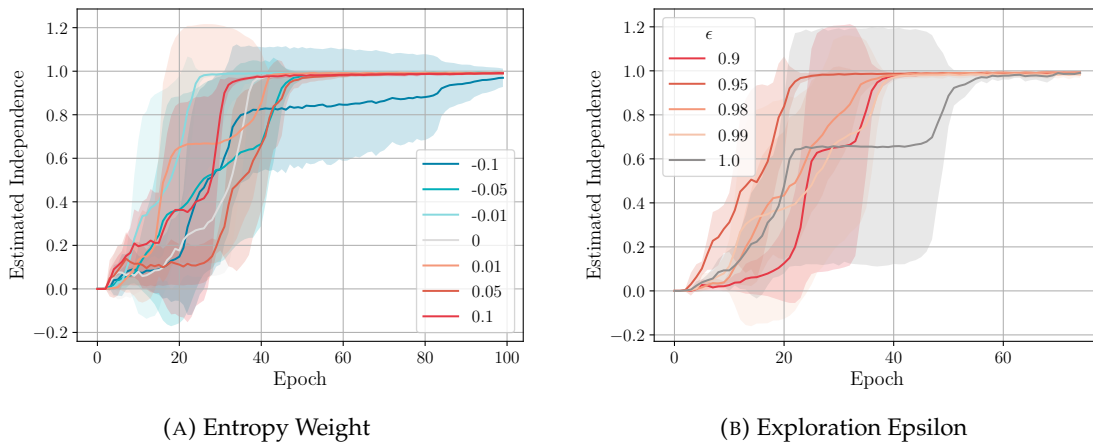
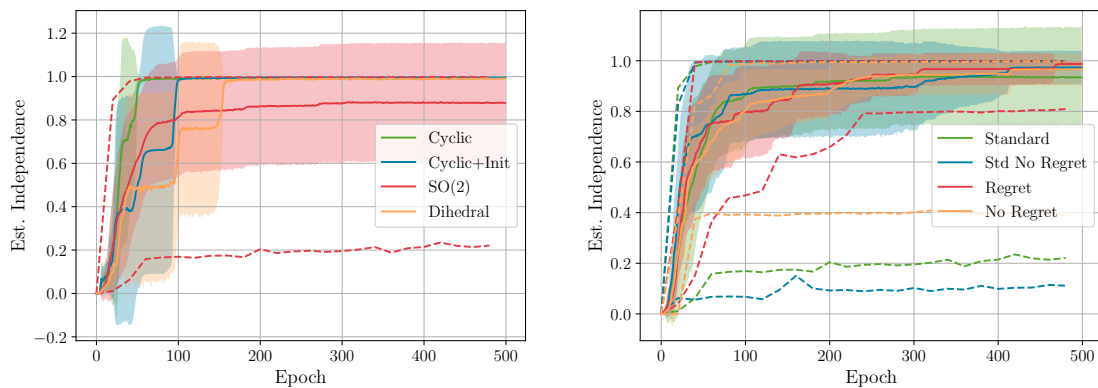


FIGURE 4.29: Exploration Strategies in RGrVAE. For FlatLand, entropy weighting was not particularly beneficial, but some amount of random action selection can offer faster convergence. Solid lines indicate mean performance with 1 standard deviation given as the shaded area.

using a change of basis results in more consistent convergence, with slightly improved convergence rate, on average.

Exploration Strategy In Section 4.3 we mentioned two different exploration methods, both of which are common in reinforcement learning and can be applied to GroupVAE - entropy weighting and random action selection. In Fig. 4.29, we report the estimated independences for different weightings of these strategies. Notice that in neither case is there a significant impact on convergence. First consider Fig. 4.29a, where we see no particular trend between the entropy weighting and performance. This suggests that for FlatLand, this exploration strategy is not vital for training. Possibly by the time the policy distribution has low enough entropy for the weighting matter, the model has already converged. Note that this may not stand for other datasets where the policy will be more complex. If we consider Fig. 4.29b, we can see that without exploration ($\epsilon = 1$), convergence is slowed, although again, not significantly. The fastest converging strategy was $\epsilon = 0.95$, which appears to be the optimal strategy for FlatLand.

Internal Representations In Section 4.3, we raised the idea of structuring internal representations to match desired symmetry groups. Thus far we have only considered data with cyclic symmetries, and so have tended to only use representations parametrised by cyclic rotation angle. In Fig. 4.30a we consider RGrVAE estimated independence on the FlatLand problem using a number of different internal representation choices - cyclic rotation angle, generic matrix $M \in GL(\mathbb{R})$, cycles or rotations from D_n , and unit determinant matrices $M \in SL(\mathbb{R})$. We can see that other than generic matrices, all methods converge rapidly to independent representations.



(A) Comparing different internal representation choices. Dashed line denotes the best/worst performing SO(2) run.

(B) Testing exploration scheme and regret choice. Dashed lines denote the best/worst run in each case. Errors are given disregarding the worst run.

FIGURE 4.30: Comparing RGrVAE internal representations on FlatLand. Solid lines indicate mean performance with 1 standard deviation given as the shaded area. Dashed lines indicate the min and max performance.

Notice that the fastest converging result was actually the vanilla cyclic representations, rather than those which are initialised to positive and negative angles (Cyclic + Init). Furthermore, notice that the introduction of reflection actions (Dihedral) seemed to reduce convergence rate. These results suggest that for FlatLand, simpler is better and we should avoid complicated internal representations.

For SO(2), we have plotted the mean and standard deviation (solid line with shading) alongside the max and min lines (dashed). Notice that they do not always fall into the local minima which are evident from the large deviations, sometimes they rapidly converge just like the cyclic representations (Fig. 4.30a, red dashed). We initially thought that the poor or local optima could be caused by the regret or the exploration strategy. Regret may be more likely to stick to local optima, since we are always encouraging the instantaneously best action. This might not be the same as the best action globally. The exploration strategy is more subtle. For FlatLand we have used a mix of entropy weighting and random action selection. Random action selection in particular can introduce a lot of noise into training, which might result in poor optima, whereas entropy weighting is less likely to cause this.

To determine if the exploration caused poor convergence, we report in Fig. 4.30b results with and without regret and also for a modified exploration scheme that removes the random action sampling - hopefully to avoid poor optima. We can see that in general, if we ignore the one poor run, the standard scheme actually beats both modified exploration schemes. Indeed both standard runs perform similarly, with a similar worst case run, although training with regret seems to slightly out-perform training without. The same is true of the modified exploration scheme. The main difference between the two exploration schemes is the worst case runs. The modified scheme tends to avoid the particularly bad runs seen in the standard scheme, however

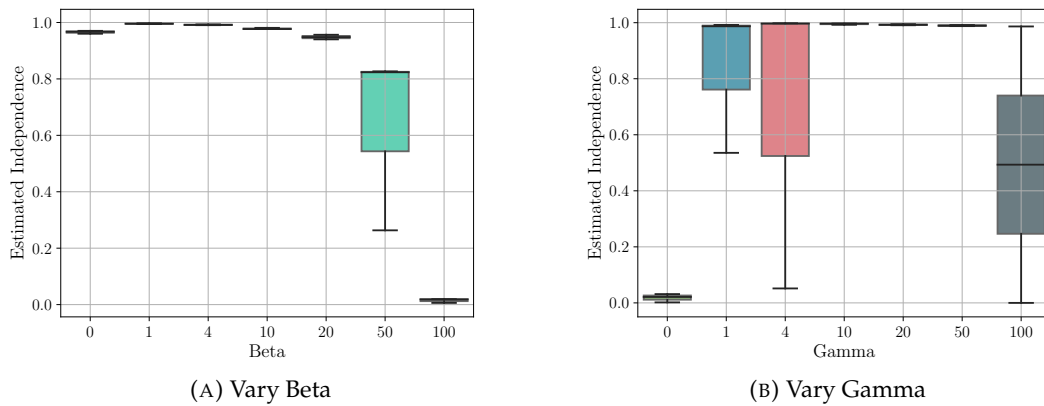


FIGURE 4.31: RGrVAE Hyperparameters. Large Beta and very small or very large gamma can result in poor performance.

cannot completely avoid poor convergence. Based on this, we can say that the random action sampling is likely introducing noise into the training and whilst it is fine for the structured internal representations (cyclic etc.), it may not be a good idea for more flexible or more complex representations. We can also say that regret is generally good for convergence although it should be noted that the runtime cost may not be worthwhile.

4.6.4 Hyperparameters

RGrVAE has two main hyperparameters which govern its behaviour. The capacity parameter β of β -VAE and the prediction weighting γ from Eqn. 4.3.

Training Scheme All models in this section were trained according to Appendix Section C, barring the hyper-parameter in testing and the training length which was set to 400 epochs.

Capacity β The β weighting in β -VAE is usually thought of as pressuring the posterior towards the prior. Indeed, it was shown by Hoffman and Johnson [2016] that $D(q(z)||p(z))$ is a component of the ELBO. The other interpretation of β is based on information channel capacity where the standard prior has 0 capacity. Thus, increasingly weighting the KL encourages reducing the information capacity of the latent space. In this light, it is unsurprising that with increased β we observe an inability to learn linear disentangled representations - indeed in the limit $\beta \rightarrow \infty$ we should be able to learn no representations at all. We demonstrate in Figure 4.31a that the estimated independence (policy distribution independence) is decreased with increasing β .

Prediction weight γ The γ term increases pressure to reconstruct the post-action latent. We can see from Figure 4.31b that when this weight is 0, the model doesn't learn to predict the actions in any way. As we increase γ , the estimated independence increases, until the model is extremely consistent at learning linear disentangled representations. Once the weight becomes too large, the learning begins to fail due to an optima where the posterior collapses to a single point. This naturally has $\mathcal{L}_{\text{pred}} = 0$ but $\mathcal{L}_{\text{VAE}} > 0$. Whilst in the figure, at $\gamma = 100$, linear disentangled representations likely form the global optima (some runs still converge), as we increase γ , this will no longer be the case. Whilst γ is obviously important, since RGrVAE continues to converge over a wide range of values, it should be relatively simple to find an appropriate choice for most problems. Indeed, we found that $\gamma = 10$ is suitable for both FlatLand and dSprites.

4.7 Conclusions

This chapter has aimed to answer 4 main questions:

1. Can we learn linear disentangled representations?
2. Are they learnt by standard VAE models?
3. How do we measure them?
4. Can we learn them without knowledge of action labels?

Can we learn linear disentangled representations? For the common disentanglement datasets (dSprites, FlatLand, etc.), any linearly disentangled representation will also have to disentangle the generative factors, since we define the symmetry groups to act on these factors. As such, we first tested the feasibility of learning an idealised simulator (decoder) and classifier (encoder). We found in Fig. 4.1 that this was feasible. Obviously there have been many VAE models which have shown disentanglement, so this was not surprising, but it was useful to begin our feasibility study with known results.

Given that linear disentangled representations have to disentangle actions, the next step was to demonstrate that we could learn to approximate these actions, given knowledge of which was applied. We found in Fig. 4.3 that this could be done when directly supervised with the post action latent and also indirectly supervised by the post action observation. For the standard VAE backbone, we did find that the ability to approximate the action is dependent on the complexity of the approximating function. In particular, a linear model was not expressive enough to capture the action

in latent space, however non linear models were. Requiring a non-linear model obviously shows that the representation was not linear, let alone linear disentangled. This however, is not sufficient to answer our question of whether standard models can learn linear disentangled representations, it simply showed that this particular model did not.

The final part of our feasibility study was to determine whether we can learn the (SBDRL) expected cyclic structure on dSprites. This was the major test of whether linear disentanglement was feasible. We found that we could indeed learn a (non-linear) mapping to a intermediate representation space where the dSprites actions (scale, rotation, translation) were cyclic of the appropriate degree. Furthermore, we found that the space had high quality linear disentanglement since the actions visualised in Fig. 4.5 showed minimal entanglement between actions.

Given these results, we find that linear disentangled representations were indeed feasible. In fact, under action supervision we could even transform the latent space (non-linearly) to a linearly disentangled one. Furthermore, we learnt an inverse mapping for this transformation which provides us with a full linearly disentangled VAE model.

Are they learnt by standard VAE models? Since linear disentangled representations are defined based on the symmetry group acting on the data, without access to actions labels, it's difficult to determine if a given model has learnt such a representation. For our tests, we used FlatLand where we know that the data was generated with cyclic symmetries acting on the data. Since standard VAE models do not observe transitions, if they learn linear disentangled representations, it is not guaranteed to be with respect to the symmetries used to generate the data.

Due to the simplicity of FlatLand, we assumed that if a model learnt linear disentangled representations, it would be cyclic, since they are one of the simplest symmetries. Since these action may not necessarily be with respect to the Cartesian, we test against both the Cartesian and 45 degree from Cartesian axes. Upon empirical search, we found that in general the classical VAE models *did not* learn linear disentangled representations with respect to cyclic groups. This is not to say that individual instances did not learn them. In fact, we found isolated instances of linear disentangled representations emerging from DIP-VAE-I and the standard β -VAE. This is very interesting since it suggests that there is no pressure against such representations in standard models, and that they might be encouraged without significant modifications to the training procedure - possibly just an additional 'linear disentanglement' loss term.

How do we measure them? Some symmetry groups can have irreducible representations of degree 2 or higher, leading to actions being encoded across multiple latent dimensions. This is at odds with classical disentanglement metrics which often (many but not all) assume good disentanglement has single generative factors encoded into single latent dimensions. As such we wanted to explore symmetry based metrics for measuring linear disentanglement. We proposed the Independence Score and the SVD overlap, which both aim to measure the extent that actions are encoded into separate latent subspaces. We also proposed the relative latent error in an effort to determine how equivariant the encoding map is. These metrics speak directly to different parts of the definition of a linear disentangled representation. The final metric we propose is a straight forward extension of the Mutual Information Gap (MIG) metric such that it can respond to factors being encoded across multiple dimensions. These metrics do not form a single metric of linear disentanglement quality, however in combination and conjunction with other metrics, they should provide a good understanding.

Can we learn them without action supervision? Prior work has shown that linear disentangled representations can be learnt when supervised directly on the actions that are observed. With GroupVAE, we aimed to show that we do not require the action labelling to learn high quality linearly disentangled representations. We show that we can infer the observed action, and use this to optimise internal representations towards the irreducibles for that action.

Whilst we proposed two GroupVAE variants, we found that the attentional model failed to learn good representations. The reinforcement variant (RGrVAE) however was found to learn high quality representations. We determined that the learnt action representations did not introduce large errors under composition, and were robust to visual noise. We also found that as long as we defined more internal representations than true actions, it continues to converge to a good representation. Furthermore, we will see in the next chapter that the policies learnt are extremely accurate at action inference and could potentially be used to label un-labelled action datasets. RGrVAE concretely shows that we can learn linear disentangled representations action supervision.

Chapter 5

Structure and Benefits of LDR

This chapter will explore the structure of linear disentangled representations through visualisation and empirical experiments. It is based on our NeurIPS workshop paper [Painter et al., 2020b]. There are two important parts of VAE representational structure, the posterior and the prior. The posterior shape determines the representation whilst the prior restricts the posterior shape. Given that we expect LDRs to be consistently structured, we might imagine that they have distinct characteristics in disentanglement metrics and performance on downstream tasks. Since it was found by [Locatello et al., 2018] that there was no correlation between disentanglement metrics and data efficiency in classical models, we will also explore this aspect.

Specifically, over the course of this chapter we look to answer the following questions,

- What is the posterior structure of FlatLand linear disentangled representations?
- Can we find better priors for FlatLand?
- Can we find evidence of LDRs in disentanglement metrics?
- How do LDRs compare to classical representations under data efficiency and downstream tasks?
- Can we find any correlations with data efficiency?

5.1 Cyclic Structures

This section will look at structure of the posterior and prior distributions of linear disentangled representations with respect to group structure $G = C_N \times C_N$. Since cyclic structure is the most common in disentanglement datasets, e.g.

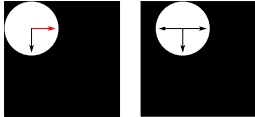

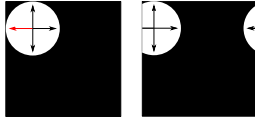
	FlatLand Boundary Condition		
	None	Contact	Gradual
Name	FlatLandN	FlatLand	FlatLandG
States	$\{(x_i, y_j) i, j \in [R, 64 - R]\}$	$\{(x_i, y_j) i, j \in [R, 64 - R]\}$	$\{(x_i, y_j) i, j \in [0, 64]\}$
Boundary	$g_x \circ x_{64-R}, g_x^{-1} \circ x_R$ are not observed.	$g_x \circ x_{64-R} = x_R,$ $g_x^{-1} \circ x_R = x_{64-R}$	$g_x \circ x_{64} = x_1,$ $g_x^{-1} \circ x_1 = x_{64}$
Symmetry	Undefined	$\approx C_7 \times C_7$	$\approx C_{13} \times C_{13}$
Samples			

TABLE 5.1: Definition of FlatLand variants and example samples. Arrows indicate possible actions. Red arrow indicates action chosen for that sample. ‘None’ restricts actions at a boundary so that the object never cross it. ‘Contact’ warps the agent to the other side of the grid on contact. ‘Gradual’ allows continuous transitions through boundaries. The radius of the agent is given by $R = 15$ and the step size corresponding to an action is 5 pixels.

FlatLand [Caselles-Dupré et al., 2018] and dSprites [Higgins et al., 2017], we will be exploring variants of FlatLand with different cyclic symmetries. This will help us answer the first question set out in the chapter introduction, “What is the posterior structure of FlatLand LDRs?”.

Varying Symmetries on FlatLand Since FlatLand is such a simple dataset, it is good for rapid experimentation and we will find it useful to define different sets of symmetries acting on it. The simplest way to do this would be to vary agents step size per action resulting in different order cyclic symmetries. Instead, we will vary how the agent interacts with the boundary, in the hope that it will add some additional complexity over simply changing the step size. We summarise the different boundary conditions in Table 5.1 in terms of the state spaces S , actions \circ and boundary conditions. The Contact and Gradual conditions are similar, the former being the usual FlatLand warping as the agent touches the boundary and the latter allowing gradual (continuous) transition through the boundary. Both these conditions lead to a cyclic symmetry structure with different degrees. The None boundary condition is different, we simply do not allow the model to observe actions into the boundary. This leads to an undefined symmetry structure, since the models will never observe all actions in all states. The main benefit of this is that we can explore symmetry based models under incomplete information, a situation which would likely be prevalent in real world problems. We have denoted the different boundary conditions by appending a letter as follows, (N)one - FlatLandN and (G)radual - FlatLandG. Contact boundary conditions is equivalent to the original dataset, FlatLand.

Dataset	$d((x_1, y_1), (x_2, y_2))$
FlatLandN	$ \Delta x + \Delta y $
FlatLand	$\min(\Delta x, (N_x - 30) - \Delta x) + \min(\Delta y, (N_y - 30) - \Delta y)$
FlatLandG	$\min(\Delta x, N_x - \Delta x) + \min(\Delta y, N_y - \Delta y)$

TABLE 5.2: Metrics which describe how many actions are required to move between two states in each FlatLand variant, assuming a step size of 1.

Action Metrics on Flatland Variants In ForwardVAE, we have the action estimation term $z_a - \hat{z}_a$, which is minimised when the action a can be approximated by it’s corresponding internal representation. Since the internal representations are fixed for any given batch, the action is encouraged to take the same form across the whole latent space.

On FlatLand, the symmetries are cyclic with order N , and we only ever observe action by the cyclic generators. As such, the internal representations should tend towards rotation matrices with rotation angle equal to $\frac{2\pi}{N}$. In this case, the distance between two states when encoded into the latent space will be encouraged to be proportional to the number of actions it requires to move between these two states.

For this reason, it will be useful to have metrics that tell us the minimum number of actions it takes to move between two states in each of the flatland variants. Each state (and observation image) in flatland is entirely described by the x-y position of the agent, so we define the metric $d((x_1, y_1), (x_2, y_2))$ in terms of the two states $s_1 = (x_1, y_1)$ and $s_2 = (x_2, y_2)$. We will also make use of the quantities $\Delta x = |x_2 - x_1|$ and $\Delta y = |y_2 - y_1|$, and the width and height of the FlatLand images $N_x = 64$, $N_y = 64$.

