Imperial College of Science, Technology and Medicine

Department of Computing

MEng Mathematics and Computer Science

Project Interim Report

Author:
Dane Sherburn

CID:
00820119

Supervisors:
Prof. Murray Shanahan
Pedro Mediano

January 27, 2017

Contents

1 Introduction

A long term goal of artificial intelligence (AI) is the development of artificial general intelligence (AGI). A number of theoretical frameworks have been presented to formalize what it means to achieve AGI, notably by Hutter **?**.

Reinforcement learning is fundamental in this framework. It's therefore quite exciting that deep reinforcement learning (DRL) systems have recently been able to master a wide range of tasks, including Atari games and Go. Though DRL systems have been remarkably successful, they still have a number of drawbacks **?**:

1. **Slow to learn**. Deep neural networks require large data sets and are therefore slow to learn.
2. **Fail to transfer past experience**. They often fail to perform well on tasks very similar to those they have mastered.
3. **Inability to reason abstractly.** They fail to exploit statistical regularities in the training data by using high-level processes like planning or causal reasoning.
4. **Hard to reason about.** It's often troublesome to understand why the DRL system chose the action it did.

Deep symbolic reinforcement learning (DSRL) is a recent advance which overcomes all of these drawbacks at once without the drawbacks from classical symbolic AI, namely the symbol grounding problem **?**.

This novel architecture relies on the unsupervised extraction of compositional features, allowing for transfer learning and high-level cognitive processes. However, the unsupervised extraction of features from a wide range of scenes is still a challenge in AI research **?**. Fortunately methods are getting better, and the first unsupervised scalable model β-VAE was developed recently.

The aim of this project is to investigate compositional representations using β-VAE and evaluate their effectiveness in deep symbolic reinforcement systems. We plan to achieve this as follows:

1. Implement many flavours of autoencoders, including mixing and matchings of standard, convolutional and variational autoencoders, and investigate the relationship between the latent space and their final reconstructions. This will hopefully convey some intuition about the way scenes are typically represented in the latent space, their relevance in transfer learning and the significance of the problem we're trying to solve.
2. Next we focus our attention on convolutional variational autoencoders. We'll experiment with different values of β two ways: through heuristic visual inspection, and by comparing the disentanglement metric and reconstruction error. All experiments will be on a range of Atari 2600 games using the Atari Learning Environment.

3. Finally we can evaluate the value of a compositional structured latent space in a deep symbolic reinforcement system. We'll do this by training a number of reinforcement agents on the latent space and compare their performance to training on the on the original space. We can also train the one agent on multiple games to assess how well this compositional structure facilitates transfer learning.

2 Background

This chapter will contain the material necessary to understand later chapters and provide the context for our contributions.

We will introduce deep symbolic reinforcement learning which will motivate our interest in unsupervised disentangled representations. Here we'll explore state-of-the-art approaches to finding such representations, which will also require a brief introduction to autoencoders. Finally we'll mention less theoretical matters, including libraries used and the Arcade Learning Environment.

2.1 Autoencoders

2.1.1 Architecture

Autoencoders are standard feed-forward networks with a "bottleneck" in a hidden layer, shown in Figure (**??**). Autoencoders learn the identity function $f(\boldsymbol{x}) = \boldsymbol{x}$, which may seem pointless at first. However, if the autoencoder closely approximates the identity function, the decoder is actually computing a reconstruction of the original image $\widetilde{\boldsymbol{x}}$ from a lower-dimensional representation $\boldsymbol{z}$, called the *latent space*. So for any given $\boldsymbol{x}$, we can compute a compressed version, $\boldsymbol{z}$, by submitting $\boldsymbol{x}$ to the encoder. This compression technique is data-specific and, at present, isn't scalable **?**. However, they are very effective at denoising images and likely to play an important role in reverse Google image searches **?**.

The type of feed-forward network chosen depends on the nature of the data. We'll be using images as our input data, so the natural choice is to use convolutional neural networks. Our encoder will include convolutional and pooling layers to learn relevant features and downsample the data respectively, and our decoder will include deconvolutional and upsampling layers (as they're natural inverses).