These metrics assume a step size of 1, and in reality, we will use a step size of 5 to be consistent with Caselles-Dupré et al. [2019], and the original FlatLand dataset. Despite this, since we will only ever observe states that differ by application of one action, the metrics will still be proportionally correct.

Training Scheme Unless otherwise stated, all ForwardVAE instances trained in this chapter use the same training scheme, as detailed in Appendix Section C.1.

5.1.1 Posterior Structure

In Chapter 4, we have seen copious examples of symmetry based models learning the FlatLand problem. In this subsection we will assume that a linear disentangled representation has already been learnt and observe the resulting posterior distribution structure. This will directly target the first question we posed in the chapter introduction, “What is the posterior structure of FlatLand LDRs?”.

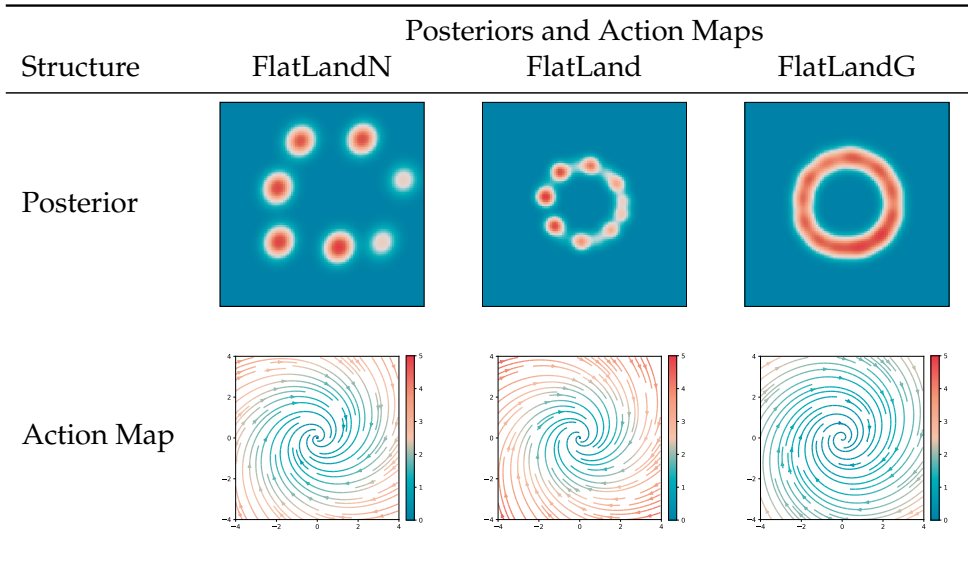


TABLE 5.3: Posteriors and action maps for FlatLand variants on the latent subspace of a ForwardVAE which corresponds to y translation on each of the dataset variants. Both posteriors and action maps have the same axis limits

Visualising the Posterior Throughout this chapter we will be making use of posterior visualisations. To do this we will need to estimate the posterior density over the latent space. For this task, we first encode all images in the dataset and store the resulting latent codes. Rather than try to visualise a high dimensional space, we will be restricting to visualising the posterior in 2D. So we next select the two dimensions which we want to visualise - generally these dimensions will be chosen based on which dimensions are affected by a group action. After selecting the two dimensions, we have a set of points in 2D on which we can run a Gaussian kernel density estimation (by Scipy¹) to get an estimate of the posterior density.

In addition to the posterior, we will also sometimes visualise the action map for the same two dimensions. To do this we take the internal representation from ForwardVAE which acts on those two dimensions and create a stream-plot for each point in the space. This is done by matrix-vector multiplication between the internal representation matrix and each point in the space. This then allows us to visualise how the action moves each point in the space.

Cyclic Posterior In Table 5.3, we present posteriors of a ForwardVAE trained on each of the FlatLand variants. In this case, the posteriors are on the plane corresponding to y translation. Since on FlatLand all component groups are identical, this plane should be fairly representative of the action plane corresponding to x translation, assuming both actions had been learnt to a good degree.

¹https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html

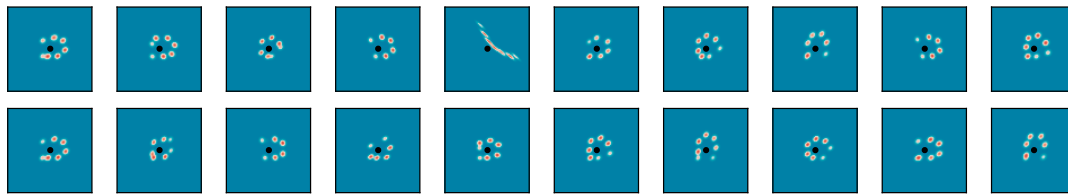


FIGURE 5.1: Visualisations of the posteriors for both action planes of 10 ForwardVAE instances trained on FlatLandN.

Notice that in all cases we observe some circular structure, with the posterior ‘wrapping’ around the origin. On FlatLand and FlatLandG this is expected, as the irreducible representations of C_N are rotation matrices. Note that as the order of the symmetry group increases, the number of distinct peaks increases and they are merged into smoother distributions. This can be seen by comparing the distinct peaks of FlatLand to the relatively smooth posterior on FlatLandG. We have also provided visualisations of the action via the ‘action maps’, which are simply stream plots of the field generated by applying the matrix representation at each point. Notice that FlatLandG shows wide and slow convergence towards the origin, whilst the other two datasets have faster rate of change and convergence. Converging towards the origin is (inversely) related to the quality of the learnt action, since it is proportional to the rate at which errors are introduced under composition of the action (assuming the action is cyclic). Beyond being the origin of the latent space, and the KL divergence term in the ForwardVAE loss encouraging the posterior to be centred around this point, to our knowledge, there is no additional significance to it.

It is particularly interesting that we continue to see circular structure even when cycles are not explicitly observed in the data (i.e. FlatLandN). Fig. 5.1 presents a set of posteriors from 10 independently trained models, where we can see that the cyclic behaviour is dominant, with non cyclic behaviour occurring only once, and only in one of the two action subplanes (rows). The non-cyclic behaviour is likely due to unlucky initialisation and the $\mathcal{L}_{\text{recon}}$ pressure overcoming that of the KL divergence. The KL divergence of a cyclic posterior is lower than that of a non-cyclic one since more of the peaks are close to the origin or overlapping. To lower the KL of the non-cyclic representation, the posterior would need to shift to the other side of the origin. From the action map (Fig. 5.2a), if we lay it over the posterior (Fig. 5.2b left) and shift the posterior to the other side of the origin (Fig. 5.2b right), the stream lines no longer travel through high probability states. As such the gradient from the two loss terms act to push the posterior in different directions, with the equilibrium appearing as the posterior shown.

How does the objective influence the latent structure In the VAE, the KL-divergence loss against the isotropic Gaussian prior tends to encourage the

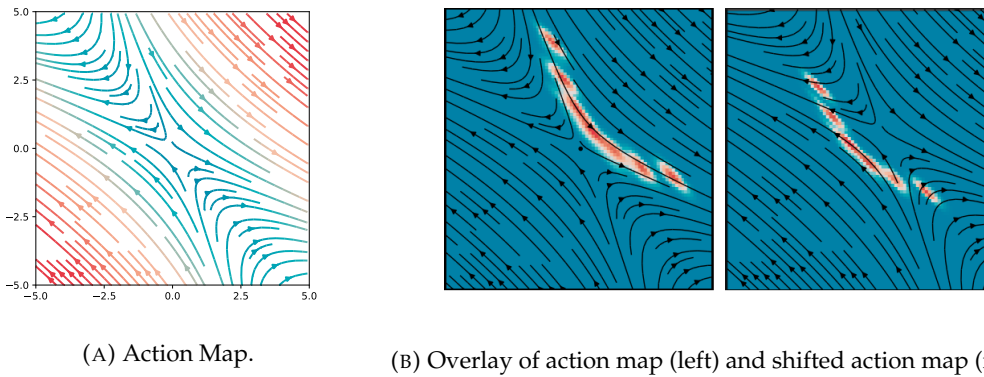


FIGURE 5.2: Action map on the non-cyclic posterior seen in Fig. 5.1.

overlap in posterior peaks (Recall the data locality argument of β -VAE outlined in Section 3.2.2). It is our intuition that the cyclic posteriors on FlatLandN are driven by this same pressure. In order to test this we can ablate it from the objective.

Does the KL cause Wrapping Posteriors on FlatLandN? We can determine if the KL divergence contributes to this unexpected circular structure on FlatLandN by ablating it or varying its weighting in the final objective. To do this, we can modify the ForwardVAE objective term from Equation 4.2 using parameter β ,

$$\mathcal{L}_{\text{ForwardVAE}}(x, z_a) = (x - \hat{x})^2 + \beta \text{KL} + (z_a - \hat{z}_a)^2 \quad . \quad (5.1)$$

We then train ForwardVAE instances with the value of β in the set $\{0, 0.1, 0.2, \dots, 0.9, 1.0, 10.0\}$.

The objective has three main components, the reconstruction term $(x - \hat{x})$, the KL term and the action estimation term $(z_a - \hat{z}_a)$. It is likely that the reconstruction term alone shouldn't have a large effect on the structure of the latent space, since a sufficiently complex decoder should be able to map any latent point to any image. As such, it is the KL and action reconstruction terms that we suspect will dictate the latent space structure. In Fig. 5.3 we present posteriors for ForwardVAE instances trained on FlatLandN (See Appendix Section C.1) whilst we vary the KL weighting β . This allows us to determine what effect the KL term has on the latent structure.

Low KL ($\beta < 0.2$) In the low KL regime (< 0.2), we see no wrapping around the origin. Indeed the posterior is arranged almost linearly and is very similar to the non-cyclic posterior seen in Fig. 5.1. This may actually be the expected structure in absence of the KL term. Since the KL term has such low weighting, the structure is almost entirely determined by the action estimation term. Consider the action metric for FlatLandN from Table 5.2. Under this metric, the distance between two states is

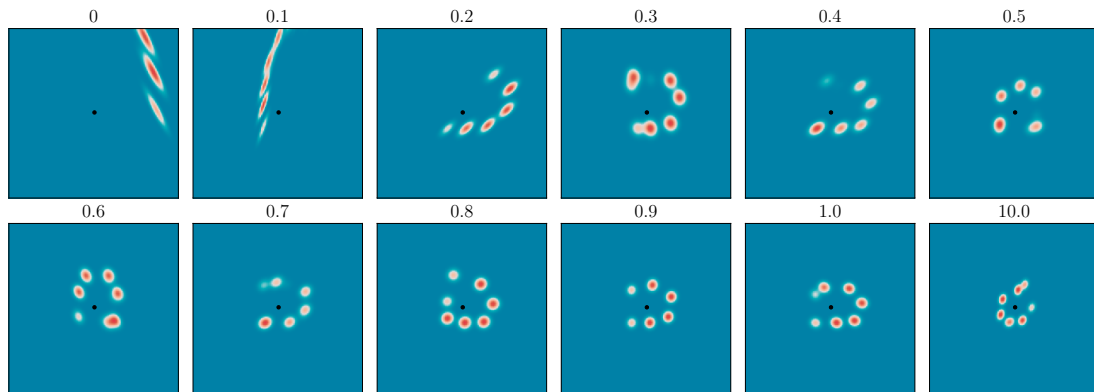


FIGURE 5.3: FlatLandN posterior distributions with varying KL-Divergence weightings. See the general trend towards clustering around the origin at higher KL weightings. Low weightings tend to align with the Cartesian axes.

proportional to how close we would consider the agent in the image. In the low KL regime, we find that the latent structure aligns with the action metric. The posterior peaks corresponding to states near opposing boundaries are mapped far away from each other.

In this specific case, of the low KL regime on FlatLandN, the action metric aligns with the data locality (Recall Section 3.2.2, β -VAE). For FlatLand and FlatLandG, the action metric is different, and points on opposing boundaries are close together under the metric but not visually. Despite this, the latent space is structured cyclically on those datasets, which shows that the action estimation term can overcome the structuring pressure that comes from the KL term. We will see in the next paragraph that on FlatLandN, it is not always the case that the latent structure adheres exactly to the action metric.

Medium - High KL ($\beta > 0.2$) As the KL pressure is increased (> 0.2), we see increased wrapping around the origin and very little structural change for weights above 0.5. In the high KL regime (10), the posterior is wrapped tighter around the origin and individual peaks have lower variance. This suggests that if we increase the KL weighting further, we will eventually lose the ability to reconstruct since all states will overlap in the posterior. This is a known phenomena in VAEs, and is easiest understood when we consider the VAE latent dimensions as a set of noisy information channels [Burgess et al., 2018] (as discussed in Section 3.2.2 paragraph entitled β -VAE).

From the figure, it is obvious that strong KL weighting causes the posterior to wrap around the origin on FlatLandN. It seems reasonable to assume that this helps increase the overlap in the posterior peaks (as discussed in Section 3.2.2, β -VAE), which is a direct consequence of the Gaussian prior. We will see in Section 5.1.2 that when we allow more expressive priors, this no longer occurs, which provides further evidence for this assumption.

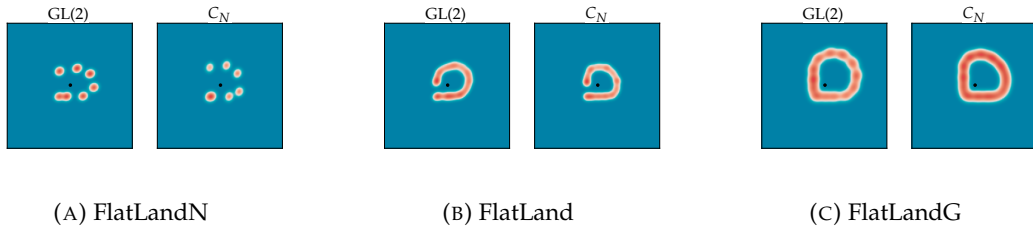


FIGURE 5.4: Posterior matrix vs cycles

Note that this discussion of the KLs effect on FlatLandN does not directly extend to FlatLand/FlatLandG, since on these datasets, states at opposing boundaries are close together under the action metric $\|\circ\|_a$ (despite being far away in pixel space). As such, the latent reconstruction term may also introduce pressure towards this cyclic posterior structure.

Internal Representation In the previous Section, and particularly in Fig. 5.3, we saw that cyclic structures were learnt even in the absence of cyclic groups acting on the data (FlatLandN). We found that this posterior behaviour was due to the KL divergence term in the loss. Choice of internal representation also provides a means to bias learnt structures. We would like to determine if there are any structural differences or advantages gained from different internal representation types and if they change the optimisation result.

ForwardVAE is defined with generic matrix internal representations (GL(2)), however as with GroupVAE, we can replace these with representations associated with different symmetry groups. In particular, we would like to compare the posteriors of GL(2) internal matrix representations with cyclic group C_N internal representations - where we learn purely a cyclic angle. In Fig. 5.4 we compare posteriors for ForwardVAE instances trained on the FlatLand variants with the two internal group structures, and see no obvious difference in the learnt posteriors. We saw in Chapter 4 (Section 4.6.3) that appropriate internal representation choice might allow for faster convergence, but there appears to be little to no *structural* benefit on FlatLand. This may differ for more complex datasets however, and we found that on dSprites, cyclic representations were overly restrictive and actually slowed down learning.

Recall (Fig. 5.3) that in the absence of the KL term, the posterior no longer had cyclic structure on FlatLandN. Whilst cyclic internal representations appear to offer no structural benefit on the standard ForwardVAE, they may have sufficient bias to learn cyclic posteriors when the KL is ablated. As previously mentioned, on FlatLandN, there is no explicit pressure towards or against cyclic structures, either from $\mathcal{L}_{\text{latent}}$ or the action metric.

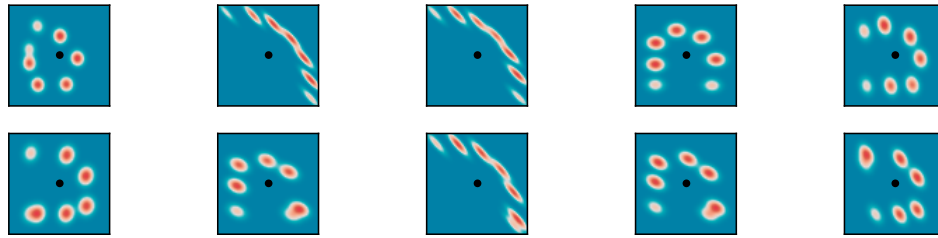


FIGURE 5.5: Example posteriors from a ForwardVAE with cyclic internal representations and 0 KL weighting. Note that the non-cyclic posteriors were significantly further away from the origin and all axes are scaled to best show structure.

Since the KL term is integral to ForwardVAE, ablating it is not particularly informative for general use cases. It does however allow us to understand the extent of the bias introduced by changing internal representations. In Fig. 5.5 we present a number of example ForwardVAE posteriors using cyclic internal representations and whilst it appears that there is a strong bias towards cyclic structures, they are not guaranteed. Indeed we continue to see similar structures as in Fig. 5.3 ($KL \leq 0.2$), which do not wrap around the origin. Note that the major difference between the posteriors presented in Fig. 5.5 is the order of the *learnt* cyclic group. Wrapping posteriors show lower order than the more linear ones, but most states in the latter don't have much posterior density. Also of note is that since the internal cyclic representations are learnt through purely cyclic angle, they force the posterior to be near, and curve (but not necessarily wrap) around the origin. When learning $GL(2)$ representations, the model is free to learn rotation matrices about any point in the latent space. The KL term generally just encourages the posterior to be located near the origin, but in its absence (i.e. Fig. 5.3), we see the posterior is free to curve around any point.

Other Data Thus far we have only shown that cyclic posteriors are present on FlatLand and its variants. In Table 5.4 we show that RGrVAE learns cyclic posteriors (of varying quality) on both dSprites and Norb. We provide these examples to show that the posterior structure which we visualise on FlatLand variants is generally representative of that found in other more complex problems also.

The symmetry for dSprites translation is C_{40} , dSprites rotation is C_{32} and Norb illumination is C_5 . For the two dSprites posteriors, we can see obvious, high order cyclic structure, as would be expected by their symmetries. The dSprites translation posterior shows density unevenly distributed between states, which may correspond with biases in the training set used (90% train with 10% test) or inherent in the dataset itself (there may be few instances of objects at boundaries). The rotation posterior shows more consistency in the density peaks, however it has additional density, not expected in the standard cyclic structure. This extra density may be simply due to limited training time, or it could be due to different shapes in dSprites having

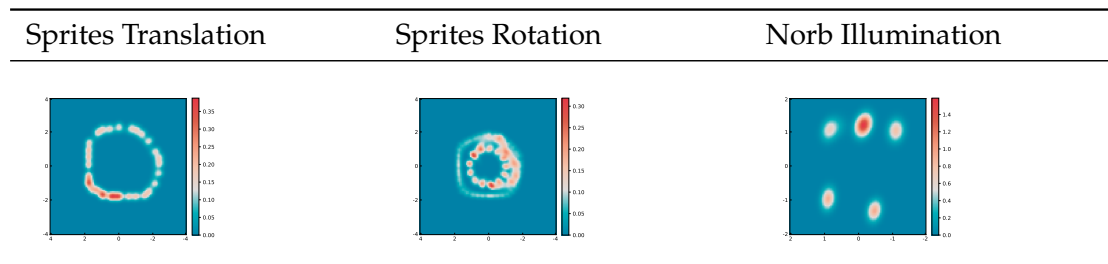


TABLE 5.4: Additional posterior visualisations for actions.

different rotational symmetries (due to rotational symmetries inherent to the shape). The final example, of illumination on Norb is fairly reminiscent of a lower dimensional FlatLand representation.

5.1.2 Prior Structure

It is obvious from our previous posterior visualisations, that the standard Gaussian is not a good prior for the FlatLand problems, at least in the sense of accurately representing the true distribution of the data. As such, the posterior which we visualise may not be an accurate representation of the true prior of the generative process. If we allow more complex priors, we may be able to better visualise this distribution. In this section, we will explore more complex priors, in aims to answer the second question posed in the chapter introduction, “Can we find better priors for FlatLand?”. Furthermore, we would like to explore whether these more complex priors, and consequently the potential for more complex posteriors, are a benefit or hindrance to the learning problem.

In VAEs, we have two easy ways to change the prior distribution. We could analytically define a set of priors, find the KL-Divergence between them and a number of different distributions, and sample in the VAE accordingly. Alternatively, we could allow a learnable prior, through methods such as normalising flows. The first approach has the advantage that we have the exact form of the prior and can perform analysis using this. The major downside is that it is limited by the set of priors that we choose, it may be that we do not test the best priors. We will instead explore normalising flows, which optimise towards the best possible prior (which minimises $KL(q(z|x)||p(z))$) for each case. The major downside of this approach is the lack of an analytical form.

Normalising Flows In general terms, normalising flows [Rezende and Mohamed, 2015] are invertible mappings $f : X \rightarrow Y, (X, Y \subset \mathbb{R}^d)$ that are differentiable by automatic gradient methods. If we consider the probabilities $p(\mathbf{x})$ and $p(f(\mathbf{x})) = p(\mathbf{y})$

then by multivariate change of variables, we have the relationship in log space,

$$\log p(\mathbf{y}) = \log p(\mathbf{x}) + \log \left| \det \frac{\partial f^{-1}}{\partial \mathbf{y}} \right| = \log p(\mathbf{x}) + \log \left| \det \frac{\partial f}{\partial \mathbf{x}} \right|^{-1} , \quad (5.2)$$

where the Jacobian $J_f = \frac{\partial f}{\partial \mathbf{x}}$ acts to scale the probabilities such that the integral remains 1. Note that the second equality follows from the inverse function theorem since f is invertible.

The change of variables can be easily composed, consider

$$z_0 = x, \quad z_i = f_i(z_{i-1}) \quad \text{where} \quad f_N(z_{N-1}) = y, \quad f_i : Z_{i-1} \rightarrow Z_i , \quad (5.3)$$

then,

$$\begin{aligned} \log p(y) &= \log \left| \det \frac{\partial f_N}{\partial \mathbf{z}_{N-1}} \right|^{-1} + \log p(z_N) \\ &= \log \left| \det \frac{\partial f_N}{\partial \mathbf{z}_{N-1}} \right|^{-1} + \log \left| \det \frac{\partial f_{N-1}}{\partial \mathbf{z}_{N-2}} \right|^{-1} + \log p(z_{N-1}) \\ &= \dots \\ &= \sum_{i=0}^N \log \left| \det \frac{\partial f_i}{\partial \mathbf{z}_{i-1}} \right|^{-1} + \log p(x) . \end{aligned}$$

We now have a means to take a (possibly complex) distribution and iteratively map it to another distribution, assuming our functions f_i are invertible and we can compute the Jacobians. Indeed, there is nothing preventing these functions from having learnable parameters enabling them to be optimised by back propagation. It should also be noted that the number of functions f_i can be thought of as the number of layers in the flow, and the more layers, the more complex the flow can be. These ideas are the driving force behind normalising flow methods in deep learning. There have been a number of such learnable functions proposed in the literature, we will be exploring planar flows, radial flows (both [Rezende and Mohamed, 2015]) and non-volume preserving flows (RealNVP [Dinh et al., 2016]) since they are relatively simple and easy to implement compared to more advanced methods such as GLOW [Kingma and Dhariwal, 2018] and PixelCNN [Oord et al., 2016].

Planar Flow Planar flows simply contract or expand space in the direction perpendicular to a hyperplane which is defined by free parameters \mathbf{w} and b (see example in Fig. 5.6a). The contraction or expansion is determined by free parameter \mathbf{u} , which also allows shearing along the plane. The flow is then defined by

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T \mathbf{z} + b) , \quad (5.4)$$

where h is a smooth non-linearity with derivative h' . Conditions for which this is invertible are given by [Rezende and Mohamed \[2015\]](#). They provide the Jacobian determinant as,

$$\left| \det \frac{\partial f}{\partial \mathbf{z}} \right| = \left| 1 + \mathbf{u}^T h'(\mathbf{w}^T \mathbf{z} + b) \mathbf{w} \right| \quad (5.5)$$

Radial Flow Radial flows expand and contract (through free parameters $\alpha > 0, \beta$) space about a point \mathbf{z}_0 (see Fig. 5.6b) rather than a plane, as in the planar flows. Where α controls the steepness at \mathbf{z}_0 , β controls the expansion/contraction degree and $r = r(\mathbf{z}) = \|\mathbf{z} - \mathbf{z}_0\|_2$. The flow is defined by,

$$f(\mathbf{z}) = \mathbf{z} + \frac{\beta}{\alpha + r(\mathbf{z})}(\mathbf{z} - \mathbf{z}_0) \quad , \quad (5.6)$$

where again, conditions for being invertible are given by [Rezende and Mohamed \[2015\]](#). They provide the Jacobian determinant as,

$$\left| \det \frac{\partial f}{\partial \mathbf{z}} \right| = \left[1 + \frac{\beta}{\alpha + r} \right]^{d-1} \left[1 + \frac{\beta}{\alpha + r} - \frac{\beta}{(\alpha + r)^2} \right] \quad (5.7)$$

RealNVP Unlike the previous two methods, RealNVP introduces autoregressive components, where dimension i of some variable z depends on all dimensions $\{0, \dots, i-1\}$ of z . RealNVP takes advantage of a particular construction of the flow function f that is invertible and allows the Jacobian to be easily calculable. It is given by,

$$f(\mathbf{z})_i = \begin{cases} z_i & \text{if } i \in \{0, \dots, M\} \\ z_i \odot \exp(s(z_{0:M})) + t(z_{0:M}) & \text{if } i \in \{M+1, \dots, N\} \end{cases} \quad , \quad (5.8)$$

where $t, s : \mathbb{R}^M \rightarrow \mathbb{R}^{N-M}$ and $z_{0:M}$ denotes the vector comprised of the first M dimensions of vector z . Notice that in the second case, s and t are independent of $z_{M+1:N}$, resulting in the following,

$$\frac{\partial f_i}{\partial \mathbf{z}_j} = \begin{cases} 1 & \text{for } i, j \in \{0, \dots, M\} \\ \exp(s(z_{0:M})) & \text{for } i, j \in \{M+1, \dots, N\} \\ 0 & \text{for } i \in \{0, \dots, M\}, j \in \{M+1, \dots, N\} \end{cases} \quad . \quad (5.9)$$

These three quantities are trivial to calculate so the only difficult terms in the Jacobian are the remaining cross terms. However since the Jacobian is lower triangular, the determinant is equal to the product of the diagonals,

$$\left| \det \frac{\partial f}{\partial \mathbf{z}} \right| = \left| \exp \left(\sum_{j=M+1}^N s(x_{0:M})_j \right) \right| \quad , \quad (5.10)$$

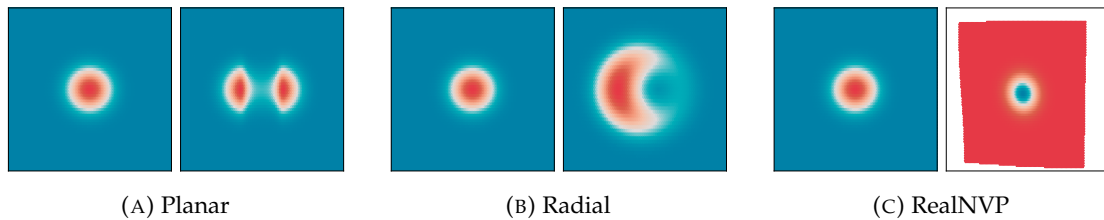


FIGURE 5.6: Example flows on a Gaussian distribution. Note that the RealNVP is simply randomly initialised, and isn't representative of all possible flows. For both Planar and Radial flows, we show a single flow layer, in general models use many layers, we use 10-20 for our models.

which is simple to compute. An example of a randomly initialised RealNVP flow is given in Fig. 5.6c. Our particular implementation of RealNVP essentially combines two steps of the above process, swapping the terms kept constant/conditioned upon, i.e. first condition upon $0 : M$ and then condition upon $M + 1 : N$. The result is that after the combined step, all dimensions have been transformed and we retain the ability to easily compute the Jacobian determinants.

Complex Priors with Normalising Flows In Fig. 5.7, we diagram the concept behind emulating a complex VAE prior using normalising flows which sequentially adapt a (potentially complex) posterior towards the standard prior. The basic idea is to map the posterior to a parametrised posterior which is as close to the standard prior as possible. This allows us to compute the standard KL term whilst retaining the flexibility of the complex posterior. It is important to note that the flow is used solely for computing the KL term, not when decoding reconstructions, since the latter would be simply adding complexity to the encoder, and not emulating a change in prior.

Experimental Setup For all experiments with flow based models, we use the same setup. We train 5 instances of a ForwardVAE using each flow type (plus no flow). Training of these models is done in the same way described in Section 5.1. All metrics reported are measured on a validation set of 10% of the dataset, which is held out from the training data.

Basic Properties To judge the effectiveness of the flows, we can consider the KL divergence. As a measure of relative entropy between the posterior and prior, this tells us the efficiency of the VAE encoding (relative to the prior).

In Fig. 5.8a we provide violin plots of the KL-divergence error for each flow type (i.e. we have 5 KL error scores for each flow type corresponding to the 5 model instances trained). Violin plots use kernel density methods to approximate a distribution over

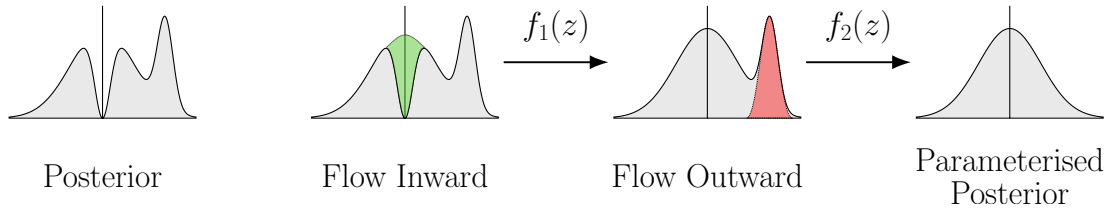


FIGURE 5.7: The concept behind normalising flows to emulating prior change. We allow a complex VAE posterior that is gradually mapped back to the standard prior.

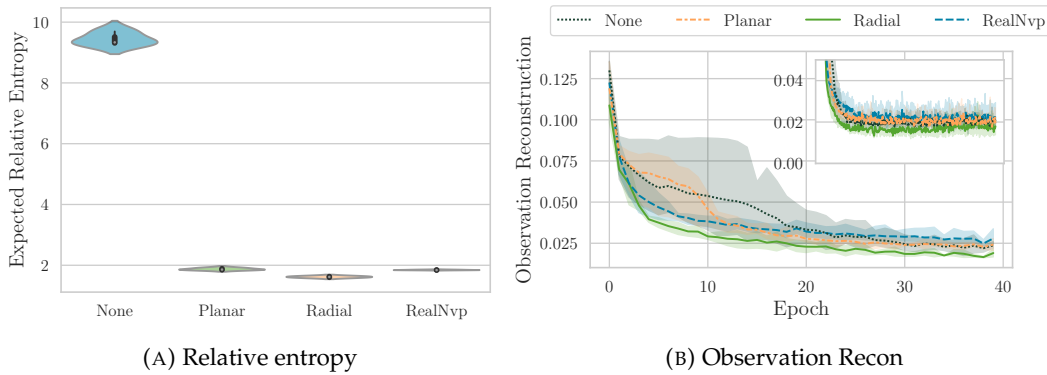


FIGURE 5.8: Comparison of priors under normalising flows on FlatLand. (A) Kernel density estimation for the validation KL-Divergence error term for flow based models. (B) Convergence of validation observation reconstruction error as the model is trained. Lines indicate mean whilst shaded area indicates 1 standard deviation.

the data, and in this instance we use the default kernel settings from matplotlib. [Hunter, 2007]. From the figure, we find that the relative entropy (expected over the validation latent space) achieved by flow based methods is far lower than that of a standard ForwardVAE model ('None'), showing the standard Gaussian prior is not a good fit for FlatLand. In addition, the flow based models were far more consistent in their scores. It does appear however, that there is very little difference between the flow types on FlatLand, likely due to it's simplicity. We evidence this further in Fig. 5.8b which reports the expected post-action observation reconstruction $\|x_a - \hat{x}_a\|$ under the different flows. We see that all flows again perform similarly whilst the standard prior performs with less consistency in early training (grey shading) however achieves similar reconstruction in the long term.

We can see examples of posterior and parametrised posterior (posterior after the flow has been applied) in Table 5.5. Observe that the standard model ('None') has a slight tendency to align with the Cartesian axes. This is a direct consequence of the KL term, the standard estimator of which is the sum of the divergence for the individual dimensions, although there are alternative forms (e.g. Hoffman and Johnson [2016]). To learn a strong representation under the standard prior, the model has to fight this axis-aligning pressure at the expense of relative entropy and slower convergence, as we saw in Fig. 5.8b. The flow based models allow the posterior to be any shape, as we see from the Planar and RealNVP models, which do not align with the Cartesian.

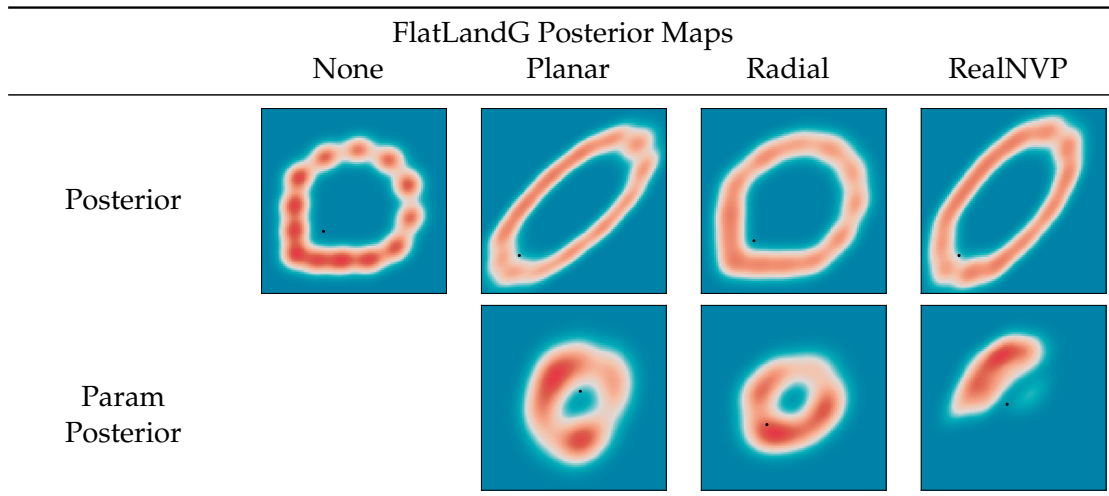


TABLE 5.5: Example posteriors and parametrised (i.e. after the flow layers) posteriors for FlatLandG. Note that colour scale is constant for the posteriors but not for the parametrised posteriors, which are significantly higher density than the posteriors. Black dot indicates the origin. ‘None’ refers to the baseline model with no flow layers.

Interestingly the Radial flow model still appears to show some alignment near the origin. Position relative to the origin is also important, since the standard prior is centred around it. For the baseline model (‘None’), we can see the origin is close to a region of higher density, which helps reduce the divergence with the prior. The flow based models no longer have this restriction and are free to map the posterior anywhere in the space, only limited by the complexity of the flow. We can see that all of the flow based models continue to map the posterior near the origin since this reduces the requirements on the flow and does not harm the likelihood term of the objective. The last noteworthy property of Table 5.5 is that the flow based models tend to show more consistent density across the posterior. The standard model posterior increases density near the origin, whereas the flow based models show lower but more consistent density throughout.

Looking at the parametrised posteriors, it is obvious that the RealNVP parametrised posterior has different structure to the others, despite a fairly similar posterior. This is likely due to the increased complexity of the RealNVP flow compared to the simple planar and radial flows. Also of note is that the planar and radial parametrised posteriors are quite similar, both appearing almost as scaled (towards the prior) versions of the original posterior. It seems that the advantage flow models have over the standard one is that they can retain discriminability when contracting the parametrised posterior close to the prior. The variational sampling limits how well this can be done in the baseline model. For the flow based models, the sampling is performed in the posterior, which has high discriminability due to the spread out density.

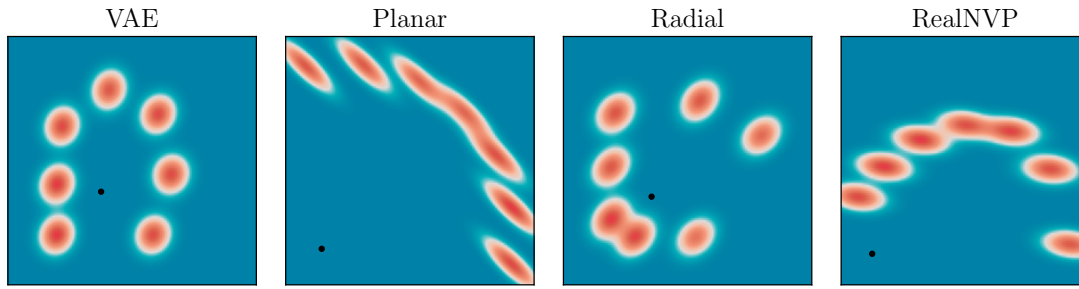


FIGURE 5.9: Comparing ForwardVAE posteriors on FlatLandN to those from flow based models.

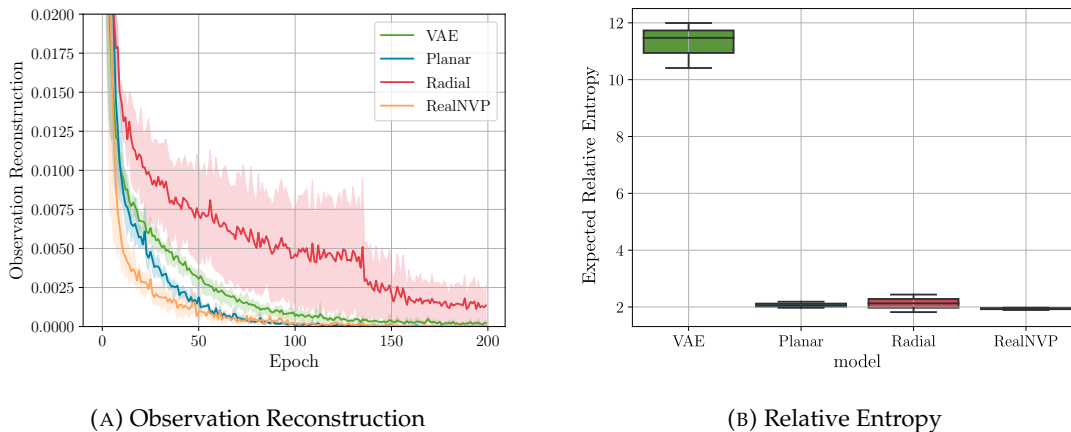


FIGURE 5.10: Exploring flow based models on FlatLandN, where there are no cycles in the data. (A) Validation observation reconstruction error over training. Lines indicate mean, shaded area indicates 1 standard deviation. (B) Boxplot of KL-Divergence error for the flow based models.

Learning the True Structure On FlatLand and FlatLandG, there is obvious true structure, we observe cycles in the data and the learnt structure reflects this. On FlatLandN however, there are (at least) two interpretations of the problem. We could consider it a problem of incomplete data, where there were cycles in the generative process and we were just unlucky to never observe them. Alternatively, we could say that since we never saw cycles, they were likely not present in the generative process, and so any learnt structure should not have cycles. Whilst in this instance we have access to the generative process, that may not always be the case, and FlatLandN could equally have come from a generative process with cycles.

In Fig. 5.9 we show example posteriors on FlatLandN which came from a standard ForwardVAE ('VAE') and flow based ForwardVAE models (the rest). Notice that the flow based models do not learn the same structure as the standard model. The Radial flow model is the closest, and it learns almost cyclic structure with some degree of alignment to the Cartesian axes, as we saw in Fig. 5.4. The Planar flow and RealNVP models do not learn cyclic behaviour, instead learning non-cyclic structures similar to those we saw on FlatLandN when we ablated the KL term (Fig. 5.3).