2.1.2 Convolutional autoencoders

2.1.3 Variational autoencoders

The autoencoders we've seen so far do the following: given input $\boldsymbol{x}$, compute its reconstruction $\widetilde{\boldsymbol{x}}$, after mapping it to a lower dimensional vector $\boldsymbol{z}$, the *latent space*. The aim of the variational autoencoder (VAE) is quite similar: given input samples from an unknown distribution $p(\boldsymbol{x})$, generate $\widetilde{\boldsymbol{x}}$ very similar to those in $p(\boldsymbol{x})$, after

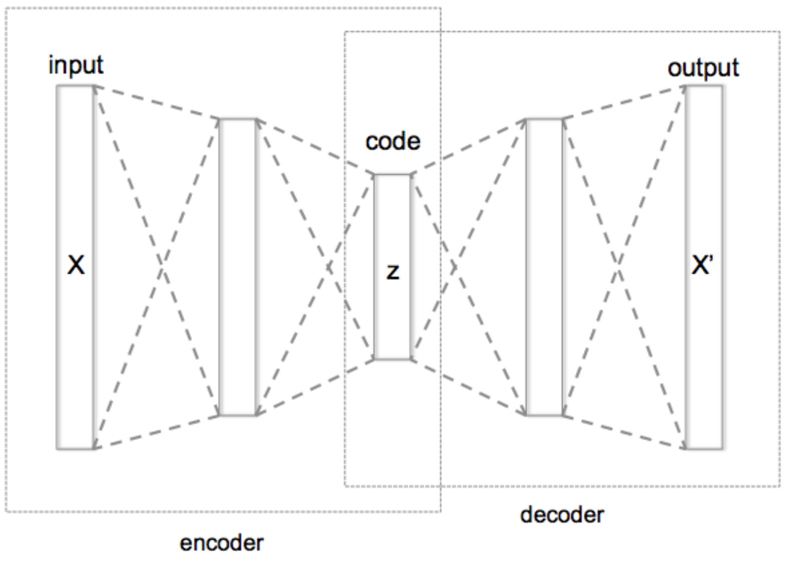


Figure 1: Autoencoders can be split into the encoder, which takes input $\boldsymbol{x}$ and outputs the low-dimensional $\boldsymbol{z}$, and the decoder, which takes input $\boldsymbol{z}$ and tries to reconstruct $\boldsymbol{x}$.

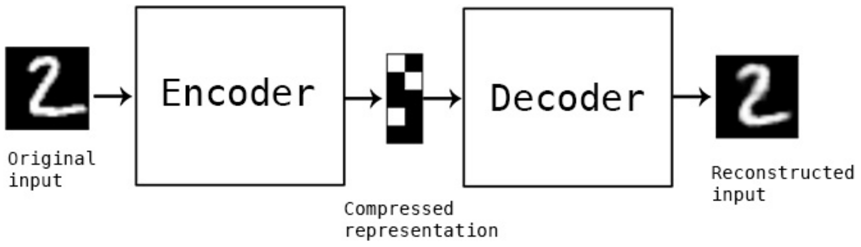


Figure 2: A black-box view of the autoencoder showing the relationship between the encoder and decoder. The latent space shown in the center is clearly of a lower dimension than the input, acting as a code. **?**

sampling a lower-dimensional vector $\boldsymbol{z}$. Since VAEs generate unseen, similar samples, they are *generative models* **?**.

To justify the statements in the next paragraph, we have to mention that $p_\theta(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})}$ is intractable due to the marginal likelihood $p(\boldsymbol{x})$ being exponentially computationally expensive, motivating the introduction of the probabilistic encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to approximate the posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. Since $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ appears in the loss function but is not differentiable (we can't differentiate random variables), we need to use the *reparameterisation trick*. The reparameterisation trick expresses $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$ as a deterministic variable $\boldsymbol{z} = g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x})$ where $\boldsymbol{\epsilon}$ is an auxilary variable with independent maginal and g_ϕ is a deterministic vector function parameterized by ϕ **?**.

Learning the optimal $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ is formally expressed as the following objective:

$$\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathbf{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathcal{D}_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})] \tag{1}$$

2.2 Deep symbolic reinforcement learning

2.2.1 Reinforcement learning

2.2.2 Symbolic artificial intelligence

2.2.3 Deep symbolic reinforcement learning

2.3 Representation learning

Suppose we were ordered to carry out the multiplication of two binary numbers, 1101_2 and 101101_2. Most of us would first convert these numbers to base-10 before performing the multiplication. We'd do this because the problem is significantly harder in base-2 than it is in base-10. It's therefore intuitive that a good representation is vital to the agent required to solve it **?**.

The subject of what makes a good representation has generated much interest in recent times. According to **?**, a good representation is simply one that makes subsequent tasks easier.

2.3.1 Disentangled representations

It's been suggested by **?** that a crucial factor in what makes a good representation is how *disentangled* it is. A disentangled representation is one where changes in latent factors correspond to changes in high-level factors in the output space **?**.

To provide some intuition of what a disentangled representation is, we'll consider an example using the Frey Face data set.