The two different behaviours are akin to the two scenarios of incomplete data and non-cyclic symmetries. The standard prior learns the simplest distribution it can, which in this case is a cyclic structure, and this minimises the KL by increasing posterior overlap. The flow models learn a more (spatially) complex posterior (with respect to the standard prior), which allows them to model the observed structure since they don't have simplifying pressure from the prior. So if we thought we had incomplete data, we would want to learn cycles and would use the standard prior. If we thought we had non-cyclic symmetries, then we'd want to learn non-cyclic latent structure and thus use flows. This is muddled by the fact that under the post action observation reconstruction error (Fig. 5.10a), there is very little difference between the flows and the standard model. Furthermore, both encodings are still very efficient, giving similar KL divergences (see Fig. 5.10b) to those we saw on the standard FlatLand (Fig. 5.8a).

In auto-encoders, the goal is to encode an image, so we generally prefer the simplest solution to a given problem - and consequently should use the standard prior. For learning symmetries acting on the data, it is not clear if we should aim for the simplest solution or the one that best fits the observed data. Depending on the true generative process and the particular observations, either one may fail to find the true symmetries.

Flow Complexity During the previous experiments, we noticed (anecdotally) that deeper flows (more flow layers) sometimes performed worse than shallower, simpler ones. As such we investigate the effect of flow complexity (defined by the number of flow layers) on the learnt representations of FlatLandG. In Fig. 5.11a we can see that Radial flows clearly perform best at 10 flow layers, with significantly worse encodings at higher or lower flow depths. Radial flow has shown to perform inconsistently in this and previous experiments, so we will discard this result, possibly there is an implementational bug or the Radial flow is just not suited for FlatLand problems. The Planar flow shows the encoding improving as we increase the complexity, and the most efficient encodings achieved with a flow depth around 10. Notice that with only 1 flow layer, the coding efficiency of the Planar and Radial flows are equivalent to those of a standard ForwardVAE (cf. Fig. 5.8a). RealNVP, which is inherently more complex than the other flows, shows very little difference in latent code efficiency as we increase the number of layers. Both the Planar and RealNVP flows showed the best performance at 10 flow layers, with (marginal) reduction in encoding efficiency with subsequent added layers.

Whilst RealNVP shows good code efficiency with few flow layers, from Fig. 5.11b we can see that its ability to accurately model the data is reduced. Notice that all models perform comparatively poorly and show low consistency with less than 5 flow layers. Once the flow is sufficiently complex, the performance of Planar flow and RealNVP

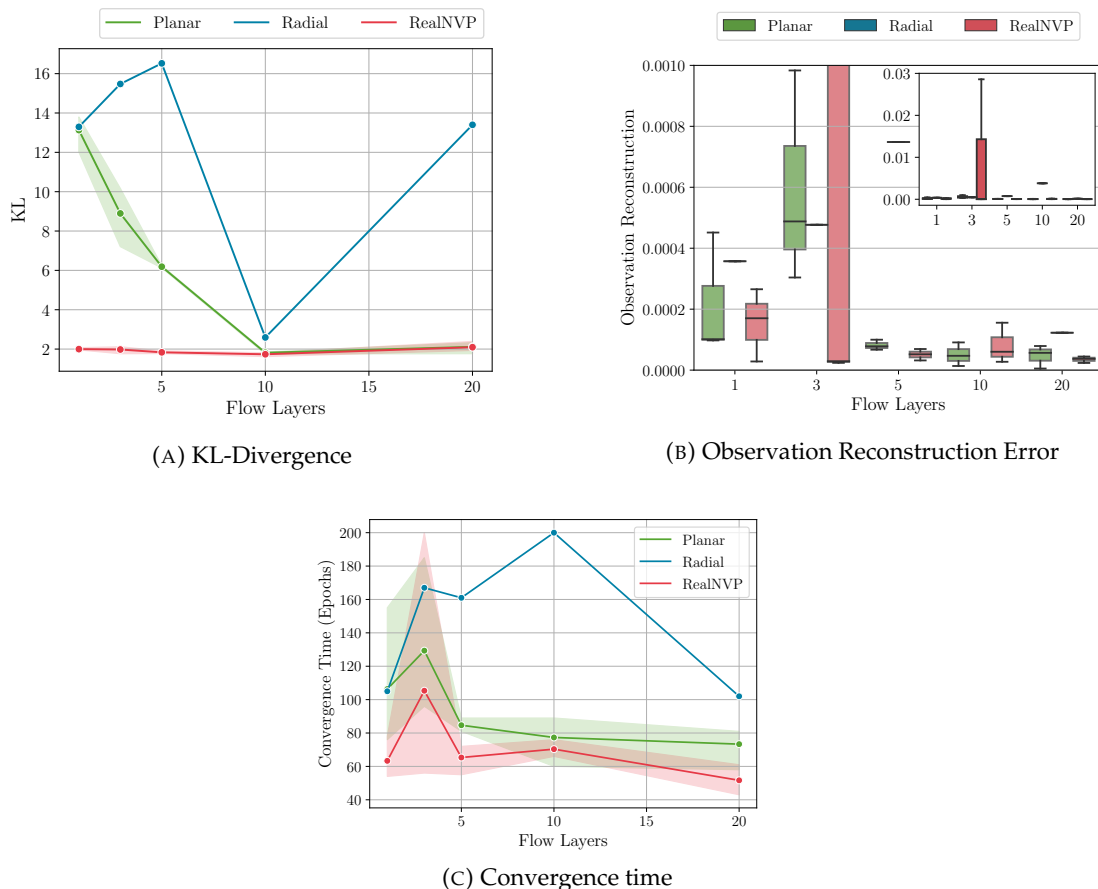


FIGURE 5.11: Flow Complexity on FlatLandG. (A) KL-Divergence error for models with different depth flows. (B) Observation reconstruction for different depth flows. (C) Convergence times for models with different depth flows.

does not significantly change with additional flow layers. From these results, we can conclude that with sufficient number of layers, the flow depth could be optimised for maximum performance, but the improvement is not significant and likely unnecessary.

Whilst performance might not be harmed by overly deep flows, depth may impact convergence time - more parameters to optimise (slower) or adequate complexity to quickly learn the optimal map (faster). In Fig. 5.11c we estimate convergence time as the number of epochs until a post action observation reconstruction error of 0.001 is achieved. We can see that for few flow layers, the convergence is slow and inconsistent, however for deeper flows the convergence time is fairly constant. Furthermore, the convergence rate does seem to be reduced with depth, however the improvement is again marginal.

5.2 Distinguishing Linear Disentangled Representations

Given the highly specific structure of the flatland linear disentangled representations, we expect to see evidence of them in disentanglement metrics. In this section we will explore whether a classifier can distinguish a linear disentangled representation from a classically disentangled representation purely from the disentanglement metric scores. This section will aim to answer the third question posed in the chapter introduction, “Can we find evidence of LDRs in disentanglement metrics?”.

Experimental Setup For this section we will need a dataset consisting of disentanglement metric scores from a number of linearly and classically disentangled models.

For FlatLand, we reuse the models detailed in Section 4.5 with additional RGrVAE instances, whilst for dSprites, we trained 18 models, the training scheme for which are detailed in Appendix Section C.2. For both datasets, we then computed disentanglement metric scores for each model on a 10% validation set held out from the training dataset. We then create a number of datasets which each consist of an individual metric score for each model instance, and an associated label which says if the representation of that model instance is linearly disentangled or not. We set these labels by hand, and assumed that ForwardVAE and RGrVAE instance were linearly disentangled, and then decided based on observation if the other model instances had linearly disentangled representations or not. This was mostly done by observing orbits of actions obtained by the process outlined in Section 4.5. For FlatLand, we had 6 linearly disentangled representations and 18 classically disentangled representations. For dSprites, we had 3 linearly disentangled representations and 15 classically disentangled representations.

Given this setup, we then train an SVM for each metric, using the default Scikit-Learn SVM settings, barring the class weighting which was set to balanced.

Disentanglement Metrics For this experiment, we compute the metrics detailed in Section 3.2.4 (Classic), the unsupervised metrics² used by [Locatello et al. \[2018\]](#) (Unsupervised), the metrics detailed in Section 4.4 (Factor Leakage and Symmetry), and finally, some additional quantities computed from the representation (Symmetry). In Table 5.6 we provide a summary on each of these metrics.

Evaluation Metrics To determine how well the SVMs can distinguish linearly and classically disentangled representations from the disentanglement metric scores, we

²Modified from: https://github.com/google-research/disentanglement_lib

Metric	Description
Val Beta	BetaVAE metric, see Section 3.2.4.
MIG	Mutual information gap metric, see Section 3.2.4
Informativeness	Metric by Section 3.2.4
DCI	DCI Disentanglement metric, see Section 3.2.4
Completeness	See Section 3.2.4
Modularity	See Section 3.2.4
Explicitness	See Section 3.2.4
SAP	See Section 3.2.4
GaussTC	Total correlation of the latent code dimensions based on fitted Gaussian. See Locatello et al. [2018] .
Wass Corr	Wasserstein correlation of the latent code dimensions based on fitted Gaussian. See Locatello et al. [2018] .
Wass Corr Norm	Normalised version of Wass Corr.
MI Score	Average mutual information between latent dimensions. See Locatello et al. [2018] .
FL Mean	Factor Leakage metric. See 4.4
FL Norm Mean	FL Mean normalised.
FL	Average over i for the factor leakage computed ignoring the i most informative latent dimensions.
FL NM	FL where the first most informative latent dimension is ignored.
Downstream Rep	Accuracy of a gradient boosted tree when trying to predict an observed action from the pre and post action latent codes.
True Indep	True independence. See Section 4.4.
Av Symmetry L1	Average difference between the known irreducible representation matrix and the learnt representation matrix for the representation (uses the methods from Section 4.5).
Av Rep Mean x2	Average observation reconstruction error.
Av Rep Mean z2	Average latent reconstruction error between predicted and true post action latent codes.
SVD Overlap	See Section 4.4.

TABLE 5.6: Disentanglement metric summary.

need evaluation metrics. Using the SVM classifiers, we can report the accuracy ‘Acc’ and the probabilities that the classifier predicts a linearly disentangled representation to be classically disentangled ($P[L|N]$) and visa-versa ($P[N|L]$). In addition to these, we also report the ROC-AUC, Precision-Recall area under curve (PR) and F1 scores. Since we are working with single (continuously valued) disentanglement metrics, we can define a simple classifier by just setting a threshold, for example, we could say that all model instances with a MIG greater than 0.1 are linearly disentangled. This allows us to compute the ROC-AUC, PR and F1 scores by varying this threshold.

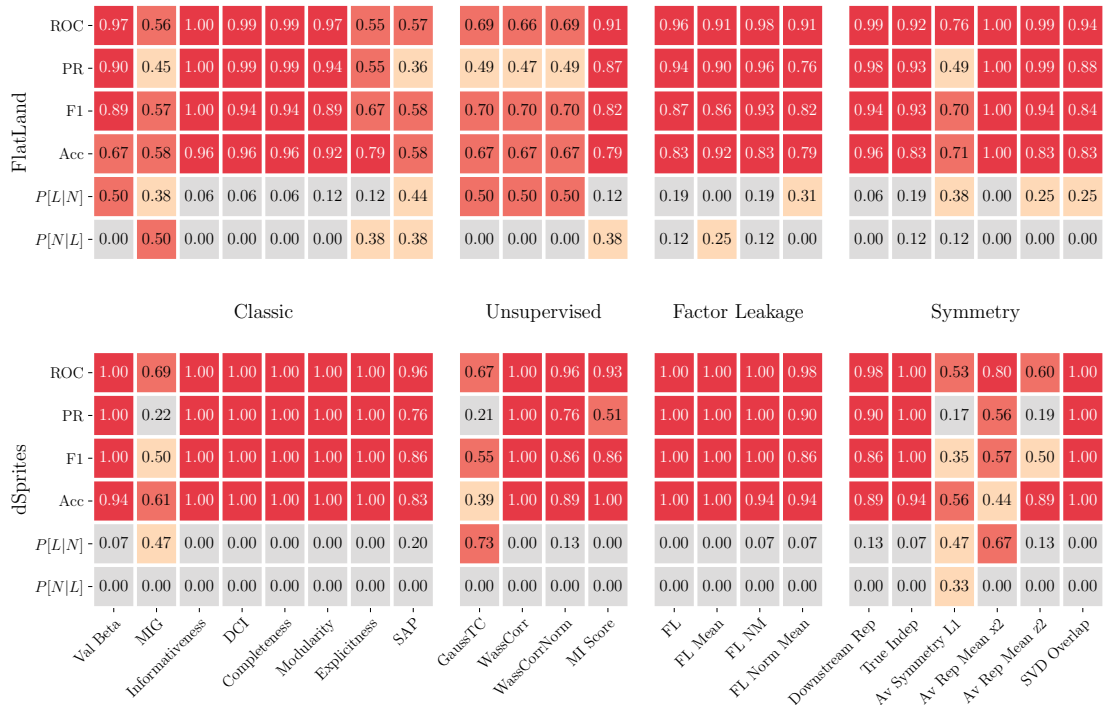


FIGURE 5.12: Relevance of each metric to distinguishing linear from classical disentanglement. ROC and PR are computed directly on the metric values and given as AUC measures. Scores are the probability that an RBF-SVM can classify the model correctly from the metric alone. $P[L|N]$ is the probability the SVM predicted a model was (L)inear disentangled when it was (N)ot/(N)ormal.

Results In Figure 5.12, the first thing to notice is that whilst on average, discriminability of each metric is similar across both datasets, they do not always agree. We attribute this to the comparative complexities of the dataset. FlatLand is very simple and models converge rapidly, resulting in very optimised representations. dSprites is much more complex and it takes significantly longer for representations to converge fully. Whilst we increased training length dramatically to account for this, models may still have rather ‘noisy’ representations - i.e. not optimised to their maximum potential. This is particularly true for RGrVAE, since representation convergence depends on the policy having first converged.

Despite this observation, we can still draw some conclusions from Fig. 5.12. We can see that, surprisingly, some of the best performing metrics are classical ones, with Informativeness, DCI Disentanglement, Completeness and Modularity all achieving high scores. This is true for all estimates, and across both datasets. This is interesting since of the four, two of them (Modularity and Disentanglement) depend on individual latent dimensions encoding information about a single factor, a property which you would expect to be strong in both forms of models. We know that linear disentangled models are strict on this property since it is similar to the independence. They also likely have cleaner (better disentangled) representations than standard VAEs, which was somewhat evident in Fig. 4.21. Of the other two metrics,

informativeness is computed as the ability to predict factors from latent values, a form of downstream task linear disentangled models are particularly good at, as we will see later. Finally, Completeness measures the extent single generative factors are encoded into a single latent. We know that for both datasets, all symmetries are cyclic and so each factor will be encoded across two latents. Since standard disentanglement would encode them across one dimension each, this allows the metric to be discriminative. This, however would not extend to data where there are symmetries acting with 1D irreducible representations. Whilst we have no such examples in our datasets, it is important to note that Completeness may not be discriminative in all cases.

The other classical metrics are less discriminative. The BetaVAE metric allows high scores for both classical and linear disentangled representations. The MIG and SAP metrics have already been discussed in previous sections. The explicitness is interesting, since it relates to the ability to determine generative factors from the latent code, seemingly similar to the Informativeness, however the major difference is that Explicitness is computed from the entire latent code, whereas the Informativeness is computed from individual dimensions. Across the whole code, it is likely that both symmetry based and classical models can predict the generative factors accurately, however symmetry based models may have the advantage when considering only single dimensions due to their highly structured posteriors.

We now consider unsupervised metrics. In general they are not consistently discriminative, with the possible exception of the MI score. We do note that the dSprites PR-AUC is fairly low, whilst the F1 score is relatively high. Possibly this is due to a small number of anomalous results prompting low precision values in the PR curve. Such anomalies mean we cannot say conclusively if this metric would be discriminative for larger sample sizes.

The third class of metrics are Factor Leakage derivatives. We consider the area under the curve (AUC), the mean (Mean), the AUC whilst ignoring the most informative latent (NM AUC) and finally the mean of the normalised curve (Norm Mean). In general all the FL metrics are fairly discriminative on both datasets, with particularly high scores on dSprites. The best overall variant appears to be the non-maximum AUC variant, at least under ROC, PR and F1 scores. The worst appears to be the norm mean variant. Since there is not much difference between the other three, we will generally consider one of the two AUC variants.

The final class of metrics (symmetry based) are a mix of metrics which we have proposed and quantities related to symmetry based models. First note that the Independence and SVD metrics are fairly discriminative across both datasets and as we might expect, they perform similarly. Neither the symmetry reconstruction or post action observation/latent errors are particularly discriminative on dSprites. This is likely due to lower quality representations, as we mentioned at the beginning of this

section. Whilst only the symmetry reconstruction performs poorly on FlatLand, we cannot be certain if these are good metrics for determining whether a model is linear disentangled or not. Since these quantities are all estimated through the process outlined in Chapter 4 (Section 4.5), it is possible that the representations learnt (independent of the models) were not well optimised due to multiple bad local optima in the loss space. We note this since the learnt (internal representations) on FlatLand were generally more accurate than those estimated independent of the model. Whilst we could use the internal representation to directly compute independence, we could only do this for the symmetry based models, so the comparison to classical metrics would be impacted by any biases in the estimation procedure.

5.3 Downstream Tasks

Strong performance in downstream tasks is the goal of most unsupervised representation learners. For disentanglement research, a simple and common downstream task is to predict generative factor values from the representation [Locatello et al., 2018].

Since Caselles-Dupré et al. [2019] showed that symmetry based models require interaction with the environment (i.e. observe actions / state transitions), they evaluated on the downstream task of predicting which action occurred between two sequential observations. We will compare symmetry based and classical models on both the downstream tasks of factor and action prediction, to determine if and how much improvement linear disentangled representations offer over classical ones. We will also confirm the findings of Caselles-Dupré et al. [2019] where it was shown that linear disentangled representations perform better on the task of predicting actions. This section will aim to answer the downstream task portion of fourth question posed in the chapter introduction, “How do LDRs compare to classically disentangled representations under data efficiency and downstream tasks?”

Experimental Setup In this section we explore two downstream tasks. In both cases, we use the same training parameters as detailed in Appendix Section C. We train 5 instances of each model type and report results using these on a 10% held out validation set. We use the same backbone architectures as in Section 4.5.

To measure performance on downstream tasks we use either a gradient boosted tree (GBT) or random forest classifier (Forest). The GBT experimental setup uses that defined by Locatello et al. [2018], whilst the Forest experimental procedure follows Caselles-Dupré et al. [2019]. Both classifiers use the sklearn implementation with default settings.

Predicting Actions For this task, the specifics of the experiment are as follows. For this task we need to work with the transition versions of FlatLand and dSprites as defined in Section 3.3.1. We take pretrained VAE models, and compute the latent representations (take the mean of the encoder output) of each data point, pre and post action. This gives us a dataset of pre-action representations, post action representations and the action label, for each VAE model instance. In this task, we train the classifiers to then predict the action label from the concatenation of pre and post action latent representation codes.

Caselles-Dupré et al. [2019] explored the question of whether linear disentangled representations are “increasingly better” for downstream tasks when compared to standard disentangled representations. To do this they evaluated this downstream task of predicting observed actions (using Random Forests) under limitations of classifier complexity and dataset size. Specifically, they limited the depth of a tree based classifier (classifier complexity) or limited the number of samples in the dataset (Dataset Size). They found that given a large enough dataset and a classifier of sufficient complexity, there is little difference between symmetry based and classical models on task. However, when the classifier complexity or dataset size (or both) are limited, the linear disentangled representations consistently allow for stronger downstream performance.

We first compare the absolute performance of symmetry based models to baselines. We find that the symmetry based models allow for better downstream scores on both datasets (Fig. 5.13a and Fig. 5.13b). On FlatLand (Fig. 5.13a), all models achieve very high scores, however ForwardVAE and RGrVAE are higher scoring. On dSprites (Fig. 5.13b), the baselines show highly variable scores, although TcBetaVAE approaches the same score as RGrVAE, with RGrVAE performing better on average. These results suggest that well trained classical models can perform well on this task, however symmetry based models still have an advantage.

We also consider restricting dataset size and classifier complexity, with an additional comparison to the RGrVAE policy distribution. The policy distribution is (under ideal conditions) a labelling of exactly which action has been observed, which is naturally a perfect representation for this particular downstream task. We can see that when provided a large enough dataset (Fig. 5.13c), the policy is a perfect classifier by itself - since the forest classifier has 100% accuracy at a depth of 1, and it simply maps policy dimensions to action labels. When data is limited (Fig. 5.13d), the policy performance becomes less consistent and based on the wide variance at dataset fraction 0.7, likely depended quite heavily on the selected data being representative of enough states. We do note that on average it achieved accuracy greater than 95%. In general, the policy will not be a particularly strong representation on its own for most tasks, however for this particular task and any that require knowledge of the action observed, it is a natural choice.

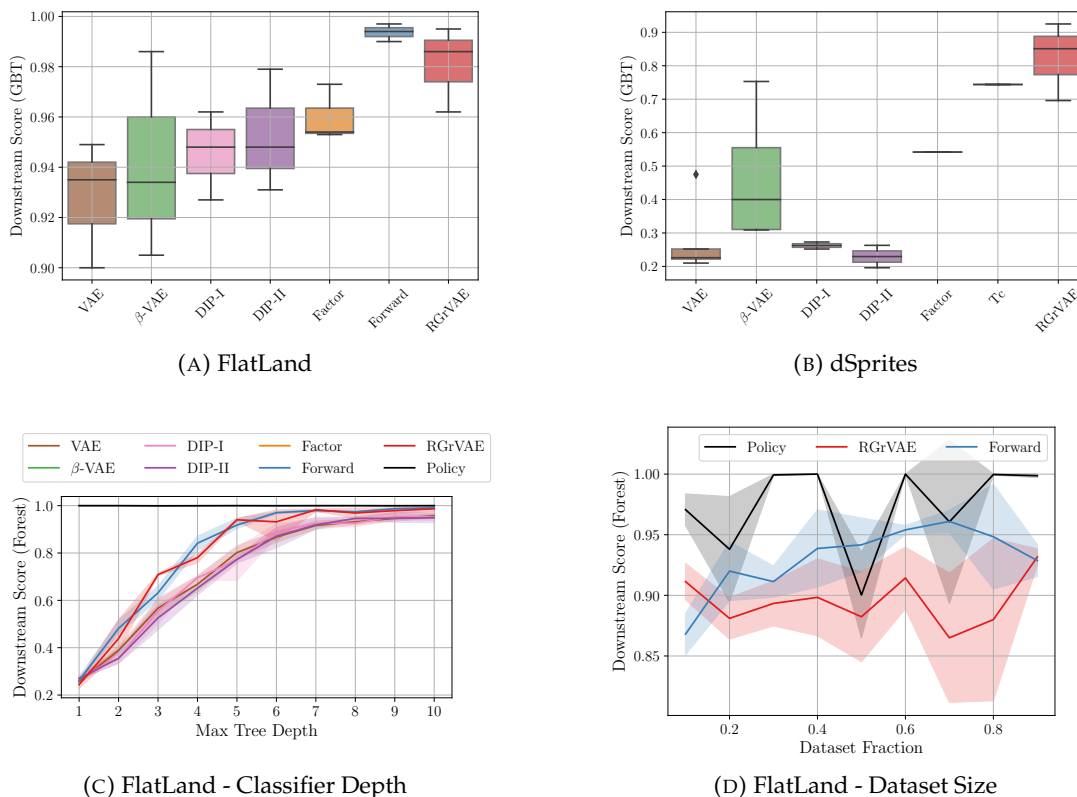


FIGURE 5.13: FlatLand and dSprites Downstream performance on action prediction for all baselines.

Returning to Fig. 5.13c, we can see that the symmetry based models (red and blue) allowed for better downstream performance with lower complexity classifiers, when compared to baseline VAEs. This is consistent with the findings of Caselles-Dupré et al. [2019], and shows that linear disentangled representations are beneficial for this task. When comparing RGrVAE to ForwardVAE, we see that in both cases of reduced data and reduced classifier complexity, performance is almost identical. This further shows that RGrVAE learns structurally the same representations as ForwardVAE when trained sufficiently, despite the lack of action supervision.

Note that for the complexity and dataset size experiments we have used Random Forest Classifiers rather than the Gradient Boosted Tree (GBT) classifiers used for computing the general downstream performance. We found that the GBT classifier complexity was mainly decided by the number of iterators, rather than the depth of the individual trees in the ensemble. For this reason, we chose to measure performance under Random Forests, which allowed us to vary classifier complexity by depth. This has the further advantaged of being consistent with the experimental procedure of Caselles-Dupré et al. [2019].

Predicting Factors Predicting factors is a standard downstream task in disentanglement research. This task does not require datasets of transitions, just the

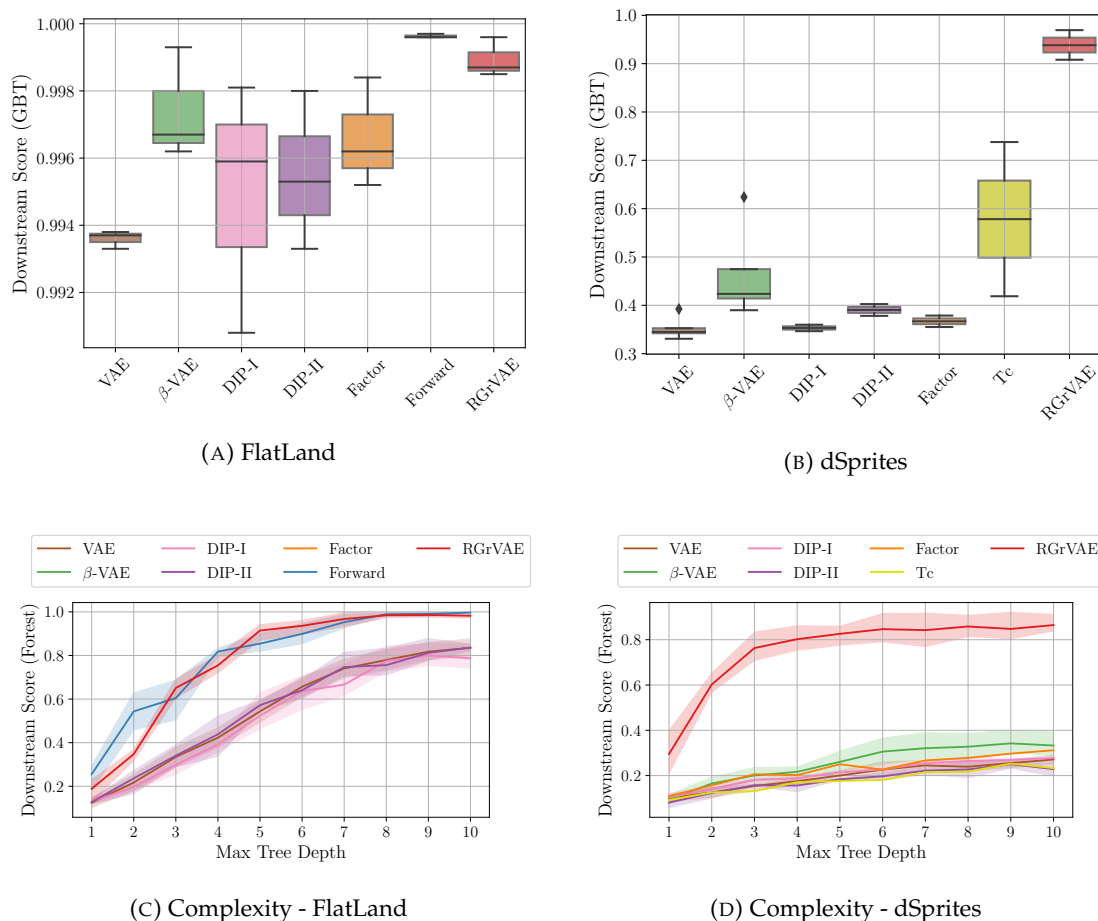


FIGURE 5.14: Generative factor prediction downstream performance for baselines and symmetry based models on FlatLand and dSprites using a Gradient Boosted Tree (GBT) classifier. Symmetry based models perform better on both datasets in this task.

base datasets. To evaluate this task, we compute the latent representation of each point in the dataset, and store the generative factor values (See Section 3.2.1) as the targets. The classifier is then tasked with predicting the exact generative factor values from the latent representation values for each state.

Locatello et al. [2018] found that disentangled representations correlated with improved performance on this downstream task for most datasets. We now compare our models under this task. We follow the same experimental procedure as Locatello et al. [2018], using a Gradient Boosted Tree (GBT) and default sklearn settings. We also note that, similar to Locatello et al. [2018], we found that the dSprites baselines had not consistently converged to disentangled representations however the majority appeared to do so.

Beginning with FlatLand (Fig. 5.14a), we can see that the models perform very well on this task, all achieving greater than 90% classification accuracy. The symmetry based models achieve consistently higher scores ($> 98\%$) with much more consistency than the baseline VAEs. Whilst for FlatLand we only see fractions of a percent

improvement, this is a very simple dataset, so it is to be expected that the improvements would be marginal. In Fig. 5.14b (right) we evaluate on dSprites, where the distinction between symmetry based models and classical models is much more distinct. The classical models achieve between 30 – 60% accuracy, whereas the symmetry based model achieves greater than 90%. When comparing the performance under the limitation of classifier complexity, we can see that similar to the previous task, symmetry based models allow for better performance at lower complexities.

It is interesting that RGrVAE achieved similar dSprites performance on the factor classification task as it did on action classification. On dSprites, there are 4 actions (we did not consider changing shape an action) and the rotation and translation factors have > 32 possible values. You would expect that models would perform significantly worse at classifying factor values than they would at classifying actions.

Symmetry Based Models on Downstream Tasks In both of the evaluated downstream tasks, symmetry based models performed the strongest in pure performance and when restricted by complexity. For the task of predicting observed actions, there is a strong case to be made that the symmetry based models have an inherent advantage. ForwardVAE is explicitly supervised by action labelling and RGrVAE has implicit bias since it observes the results of single actions, and not a mixture of actions. The baseline VAEs by comparison are completely unsupervised and have no concept of actions during training or evaluation. This advantage could be the reason symmetry based models perform better on this task. The same cannot be said of the factor prediction task. During training, neither ForwardVAE or RGrVAE have any knowledge of the generative factors, other than the fact that they only ever observe actions on single factors. Whilst it may be true this could provide symmetry based models cleaner gradients towards separating factors in the latent space, this does not make predicting the full set of generative factors any easier. However, we have seen that linear disentangled representations have significant posterior structure, which offers rich information about the generative factor values. The consistency in the learnt representations also allows for consistency in downstream task performance, although more so in ForwardVAE due to supervision. Consistency is not always a given with VAE models, as can be seen by the disentanglement metrics and downstream scores. This may be considered another advantage of symmetry based models, a single instance will often provide the best performance of that model class, whereas with a standard VAE, you may have to train/evaluate multiple instances to achieve top performance.

5.4 Data Efficiency

Whilst [Locatello et al. \[2018\]](#) found that disentanglement metrics correlated (on most datasets) with factor prediction performance, they also found that there was no consistent correlation with data efficiency. This is important as it has been cited as a potential benefit for learning disentangled representations in the past [[Bengio et al., 2013](#)]. Data efficiency (aka sample complexity) of a representation, can be thought of as how easy it is to learn downstream tasks from the representation. [Locatello et al. \[2018\]](#) measure this by evaluating a classifier which has been trained on a restricted number of samples from the representation. Specifically, for the task of generative factor prediction, they evaluated the performance (classification accuracy) conditioned on 100 samples divided by the performance conditioned on 10000 samples.

We will look to answer the following two questions: are symmetry based models more data efficient, and can we correlate symmetry based metrics with efficiency? Towards this, we evaluate the data efficiency on FlatLand and dSprites for symmetry based models and baselines. Since we evaluated two downstream tasks in the previous section, we will continue to compare them under efficiency.

Experimental Setup All data efficiency experiments use a GBT classifier using default Scikit-Learn parameters. For each downstream task, the classifier accuracy was evaluated using 10, 100, 1000 and 10000 data points, and data efficiency was reported as the ratio of the performance using 10000 data points to that using 100 data points. The backbones are the same trained backbone instances as used in the downstream score experiments.

Are symmetry based models more efficient? First discussing the factor prediction task, we can see from Fig. 5.15a that on FlatLand, the symmetry based models have a clear advantage in efficiency. On dSprites (Fig. 5.15b), RGrVAE shows a higher median efficiency than the classical models, albeit the improvement is far less distinct than on FlatLand. Indeed the TcBetaVAE upper quartile is close to the median RGrVAE score. Also of note is FactorVAE, which shows strong scores, approaching those of RGrVAE. In general, we can say that the symmetry based models achieve higher efficiency scores on this task. Indeed this will be reinforced when we consider an efficiency score normalised by the downstream performance of the representation (Fig. 5.17).

For action prediction task, the comparison between symmetry and classical models is not clear. For FlatLand (Fig. 5.15c) we see a similar results to the previous task, although RGrVAE shows an extremely large variance. For dSprites (Fig. 5.15d), the classical models show far higher efficiencies than the symmetry based models, which have surprisingly low efficiency scores. Upon examining the downstream scores for

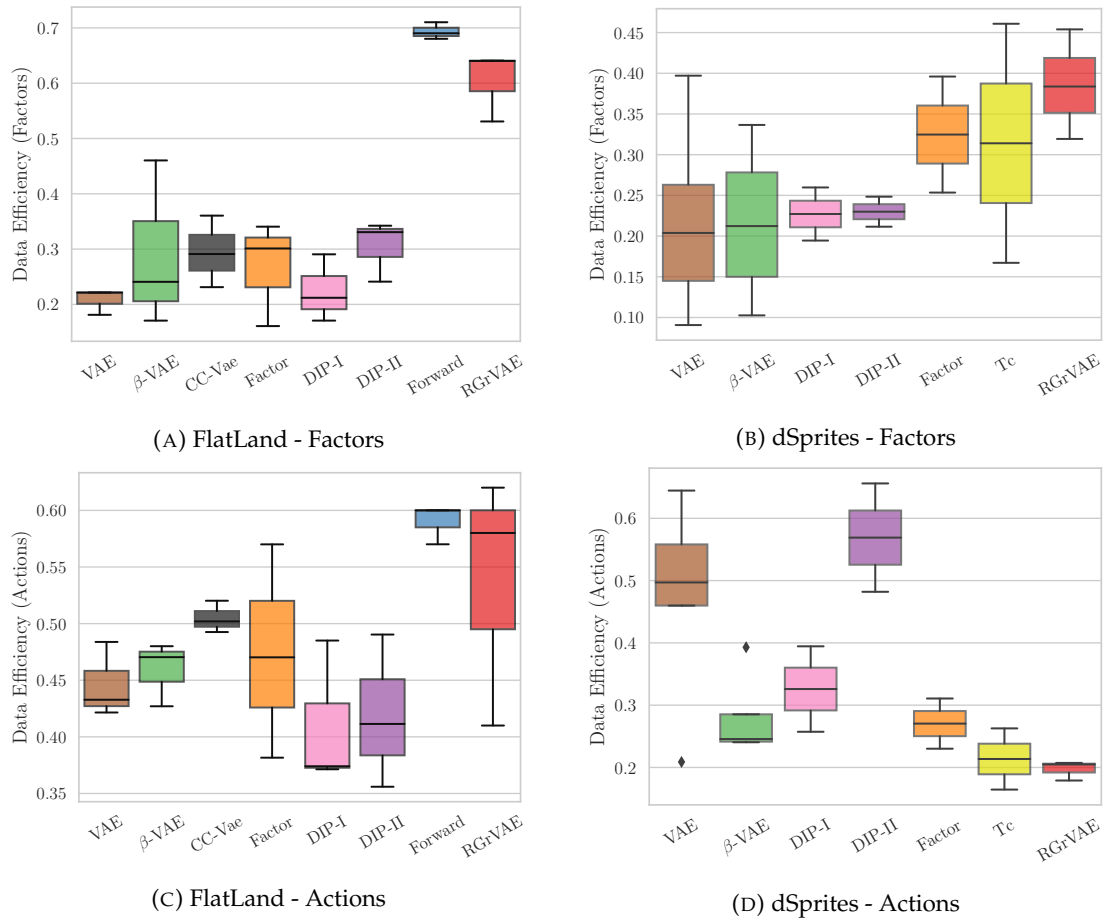


FIGURE 5.15: Data Efficiency Scores

this task (recall Fig. 5.13b), we can see that the classical models achieve very low accuracies. Since the performance conditioned on 100 samples is very similar to that on 10000 samples, the efficiency is high, despite the performance being poor. This can be seen in Fig. 5.16a which scatters the mean efficiency against the mean downstream score for dSprites. Notice that, if we disregard the RGrVAE policy, there is an inverse relationship between the two, higher efficiencies seem to correlate with lower downstream scores. This downstream task appears to be very hard to solve with access to small amounts of data, and even the highly structured RGrVAE representations cannot achieve strong scores in this domain. We can see this in Fig. 5.16b which plots the downstream score with increasing data. The classical models do not improve significantly as the available data increases, whereas RGrVAE has a large jump between 100 samples and 10000, resulting in the data efficiencies shown in Fig. 5.15d. Notice that all models still appear to be improving at the 10000 data size. This suggests that we could achieve higher downstream scores if we included more data on which to train. However, for consistency with [Locatello et al. \[2018\]](#), we shall stick to the scheme of 10000/100 data points. Of course, as we can see from Fig. 5.16a, none of this analysis includes the policy representation, which

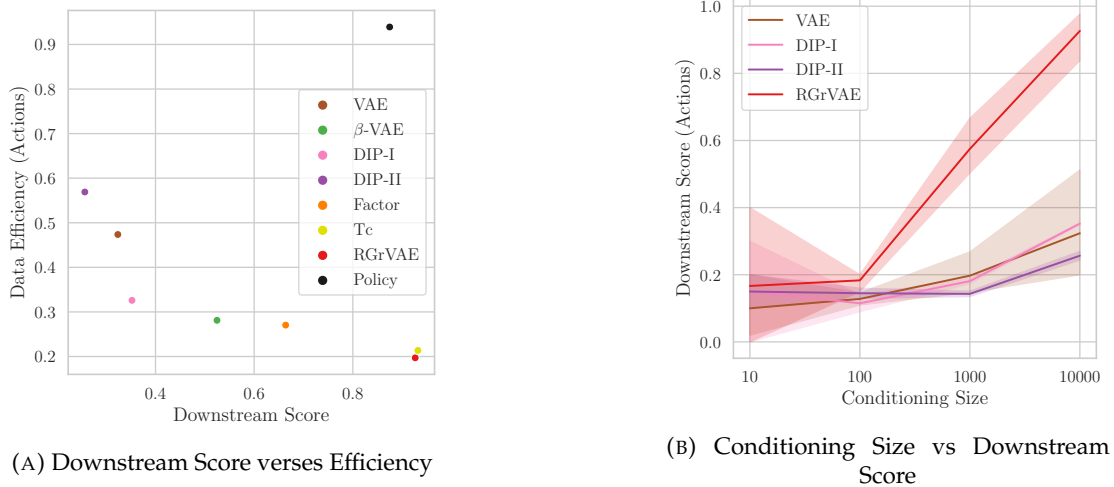


FIGURE 5.16: Downstream Score Comparisons

achieves very high downstream scores and data efficiency - since it is an extremely good representation for this task.

Given that the data efficiency can be heavily impacted by the final downstream performance, we now compute a ‘Scaled Efficiency’. We take into account the overall quality of the representation by multiplying the efficiency by the downstream score, through which we arrive at the ‘Scaled Efficiency’ score reported in Fig. 5.17. From Figures. 5.17a and 5.17b, we continue to see symmetry based models achieving higher data efficiencies on both datasets for the Factor prediction task. Indeed the improvement over classical models is increased compared to Fig. 5.15. On the action prediction task the story is still not clear. On FlatLand (Fig. 5.17c) we can see ForwardVAE achieves very strong scores, however RGrVAE continues to show high variance. This could be due to one low quality representation degrading the results. On dSprites (Fig. 5.17d), we see that RGrVAE still does not offer the best scaled efficiency, with an almost identical median to TcBetaVAE, but a much worse upper quartile. The proximity of the median and upper quartile suggests a single representation lowering the score. However, since the median is the same as TcBetaVAE, it is unlikely that we would see significantly higher scaled efficiency even with retrained models.