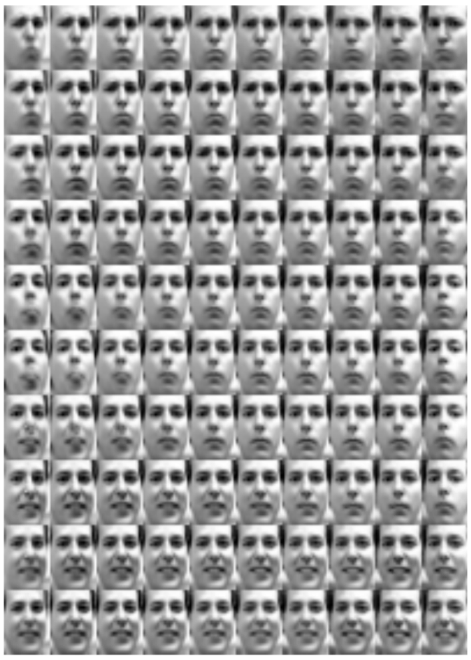


Figure 3: A learned Frey Face manifold with two latent variables: one sensitive to the expression and one to the orientation. **?**

The Frey Face data set consists of almost two-thousand 28×20 pictures of Brendan Frey, a Canadian machine learning researcher. The samples include Frey with varying facial expressions and orientations.

Shown in Figure (**??**) is a learned Frey Face manifold with two latent variables. One latent variable is sensitive to changes in the expression and one to changes in orientation, making the manifold a highly disentangled representation. In an entangled representation, changing one latent variable would change both the expression and the orientation.

Disentangled representations facilitate the reuse of previously learnt factors, called *knowledge transfer*; and the extreme case, *zero-shot inference* **??**.

2.3.2 Transfer learning

Transfer learning is the reuse of learnt factors from a previous task when learning a new one **?**. If the tasks are sufficiently similar, there are three ways the learning may be improved **?**:

1. Initial performance
2. Faster to learn
3. Final performance

These improvements are shown on a performance-training graph in Figure **??**.

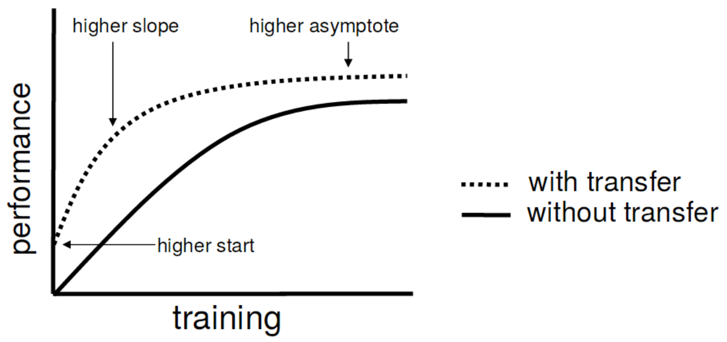


Figure 4: An illustration of the three ways in which transfer learning may improve the learning of a new task. **?**

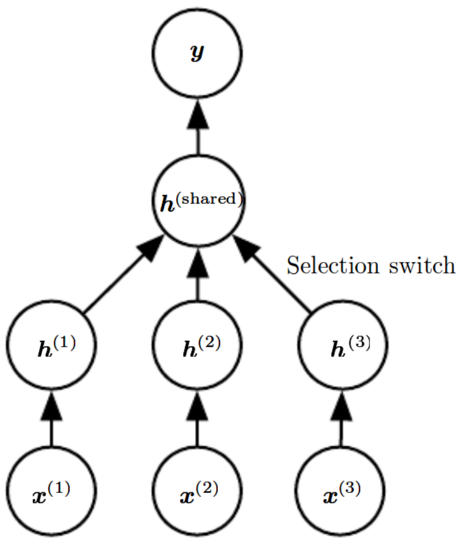


Figure 5: An illustration of how tasks may be translated into intermediate representations, then into a shared representation. The representation of different tasks in terms of common concepts is key to transfer learning. **?**

Entangled representations make it difficult to learn common generative factors among different tasks, as no one latent factor has meaning itself **?**.

2.3.3 Zero-shot understanding

2.4 Deeper investigation of β-VAE

2.4.1 Framework

Suppose we're given a set X of images $\boldsymbol{x} \in \mathbb{R}^N$ and ground truth data generative factors (that is, factors observed directly from the data as opposed to being inferred). We'll partition these factors into two sets, V and W, where vectors $\boldsymbol{v} \in \mathbb{R}^K$ are independent and $\boldsymbol{w} \in \mathbb{R}^H$ are dependent.

The aim of β-VAE is to factorise the latent space $\boldsymbol{z} \in \mathbb{R}^M$ into disentangled factors $\boldsymbol{v}$ and the rest. This enforces $M \geq K$, since we'd like the latent representation to be capable of learning all independent ground truth data factors of X.

β-VAE achieves this by adding an important multiplicative factor β to the KL-divergence term, making the objective:

$$\max_{\phi,\theta} \mathbf{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta \mathcal{D}_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})] \tag{2}$$

The hyperparameter β balances reconstruction error with independence constraints **?**.