From the results so far, we can say that for Factor prediction, symmetry based models are more efficient than classical models on the datasets tested. Whilst dSprites showed Tc, Factor and RGrVAE having overlapping scaled efficiency distributions, the median of the latter was much higher. This suggests it’s unlikely that running more tests would show classical models being consistently more efficient than RGrVAE. This does not preclude single instances of (for example) TcBetaVAE being more efficient than an RGrVAE instance, however in general this outcome is unlikely. When we consider the action prediction task, the results are not as conclusive. Under the

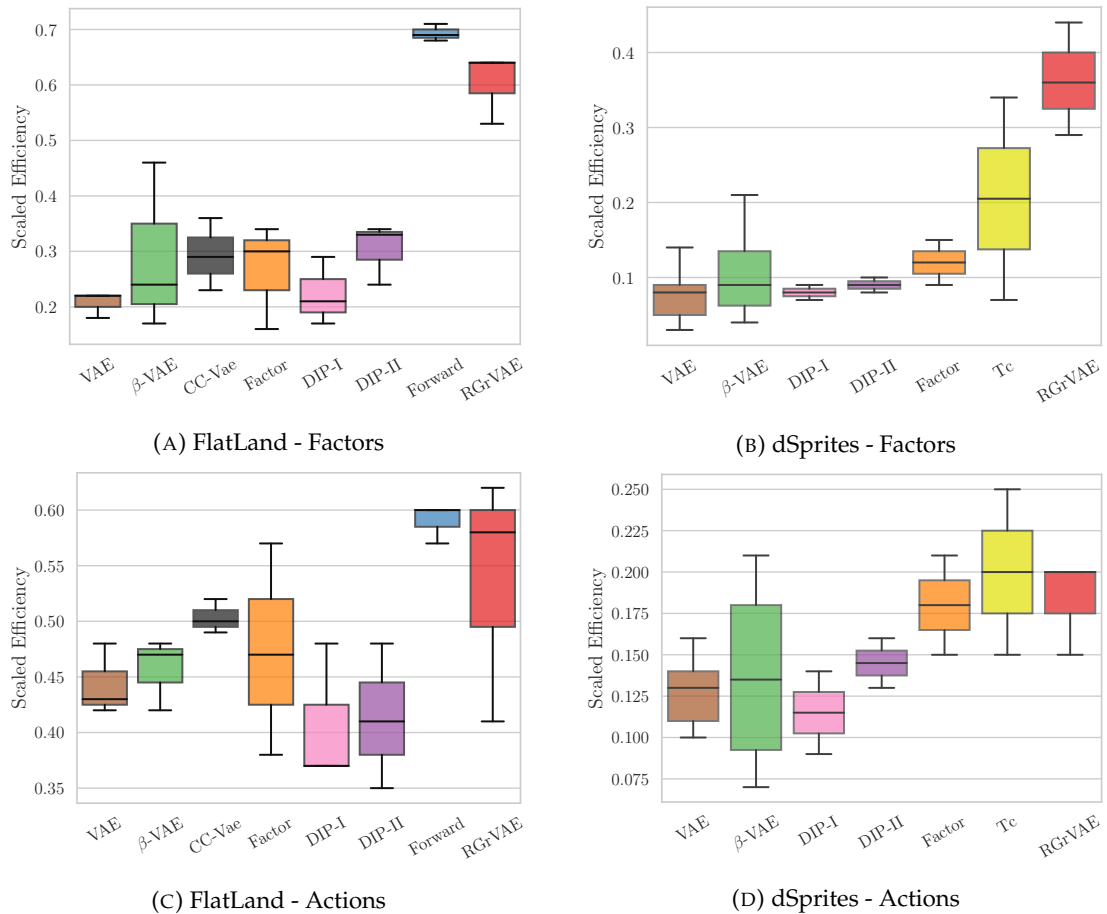


FIGURE 5.17: Scaled Data Efficiencies.

standard data efficiency, ForwardVAE shows consistently high efficiency on FlatLand, whilst RGrVAE shows high efficiency on average with a large tail. Since in the ideal case RGrVAE should learn the same representation as ForwardVAE, this is likely due to a single RGrVAE instance performing poorly for this task. On dSprites, RGrVAE shows the lowest efficiency tested. This, however was due to the classical models being unable to learn the downstream task effectively, resulting in unrepresentative efficiency scores. When scaled by the downstream performance, we find that RGrVAE performs as well as the best classical model, but notably not better, if we disregard the policy as a representation in itself. From these results, we cannot conclude that symmetry based models are more efficient on this task. It is interesting, however, that the action prediction task is so reliant on large amounts of data. For our testing, we concatenated the representations of the pre and post action observations as the representation for the action. It is possible that this is not particularly suitable for the classifiers tested and that a better way to combine the pre and post action representations would result in better performance with low data. We chose our method as the simplest available without introducing biases by performing operations on the data.

Do symmetry metrics correlate with efficiency? Arguably the most important result of [Locatello et al. \[2018\]](#) was that disentanglement metric scores do not consistently correlate with data efficiency across all datasets. Some metrics such as DCI disentanglement were found to correlate on, for example, Shapes3D, however there was no general trend across all tests. Accurately estimating correlations requires a dataset of large enough size to properly represent the general trends and not be susceptible to outliers. [Locatello et al. \[2018\]](#) evaluated over 12000 models, which should be sufficient to represent true trends (for the models tested). For our work, we cannot evaluate such a large number of models due to compute limitations, and will have to evaluate on the limited number models we have. As such, our correlations should be considered rough estimates, even if the reported p-values are low. This is particularly true for the dSprites results, for which we could train a very limited number of models. To mitigate this, for all the correlation figures, we will provide the Pearson correlation here and a scatter of the results in appendix A, so that correlations can be judged accordingly.

Fig. 5.18 reports correlation between metric scores and efficiency for both downstream tasks and datasets. First we consider the task of factor prediction. From Fig. 5.18a we find that on FlatLand we have strong correlation between the classical DCI metrics, as well as the Factor leakage metrics and the symmetry reconstruction metric. These correlations however, are much weaker on the dSprites dataset. The classical metrics show similar correlations on dSprites as those found by [Locatello et al. \[2018\]](#), bar the Modularity, which they found to be negatively correlated and we find to be positively correlated. These results are influenced by the presence of the symmetry based models however. We can see from Fig. 5.18b that when we remove the symmetry based models, the classical metric correlation on dSprites is reduced dramatically, and the p-value is heavily increased, indicating uncertainty. Interestingly, the Modularity continues to show some correlation. Similarly, the Factor Leakage metrics and the symmetry based metrics no longer show strong correlations once the symmetry based models are removed - on either FlatLand or dSprites. In general, we believe it is fair to say that if there is any correlation between any of the metrics tested and efficiency on this task, we would likely see stronger results than this. As such we believe that, as found by [Locatello et al. \[2018\]](#), any correlation is likely to be weak, and cannot be accurately estimated by our tests. In particular, there is no evidence that symmetry based metrics correlate any stronger with data efficiency than classical metrics.

Moving on to the action prediction task, we find that Figure 5.18c shows conflicting results between the two datasets. On FlatLand, the DCI metrics are generally positively correlated with data efficiency, whereas on dSprites they are negatively correlated. This is consistent both with and without symmetry based models. This relationship continues across the Factor Leakage and symmetry based metrics, with the direction of correlation inverted between the datasets. Recall from the previous

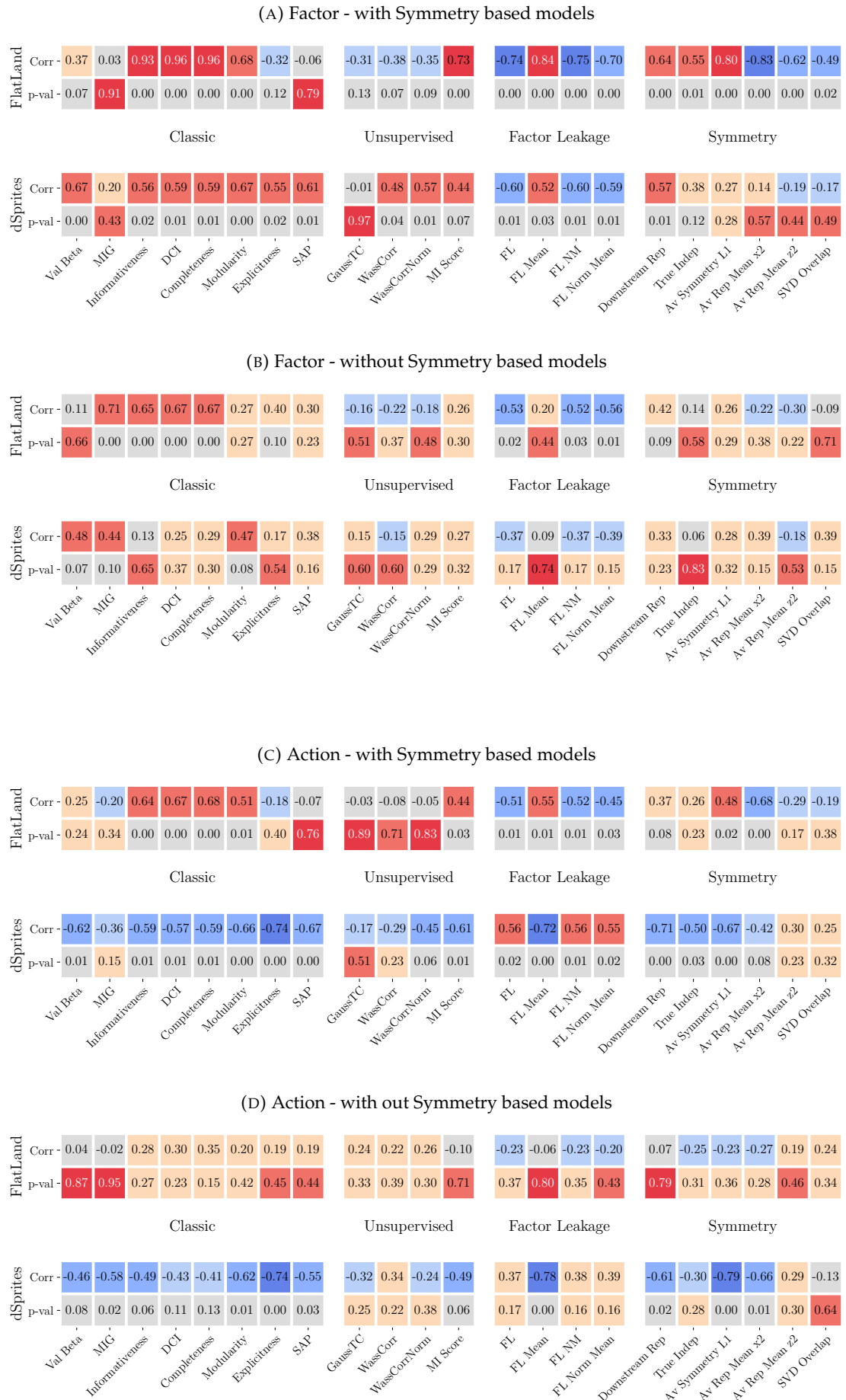


FIGURE 5.18: Correlating data efficiency scores with disentanglement metrics.

section that we considered a scaled data efficiency to compensate for poor performance on the task. In Fig. 5.19 we consider correlation of metrics with the scaled efficiency.

Under the scaled efficiency and the Factor prediction task, the correlations seen in Fig. 5.19a are simply stronger versions of those seen in Fig. 5.18a. Structurally, no changes are observed. We also see from Fig. 5.19b that the symmetry based models are very influential, where once again correlations are reduced when the symmetry based models are removed. Whilst the dSprites correlations are stronger (with and without symmetry based models) under the scaled efficiency, we still believe that we cannot conclude there is a true correlation, particularly since the DCI metrics are not strongly correlated in the absence of symmetry based models. Furthermore, we once again see that there is no evidence that symmetry based metrics correlate stronger than classical ones against the scaled data efficiency.

Returning to the action prediction task (Figures 5.19c,5.19d), we see a clearer picture than under the standard efficiency. In general the correlations have the same sign across both datasets. We again see fairly strong correlation with the DCI metrics on FlatLand and dSprites, although this is reduced on FlatLand when the symmetry based models are removed. Of the symmetry based metrics, those that measure independence (true independence and SVD overlap) are very uncertain in most cases, which precludes us from drawing any conclusions on them. It does appear however that the factor leakage metrics are consistent across the datasets (excepting FL Mean) in both Figures 5.18 and 5.19, suggesting there may be some true (small) negative correlation there.

From these results, we cannot provide a conclusive answer to the question ‘what is the correlation between symmetry metrics and efficiency’ due to lack of data. We can say that it is unlikely that symmetry based metrics correlate with efficiency any more than classical metrics. This is interesting since symmetry based models appear to have better downstream performance and efficiency than classical models. This suggests that none of the symmetry based metrics individually correlate with whether a model is symmetry based or not. For example, strongly disentangled classical models will likely have high independence - since they encode factors into single dimensions. Whilst this may appear to contradict the results in Section 5.2, we note that the correlation scores are computed based on the pair (metric, efficiency) for each model instance. As such, even if the symmetry based models had perfect efficiency and very distinct metric scores (neither of which are the case), the correlations would still depend on the scores of the classical models. It may be that some classical models show high metric scores and low efficiency, whereas others show the opposite, weakening the correlation. We highlight this by considering the correlation with and without the symmetry based models, and in general we did see reduction in correlation when the symmetry models were ablated. Whilst we have concluded that

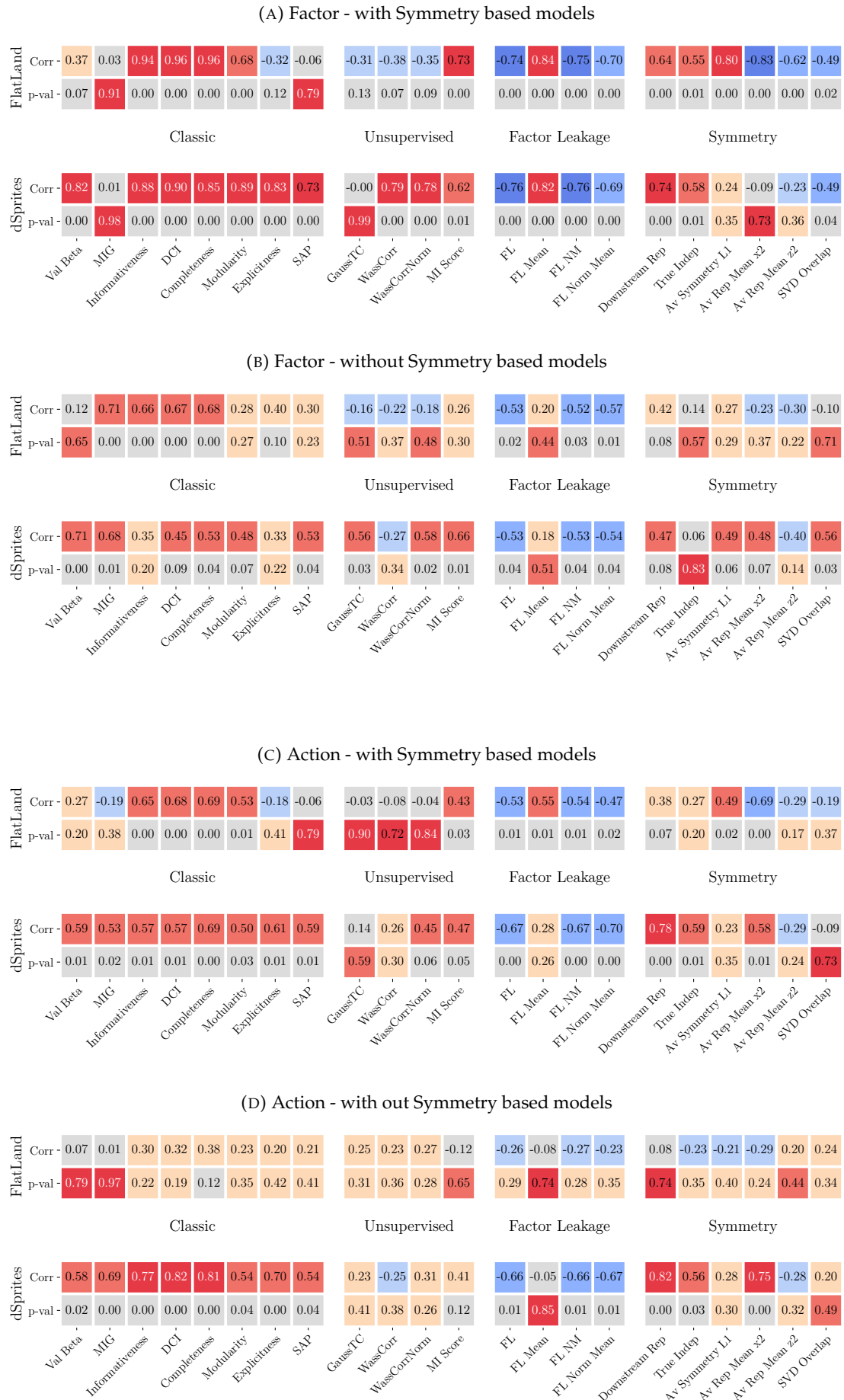


FIGURE 5.19: Correlating scaled data efficiency scores with disentanglement metrics.

there is likely no strong correlation with the symmetry based metrics and (scaled/standard) data efficiency, there are intriguing results for the factor leakage metrics, which, alongside a much larger study of the symmetry based metrics, would be an good starting point for future work.

Combined Data Efficiency For single metrics, we saw fairly high correlations in some cases, although our confidence is limited by the small amount of data. We will now consider correlation over both datasets to alleviate this problem somewhat. Subsequently, we will optimise linear combinations of metric scores for best data efficiency over both datasets.

Comparing Fig. 5.20a to Fig. 5.18a, we can see that the standard efficiency score continues to correlate fairly strongly with the DCI classical metrics and the Factor Leakage metrics. The major difference is that the unsupervised and symmetry based metrics now show very low correlations, with the exception of the MI Score, which correlated fairly strongly on both datasets. These observations hold true for both the efficiency and the scaled efficiency, although this is unsurprising since there is only a small discrepancy between them on this task.

For the action prediction task (Fig. 5.20b), the standard efficiency appears to correlate with very few metrics, only showing weak correlations with the FL mean and reconstruction symmetry metrics. This is likely due to dSprites efficiency results being strongly biased towards models which perform poorly on this task. When considering the scaled efficiency, we find the BetaVAE metric shows middling correlation and the Informativeness shows strong correlation, alongside the MI Score and the FL Mean metrics.

Considering both tasks together, we can see that the FL Mean metric seems to correlate quite strongly in all cases. The Informativeness and MI Score metrics correlate quite strongly in all but the dSprites action prediction task. Notably the symmetry metrics do not correlate consistently across tasks and neither do the Factor Leakage metrics beyond FL Mean.

Optimal Metric Combination for Data Efficiency Having considered single metric correlations, we now explore the optimal set of metrics, which, when combined linearly, have the maximal (Pearson) correlation with the data efficiency. To do this we simply gradient ascend a weight for each metric, optimising to maximise the final Pearson correlation coefficient. To find the minimal set of metrics, we introduce an L1 penalty on the weights such that many are set to 0. In addition to this, we apply a soft threshold to the weights (sharp sigmoid rising at 0.05) such that low weights (< 0.05) can be set to 0 in the final correlation computation, whilst still retaining gradient in the optimisation process. As the optimiser, we use Adam with a learning rate of 0.001.

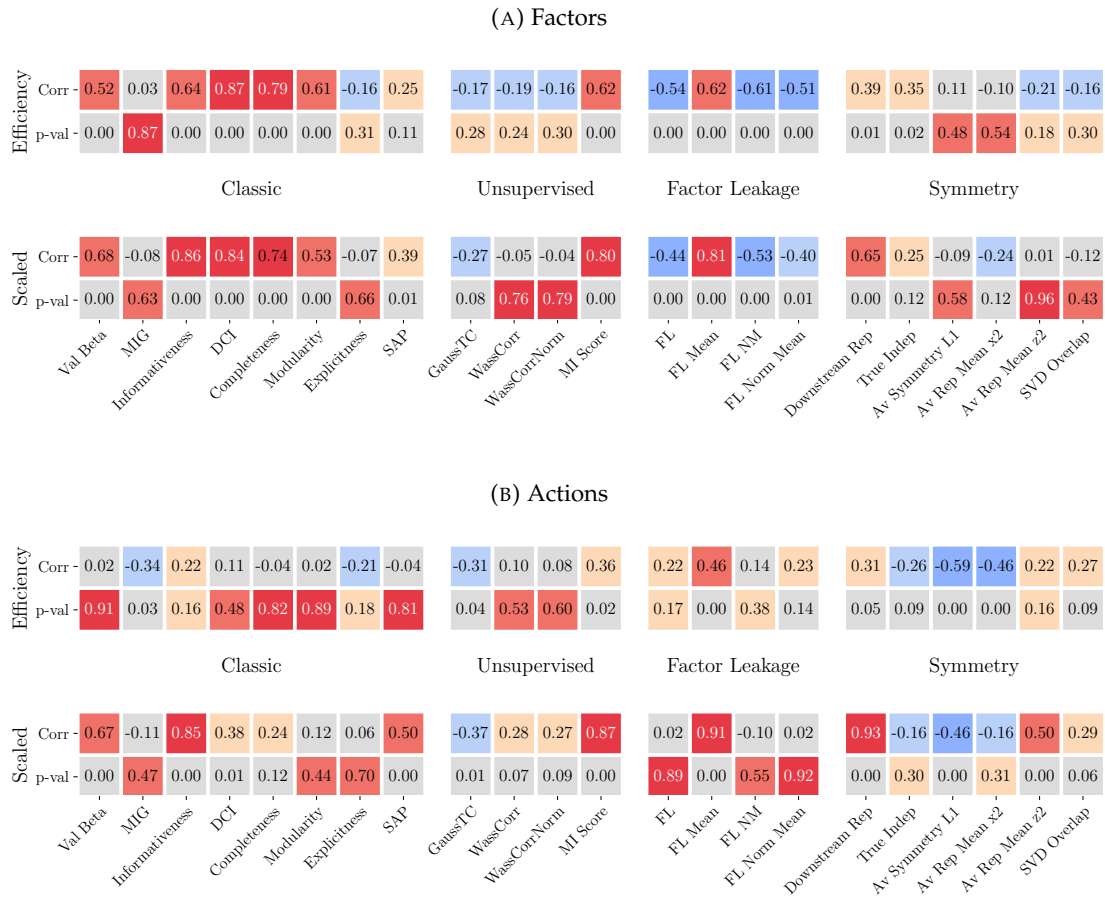


FIGURE 5.20: Correlation of data efficiency with metrics for combined FlatLand and dSprites.

In Fig. 5.21 we report non-zero metric weights as we vary the L1 penalty. In both figures, the low penalty weights are not particularly interesting, with Fig. 5.21a showing multiple equivalently weighted metrics and Fig. 5.21b massively prioritising a training metric - which likely doesn't capture much true information. With a weight of 1 we start to see trends occur. In both figures the FL Mean metric is strongly weighted throughout, which correlates with its strong individual correlation scores as seen in Fig. 5.20. Alongside this metric, for the factor prediction task we see the DCI disentanglement being prominent whilst for the action prediction task, FL Mean is basically the only important metric. Note that the correlation of the linear combination is larger than the best individual correlation score (0.87) for the factor prediction task. For the action task however, the correlation is stronger for lower weights (< 20) but returns to the individual score for the highest weighting.

The Pearson correlation coefficient can be slightly misleading, especially for small amounts of data. In Fig. 5.22 we scatter the metric combinations against (scaled) data efficiency for both tasks and for two different L1 penalty weights. For the factor prediction task we can see that the correlation looks strong, and matches with the high Pearson correlation score. Furthermore, both datasets seem to show the same



FIGURE 5.21: Optimal linear combination of metrics to correlate with data efficiency as the weight penalty is increased.

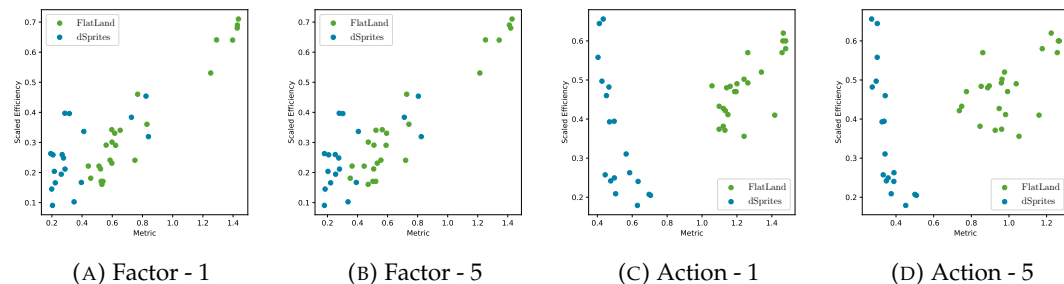


FIGURE 5.22: Scatter metric combinations against efficiency for L1 penalty weights 1 and 5.

relationship. For the action prediction task however, we see two distinct populations, corresponding to the two datasets. We also see that the relationship for dSprites is very different to that of FlatLand. dSprites seems to show some slight negative correlation, whereas FlatLand shows very little correlation at all. As we have seen in previous experiments, the action prediction task is extremely messy and doesn't seem consistent across datasets.

5.5 Conclusions

This chapter has aimed to address the following questions:

- What is the posterior structure of FlatLand linear disentangled representations?
- Can we find better priors for FlatLand?
- Can we find evidence of LDRs in disentanglement metrics?
- How do LDRs compare to classical representations under data efficiency and downstream tasks?
- Can we find any correlations with data efficiency?

What is the posterior structure for ForwardVAE on FlatLand? Linear disentangled representations offer more structured distributions than standard disentanglement, and through SBDRL, their structure can be defined very precisely with respect to the symmetry groups acting on the data. By comparing different variants of FlatLand, we saw that the posterior structure changed correspondingly. For instance, by increasing the order of the cyclic symmetries, we saw a posterior with more density peaks. When considering FlatLandN, we found that cyclic structures were present in the posterior despite not being observed in the data. When ablating the KL divergence, we saw that this no longer occurred, suggesting that cycles helped minimise this divergence by increasing the overlap between density peaks and increasing the density at the origin.

It is interesting to consider what structure is desired for FlatLandN. For FlatLand/G it is obvious that cyclic structures are both aesthetic and a good representation of the data. For FlatLandN cycle-less posteriors are the best representation of the data as observed. Furthermore for a downstream task such as planning, this would be the best representation assuming boundary warping is not a valid action in this task. As such, the standard ForwardVAE model would not perform well, since it continued to learn cyclic structures. Viewing the KL as a capacity constraint, the cyclic posterior learnt by ForwardVAE is the simplest (lowest capacity) representation available under the standard prior. This may not always match the actual observed structure of the data.

Since ablating the KL biased the model away from cycles on FlatLandN, this allowed us to observe the bias introduced by cyclic internal representations. Interestingly, we saw that this bias was not strong for either the standard model, or when ablating the KL. In the latter case we did observe cyclic behaviour more frequently, but it was far from guaranteed, as might be expected for purely cyclic internal representations.

Can we find better priors? We observed that ForwardVAE learns cyclic structure for FlatLandN, a dataset without observed cycles. We noted that this was the lowest capacity representation under the standard prior. This suggests that the standard prior is too simple to capture the true observed structure of this dataset.

To explore this we decided to emulate a change in prior through normalising flows. By mapping the posterior to a parametrised posterior, we can calculate its KL against the standard prior. This allows more complex posteriors whilst retaining a simple (to compute) KL divergence. When comparing the divergence under the normal prior to that when using the flow layers, we saw that the latter was indeed more efficient, by a factor of 5.

As mentioned, a complex posterior should be able to represent the observed data more faithfully than a simple one, since it has more capacity. Visualising the posteriors under the normalising flows, this is exactly what we saw. The flow based models did not learn cyclic structures on FlatLandN whilst continuing to learn them on FlatLand/G. This observation highlights a choice we have when defining the model. Do we prefer to model the true observed structure accurately or do we prefer to have a simple representation of the data. As mentioned, for tasks such as planning, an accurate representation is vital, however this may not be true of other downstream tasks. Furthermore, the observed data may not actually fully represent the distribution from which it is drawn - i.e. we may have incomplete data. In this case, the simplest representation may actually be more accurate and more useful than representing the observed data well.

Can we find evidence of LDRs in disentanglement metrics? Given the highly structured nature of the LDR representations, we wanted to determine whether this is reflected in disentanglement metrics. Under the classification accuracy, ROC-AUC, PR and F1 scores, we saw good discriminability from a number of classical metrics, the Factor Leakage metrics and the independence based metrics. Whilst other metrics were discriminable on one of the two datasets tested, these were the only ones consistent across datasets and under different scores.

These results indicate that even if we do not know the symmetry structure, we may be able to determine if a representation is linearly disentangled or not based solely on metric scores. This would be a very useful result. Since we have only explored this for FlatLand and dSprites, we cannot conclusively say whether the metrics would continue to be discriminative across more datasets. It would be interesting to see a larger scale study of this, which forms potential future work.

How do LDRs compare to classical representations under data efficiency and downstream tasks? Performance on downstream tasks is perhaps the most important judge of a representation. To that point, we find that symmetry based models score consistently higher downstream scores on both the task of generative factor estimation and observed action estimation. Whilst the latter is unsurprising since they are trained using action transitions, the former is much more interesting. It

is likely that the constrained structure of linear disentangled representations is particularly suited to this task. This was seen in Figures 5.13 and 5.14 where we saw that you could achieve stronger performance with simpler classifiers using LDRs as opposed to classical (disentangled) representations. This is obviously a beneficial property for downstream tasks, since it allows downstream models to be less computationally intensive.

Sample efficiency is another method to demonstrate the utility of a representation. If we can learn a downstream task from a small dataset drawn from the representation, then it is obviously a good representation. Admittedly this may be susceptible to task complexity, since a small dataset may not have enough samples to represent the entirety of a complex task, no matter how good the representation is. Nevertheless, we saw that for the factor prediction task, symmetry based models consistently offered better data efficiency than classical models. The action prediction task was less clear, partially due to extremely poor performance by classical models regardless of dataset size. When efficiency is scaled by the end performance, we see that the symmetry based models have efficiency equivalent to the best classical model, or better. It is important to note, however that efficiency is more difficult to estimate accurately than simple downstream performance. Whilst we see that symmetry based models generally have better efficiency, this may not hold over more datasets or with many more models trained.

Can we find any correlations with data efficiency? [Locatello et al. \[2018\]](#) found through large scale study that there was no consistent evidence of strong correlations between disentanglement metric scores and data efficiency scores. Whilst individual datasets and metrics were seen to have correlation in isolation, this was not the case when considered across all datasets. In Figures 5.18 and 5.19 we look for consistent correlations in classical and symmetry based metrics with and without symmetry based models. We found no consistent correlations between any metrics and the efficiency, similar to [Locatello et al. \[2018\]](#), despite the addition of symmetry based metrics. In general, the inclusion of symmetry based models increased the correlation scores across all metrics, however this is due more to efficiency correlating with the type of disentanglement - linear vs otherwise. Since symmetry based metrics do not consistently correlate with efficiency, this does suggest they do not (individually) capture the extent to which a model is linearly disentangled - even if they may be able to discriminate linear vs not. These results were also consistent when combining the data points of FlatLand and dSprites - although it is not obvious if this is a useful test since scores across datasets can vary dramatically.

Action Prediction vs Factor Prediction In general, we found that the factor prediction task was a more suitable evaluation problem, with the action prediction

task being simpler yet far more inconsistent in the results. For example, all models performed worse on the action prediction task than the factor task, despite there being far more possible factor values. The exception to this is the RGrVAE policy distribution, which was (unsurprisingly) extremely effective on the action prediction task. We did not evaluate it on the factor prediction task since we knew there would be no signal. It is interesting to note however, that this could be a useful (complementary) representation for tasks in which knowledge of observed actions is important.

Chapter 6

Conclusions

This work was comprised of two main parts, Chapter 4 concerned with learning linear disentangled representations and Chapter 5 concerned with the structure of such representations.

Chapter 4 addressed the following questions:

1. Can we learn linear disentangled representations?
2. Are they learnt by standard VAE models?
3. How do we measure them?
4. Can we learn them without knowledge of action labels?

Answering these questions (Section 4.7) lead to a number of contributions. The first of which, GroupVAE was introduced to extended the domain in which we can learn linear disentangled representations to problems with actions which were completely unlabelled. In order to measure the extent to which a representation was linear disentangled, we had to introduce the independence, SVD and factor leakage metrics discussed in Section 4.4. We could then introduce a method to optimise for the best representation of symmetries on top of pre-existing representations, and determine through the metrics if these representations were linearly disentangled or not.

Disentanglement research has generally be limited to synthetic datasets where the generative factors are available, e.g. dSprites. Recently there has been work towards creating real world datasets in a constrained environment, where we can control the generative factors ([Gondal et al., 2019]). As such, our work to extend the domain in which we can learn linear disentangled representations is a particular strength. At the same time, by still requiring knowledge of transitions, it can also be considered a weakness, which leads to future work in completely unsupervised learning of linear

disentangled representations. Similarly, our methods to determine if a pre-existing representation is linearly disentangled or not are important tools in this area, but by requiring knowledge of the symmetry structure, cannot be applied to all problems.

Chapter 5 addressed the following questions:

- What is the posterior structure of FlatLand linear disentangled representations?
- Can we find better priors for FlatLand?
- Can we find evidence of LDRs in disentanglement metrics?
- How do LDRs compare to classical representations under data efficiency and downstream tasks?
- Can we find any correlations with data efficiency?

Through these questions we explored the posterior structure of linearly disentangled models on data with different symmetry structures. We found that by allowing for more complex posteriors, we learn structures which were closer to those which generated the data (i.e. without cycles). Through evaluating a variety of model instances, we then found evidence that linear disentangled representations result in better downstream performance and high data efficiency.

Consistently stronger performance on downstream tasks would be a significant advantage of linear disentangled representations over classically disentangled versions. It is interesting to find evidence of this in this work, however a weakness which can form the basis of potential future work is to explore this on a much large scale in order to determine strong correlations.

6.1 Future Directions

Throughout this work there have been directions and questions which we would like to have explored but could not due to time limitations. We will now briefly discuss some of these directions.

How can we determine the symmetry structure of a linearly disentangled representation without prior knowledge? Our exploration of structure is limited to cyclic symmetries, since it is hard to determine and model other symmetries which might be relevant in physical problems. Consequently, we knew which structures to search for when trying to determine if a given representation is linearly disentangled. Furthermore we could easily sample transitions based on the actions by these

symmetries. For general data, it is not obvious what the true symmetry structure is, or even if there is a notion of truth with regards to symmetries acting on the data.

Models which do not observe transitions are free to learn representations with respect to any symmetries (or none), and we currently have no way to determine if this has happened unless by chance they coincide with those we choose.

Assuming we search for the correct type of symmetry (i.e. cyclic) but not the specifics (i.e. order), then directly learning an approximation of the representation, or an intermediate space, would likely suffice to determine if a representation is linearly disentangled. When we search for the wrong type of symmetry, then none of the methods we have explored would suffice. In general, cyclic symmetries may be the best default choice. [Quessard et al. \[2020\]](#) believe that cyclic symmetries are dominant in nature, and the fundamental theorem of finite abelian groups shows that cyclic groups are expressive, however there are no guarantees that baseline models will learn them.

Can we encourage LDRs in standard VAEs? We have seen in earlier chapters that LDRs are learnt in standard VAE models, however only rarely. This suggests that whilst LDRs are not discouraged by the objectives of these models, they are not a global optima - perhaps not even a good optima. There have been many objective functions tailored towards classical disentanglement, indeed all the baseline VAEs we test do this to some extent. In most cases, they do not change the model architecture significantly, and simply modifying the objective often results in improved results. Since LDRs are not discouraged, it would be ideal if they could be encouraged through a new objective term, rather than having to directly observe actions of each symmetry. Whilst [Caselles-Dupré et al. \[2019\]](#) show that we cannot choose the symmetry structure without observing their actions, this does not imply that linear disentangled representations cannot be learnt without such observations.

Large Scale Study of LDR Efficiency In this work, we have presented preliminary results on the data efficiency of LDRs compared to classical models, and preliminary correlations between efficiency and metrics. However accurate estimates of these requires consideration of more datasets and larger samples sizes than was feasible for this work. [Locatello et al. \[2018\]](#) found that performance varied quite significantly across datasets, and whilst we considered two different datasets, this possibly does not provide the full picture. A larger scale study would allow us to better determine if these results are representative, and could also

Planning Tasks We have seen that ForwardVAE and RGrVAE learn actions which are accurate and consistent over composition. The ability to accurately model states

over time is a vital component of any planning model. Whilst there have been reinforcement learning models which learn to approximate actions, to our knowledge, none have done so in a linear way. It seems that linear disentangled representations should be a good fit for planning tasks. This could provide an additional downstream task with which to compare linear disentanglement to standard disentanglement. Planning is much harder and more useful task than action/factor prediction, so this would better determine which might be better for real world scenarios.

Representations of Continuous Groups We have limited our study to finite groups so that we can explicitly learn a representation for each group element or group generators. Many important symmetries such as rotations are fundamentally continuous in nature, and whilst we can learn finite approximations (i.e. approximately generate $SO(2)$ by a small rotation angle), we could not capture the full symmetry. This was studied by [Quessard et al. \[2020\]](#) outside of the VAE framework, and could be applied to ForwardVAE or GroupVAE models.

On a related note, continuous representations are already achievable by the attentional variant of GroupVAE, however the variant does not learn in practice. If it were to work as intended, AGrVAE would learn a basis for the space of actions that it observes. This would allow for learning of continuous symmetries at the expense of strongly associating each internal representation with a known action.

Can we learn intermediate spaces alongside training a VAE In Chapter 4, we learnt a linearly disentangled intermediate space from a standard VAE representation. We found that this could not be done with linear models (the space isn't already linear) but non-linear models could find a good representation given time. Theoretically, there is no reason that we cannot learn the intermediate space alongside learning the base VAE representation. An important consideration is that gradients from the intermediate network would need to be detached from the VAE backbone, since otherwise the model becomes equivalent to ForwardVAE. This will likely slow down training, so this form of model would probably be slower to train than ForwardVAE, and potentially have lower quality (linear) representations, since the VAE is not being optimised explicitly for that. The ideal result would be learning a linear disentangled intermediate space on top of a VAE with high fidelity samples - such as a PixelVAE ([\[Gulrajani et al., 2016\]](#)) - so that we benefit from quality samples and an interpretable latent space.

Another minor direction to explore with this model would be the use of invertible network layers. My current implementation learns two separate networks, which are optimised to be forward and inverse functions for the intermediate space. Invertible network layers such as the invertible 1x1 convolutions by [Kingma and Dhariwal](#)

[2018] would enable us to learn a single layer with the inverse being exact, rather than an optimised approximation. This would also allow us to be more confident that the inverse network is not introducing information, which is not guaranteed in our current implementation.

Learning LDRs by Reinforcement Tasks Caselles-Dupré et al. [2019] showed that interaction with the environment was required to learn linear disentangled representations. In our work, we used REINFORCE to learn an action estimation policy. Despite this, neither of these works used particularly difficult reinforcement learning tasks, or any which are more suited to the observation-action framework (such as navigating a maze). It would be interesting to see if LDRs are beneficial to solving these tasks, or indeed if representations of these tasks tend to be more linear in nature.

Appendix A

Correlation Scatters

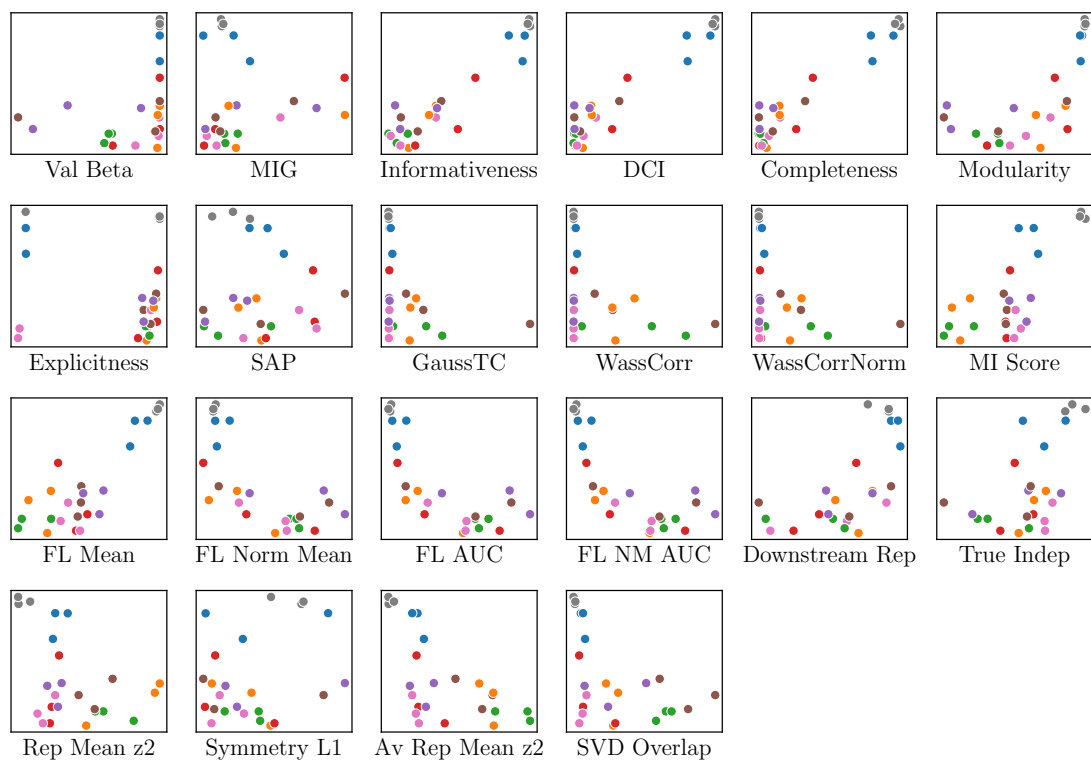


FIGURE A.1: Scatter of data efficiency (y) and disentanglement metrics (x) for FlatLand factor prediction task.

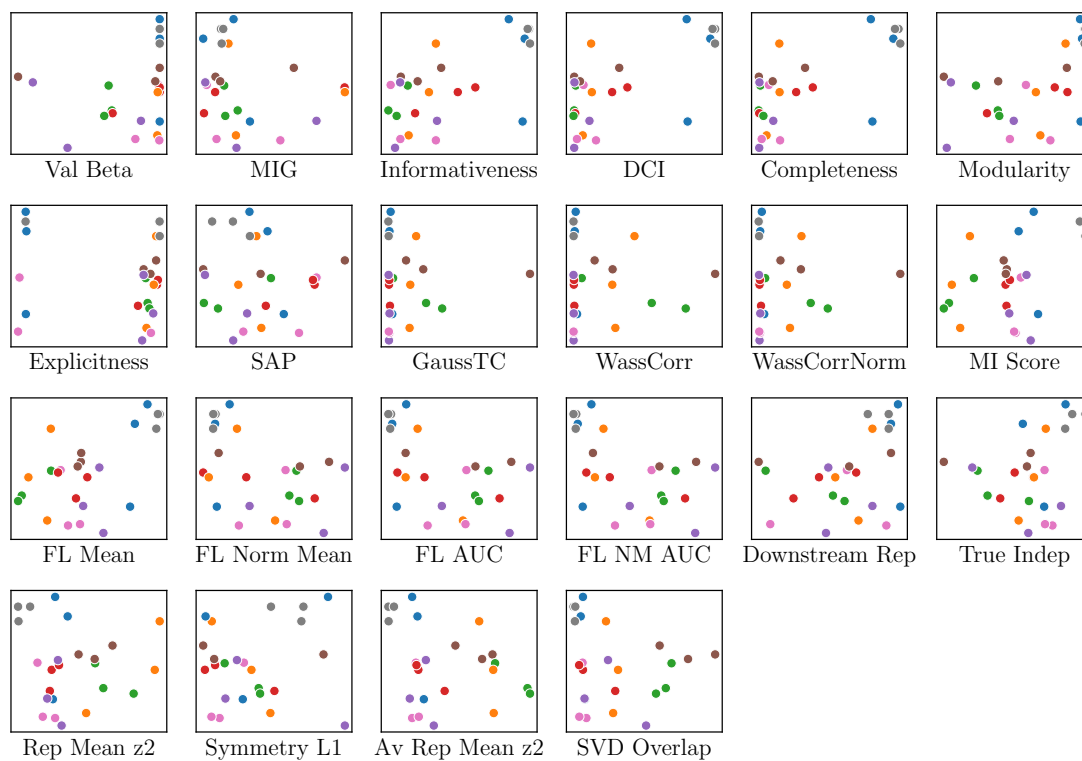


FIGURE A.2: Scatter of data efficiency (y) and disentanglement metrics (x) for FlatLand action prediction task.

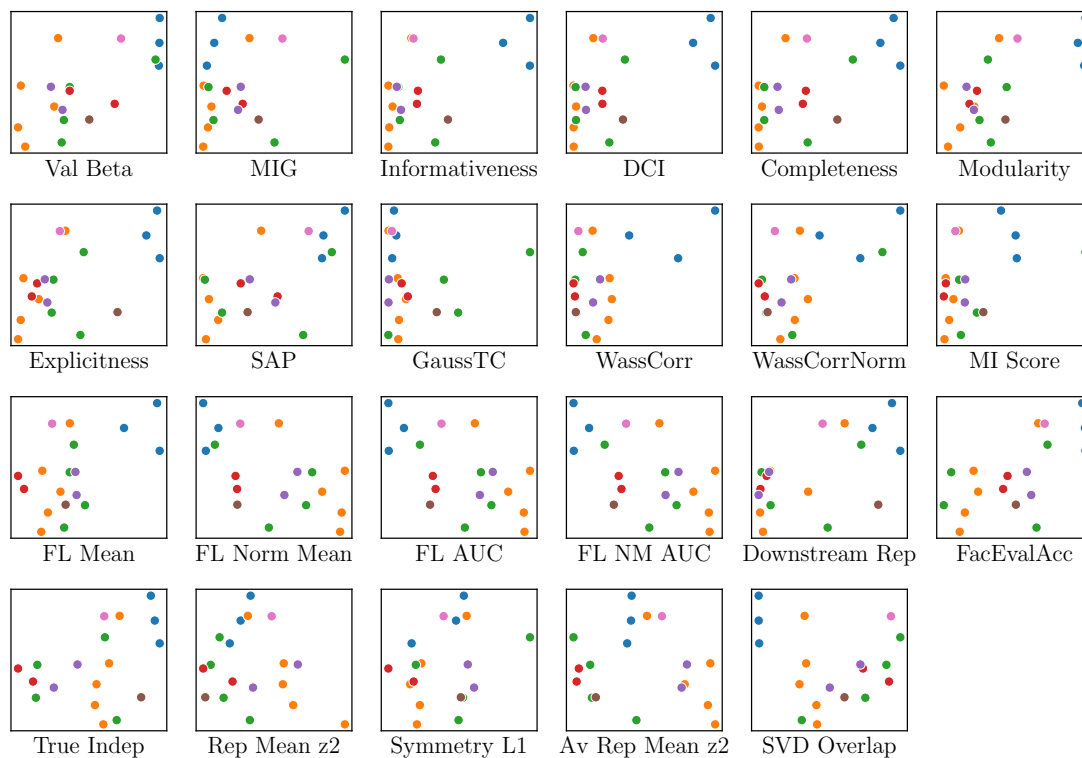


FIGURE A.3: Scatter of data efficiency (y) and disentanglement metrics (x) for dSprites factor prediction task.

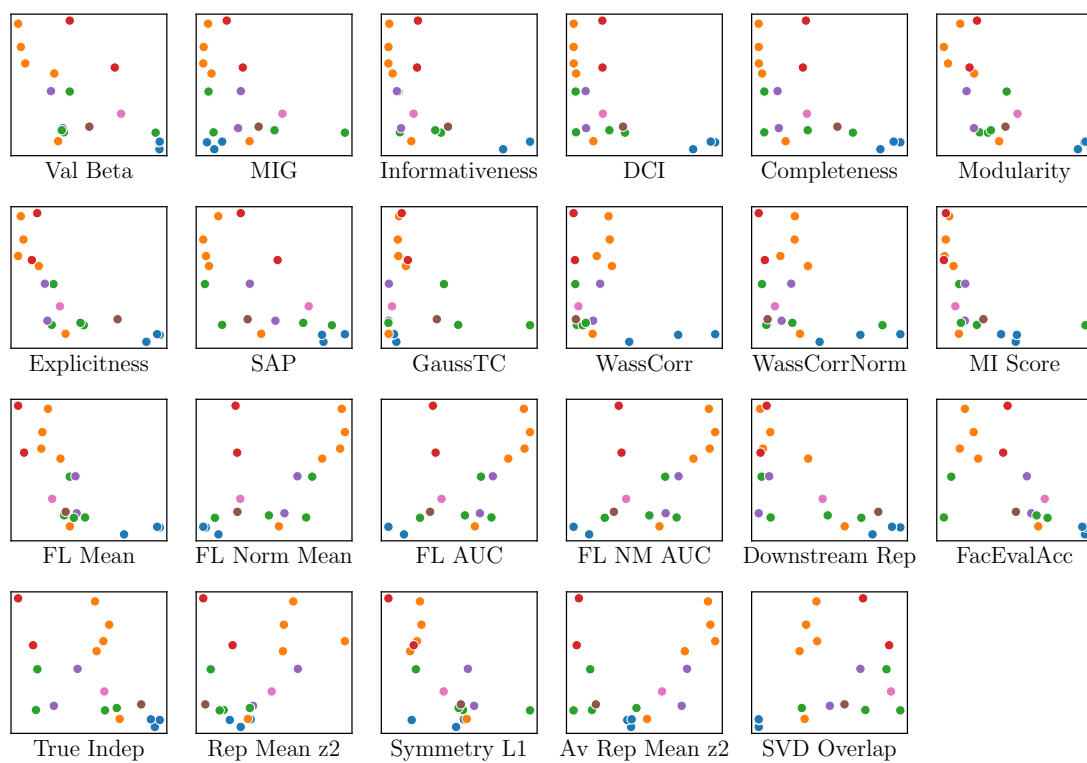


FIGURE A.4: Scatter of data efficiency (y) and disentanglement metrics (x) for dSprites action prediction task.

Appendix B

Standard Model Architectures

Throughout this work we have used a number of models which work on top of a VAE backbone. In general we use the same VAE backbones for different types of VAE. We now define some of the common backbone architectures which we have used.

B.1 Shapes

The Shapes backbone encoder is a 3 layer MLP with $(1200, 1200, N_{\text{latent}})$ units and ReLU activations. The corresponding decoder is a 4 layer MLP with $(1200, 1200, 1200, N_{\text{out}})$ units, and tanh activations.

B.2 Celeb

The Celeb backbone encoder consists of 2 convolution layers with 32 filters, kernel size of 4, stride of 2 and padding of 1, followed by another two convolution layers with 64 filters and the same kernel size, padding and stride. The convolution layers are followed by two linear layers with $(256, N_{\text{latent}})$ units. All layers use the ReLU activation.

The Celeb decoder then consists of two linear layers with $(256, 1024)$ units followed by 4 transpose convolution layers with $(64, 32, 32, N_{\text{out}})$ filters, kernel size of 4, stride of 2 and padding of 1. All layers again use the ReLU activation.

B.3 Forward

The Forward backbone encoder consists of 4 convolutional layers with 32 filters, kernel size of 4, stride of 2 and padding of 1 for each layer. The convolution layers are followed by 3 linear layers with (256, 256, N_{latent}) units. All layers use the SeLU activation [Klambauer et al., 2017].

The forward decoder consists of 2 linear layers with (256, 512) units. This is followed by 4 transpose convolution layers which have (32, 32, 32, N_{out}) filters, kernel size of 4, stride of 2 and padding of 1. All layers use the SeLU activation.

B.4 TC Model

The TC model encoder consists of 4 convolutional layers with 32 filters, kernel size of 4, stride of 2 and padding of 1 for each layer. The convolution layers are followed by 2 linear layers with (256, N_{latent}) units. All layers use the Leaky ReLU activation.

The decoder consists of 1 linear layer with 512 units. This is followed by 4 transpose convolution layers which have (32, 32, 32, 32) filters, kernel size of 4, stride of 2 and padding of 1. There is a final convolutional layer (no activation) at the output which has N_{out} filters, a kernel size of 3 and padding of 1. All layers use the Leaky ReLU activation.

B.5 MMD-VAE

The MMD-VAE model used 5 convolution layers in the encoder (32, 64, 128, 256, 512) filters, with kernel size of 3, stride of 2 and padding of 1. This is followed by 1 linear layer with N_{latent} units. The decoder has a single linear layer with 2048 units followed by 5 transpose convolutions (512, 256, 128, 64, 32) filters, kernel size of 3, stride of 2 and padding of 1. There is a final convolution layer at the output (no activation or batch norm) which has N_{out} filters, kernel size 3, stride 2 and padding 1.

All layers in both the encoder and decoder used leaky ReLU activations.

Convolutional (normal and transpose) layers have a batch norm layer before the activation is applied.

B.6 RGrVAE Action Estimator

The CNN which learns to estimate actions in RGrVAE consists of 3 convolution layers with (32, 16, 16) filters followed by a linear layer with N_{actions} units. The convolutional layers have kernel size 3, stride 2 and use ReLU activations. The output is then also softmaxed.

B.7 RGrVAE

All RGrVAE instances used the action Estimator network defined in B.6. The backbone VAE used for both FlatLand and dSprites was the backbone defined in B.3.

Appendix C

Training Schemes

In all experiments, results (metrics or figures) are reported using a 10% validation set which was held-out during training. Metrics are reported as averages over this validation set.

C.1 FlatLand

All models trained on FlatLand share a number of parameters. All models use the same networks for encoder and decoder (See Forward B.3). All models have 4 latent dimensions, batch size of 128, base learning rate of 0.0001, Adam optimiser and mean square error loss function. Each model type has some additional parameters which we now detail. All models not listed were trained for 100 epochs. Error bars are always provided over 5 repeats for each model.

ForwardVAE The ForwardVAE instances were trained for 300 epochs.

RGrVAE The RGrVAE instances were trained for 300 epochs, used $\gamma = 10$, exploration epsilon of 0.95, and entropy weight of 0.01. The internal group structure was C_N , and used a learning rate of 0.1.

BetaVAE The BetaVAE instances were trained with $\beta = 10$.

DIP-VAE The DIP-VAE variants used parameters $\lambda_{od} = 10$ and $\lambda_d = 5$.

FactorVAE The FactorVAE used the parameter $\gamma_{\text{factor}} = 6.4$.

BetaTCVAE The BetaTCVAE model used parameter $\beta_{TC} = 6$.

MMD-VAE The MMD-VAE used parameters $\alpha_{MMD} = -0.1$ and $\lambda_{MMD} = 20$.

C.2 dSprites

All models trained on dSprites share a number of parameters. All models use the same networks for encoder and decoder (See Forward B.3). All models also have 16 latent dimensions, base learning rate of 0.0001, Adam optimiser and binary cross entropy loss function. Each model type has some additional parameters which we now detail.

RGrVAE The RGrVAE instance on dSprites use an internal representation learning rate of 0.1, internal group structure of $GL(2)$ matrices, $\gamma = 10$, exploration epsilon of 0.95, batch size of 1024, and trained for 5 million iterations.

VAE The VAE models use a batch size of 128, and were trained for 2.5 million iterations.

BetaVAE The BetaVAE models were trained using a $\beta = 5$ for 2.5 million iterations and a batch size of 1024.

Dip VAE i The DIP-VAE-I model was trained for 1 million iterations using a batch size of 1024.

Dip VAE ii The DIP-VAE-II model was trained for 2.5 million iterations using a batch size of 1024.

Extental models Two external models were sourced from the PyTorch-VAE github project [Subramanian, 2020], and their pretrained weights were used on dSprites experiments. These models were the BetaTcVAE and FactorVAE.

Bibliography

- Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.
- Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.
- William Burnside. On groups of order $p\alpha q\beta$. *Proceedings of the London Mathematical Society*, 2(1):388–392, 1904.
- Hugo Caselles-Dupré, Louis Annabi, Oksana Hagen, Michael Garcia-Ortiz, and David Filliat. Flatland: a lightweight first-person 2-d environment for reinforcement learning. *arXiv preprint arXiv:1809.00510*, 2018.
- Hugo Caselles-Dupré, Michael Garcia Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. In *Advances in Neural Information Processing Systems*, pages 4608–4617, 2019.
- Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.
- Xi Chen, Yan Duan, Rein Houthoofd, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.
- Brian Cheung, Jesse A Livezey, Arjun K Bansal, and Bruno A Olshausen. Discovering hidden factors of variation in deep networks. *arXiv preprint arXiv:1412.6583*, 2014.

- Jongwook Choi, Yijie Guo, Marcin Moczulski, Junhyuk Oh, Neal Wu, Mohammad Norouzi, and Honglak Lee. Contingency-aware exploration in reinforcement learning. *arXiv preprint arXiv:1811.01483*, 2018.
- Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.
- Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.
- Emilien Dupont. Learning disentangled joint continuous and discrete representations. *Advances in Neural Information Processing Systems*, 31, 2018.
- Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.
- Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *International Conference on Machine Learning*, pages 1755–1763, 2019.
- Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/d97d404b6119214e4a7018391195240a-Paper.pdf>.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.
- Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017.

-
- Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.
- Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2, 2016.
- J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. .
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations, ICLR*, 2014.
- Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27:3581–3589, 2014.
- Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proceedings of the 31st international conference on neural information processing systems*, pages 972–981, 2017.
- Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015.
- S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951. . URL <https://doi.org/10.1214/aoms/1177729694>.
- Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104. IEEE, 2004.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv preprint arXiv:1811.12359*, 2018.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. <https://github.com/deepmind/dsprites-dataset/>, 2017.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

Matthew Painter, Jonathon Hare, and Adam Prugel-Bennett. Linear disentangled representations and unsupervised action estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 13297–13307. Curran Associates, Inc., 2020a. URL <https://proceedings.neurips.cc/paper/2020/file/9a02387b02ce7de2dac4b925892f68fb-Paper.pdf>.

Matthew Painter, Jonathon Hare, and Adam Prugel-Bennett. On the structure of cyclic linear disentangled representations. In *1st NeurIPS workshop on Interpretable Inductive Biases and Physically Structured Learning*, 2020b.

Robin Quessard, Thomas D Barrett, and William R Clements. Learning group structure and disentangled representations of dynamical environments. *arXiv preprint arXiv:2002.06991*, 2020.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

Karl Ridgeway and Michael C Mozer. Learning deep disentangled embeddings with the f-statistic loss. In *Advances in Neural Information Processing Systems*, pages 185–194, 2018.

Oleh Rybkin, Karl Pertsch, Konstantinos G Derpanis, Kostas Daniilidis, and Andrew Jaegle. Learning what you can do before doing anything. *arXiv preprint arXiv:1806.09655*, 2018.

A.K Subramanian. Pytorch-vae. <https://github.com/AntixK/PyTorch-VAE>, 2020.

Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable factors. *arXiv preprint arXiv:1708.01289*, 2017.

William F Whitney, Michael Chang, Tejas Kulkarni, and Joshua B Tenenbaum. Understanding visual concepts with continuation learning. *arXiv preprint arXiv:1602.06822*, 2016.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

H. Wussing. *The Genesis of the Abstract Group Concept: A Contribution to the History of the Origin of Abstract Group Theory*. Dover Books on Mathematics Series. Dover Publications, 2007. ISBN 9780486458687. URL <https://books.google.co.uk/books?id=Xp3JymnfAq4C>.

Tao Yang, Xuanchi Ren, Yuwang Wang, Wenjun Zeng, and Nanning Zheng. Towards building a group-based unsupervised representation disentanglement framework. In *International Conference on Learning Representations*, 2021.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.