2.4.2 Disentanglement metric

2.4.3 Quantitative performance

2.4.4 Qualitative performance

2.5 Libraries

2.5.1 Keras

Keras is a high-level neural network library written in Python **?** and was used to implement the frameworks in Chapter X. It was a suitable choice because it supports:

- Convolutional, deconvolutional and pooling layers
- Lambda functions, which is necessary for sampling in variational autoencoders
- Custom loss functions, which is necessary to implement the β-VAE framework
- Able to save weights, so the same network can be trained on multiple games
- Able to run on the GPU

2.5.2 Arcade Learning Environment

The Arcade Learning Environment (ALE) is a framework built on top of the Atari 2600 emulator Stella **?**. This library was a suitable choice because:

- Emulation details are abstracted away from the researcher
- The same agent can be used for all games, due to its modular design
- Frames can be saved during game play, which is used to collect training data
- A Python wrapper for the entire framework is provided, so our agents developed in Keras interact seamlessly

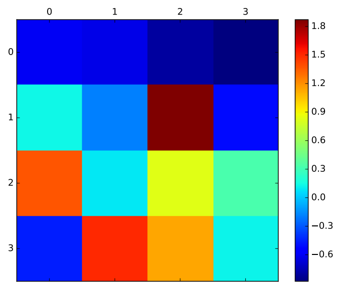
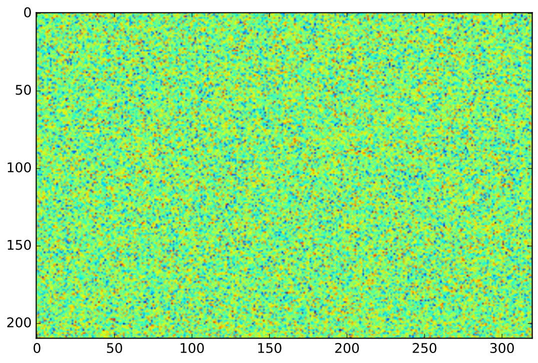
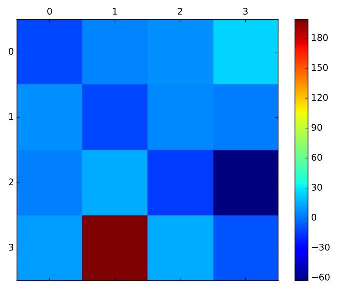
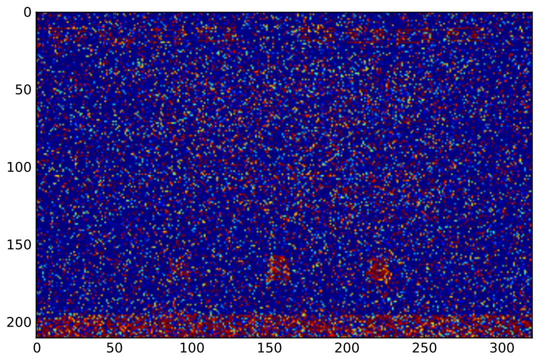
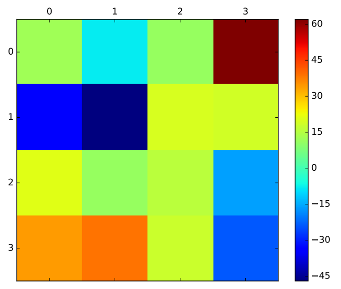
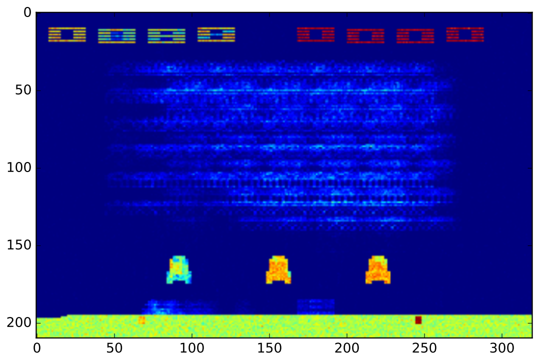
3 Results

The ALE was used to capture frames from the emulator Stella at regular intervals, which were dumped into a common location, forming the training set. A convolutional variational autoencoder was then implemented in Keras to train on this data. A number of scripts (that partition the data into training, validation and test sets, for instance) were used in creating generic data sets that are easily loaded into Keras.

The VAE was used to generate new samples from the test set using a sample input, shown in Figure (**??**). The sampled latent space and its reconstruction are shown in Figure (**??**) for different epochs. As expected, the reconstruction improves with the number of epochs.

Interestingly, the VAE has already learnt that the upper triangler section of the enemies is more dense than the lower half. It also has picked up on persistent objects, like the three barriers in the center and the floor.

Table 1: The latent space and reconstruction for 10, 50 and 100 epochs given Figure (**??**) as input.

Latent space	Reconstruction
	
	
	

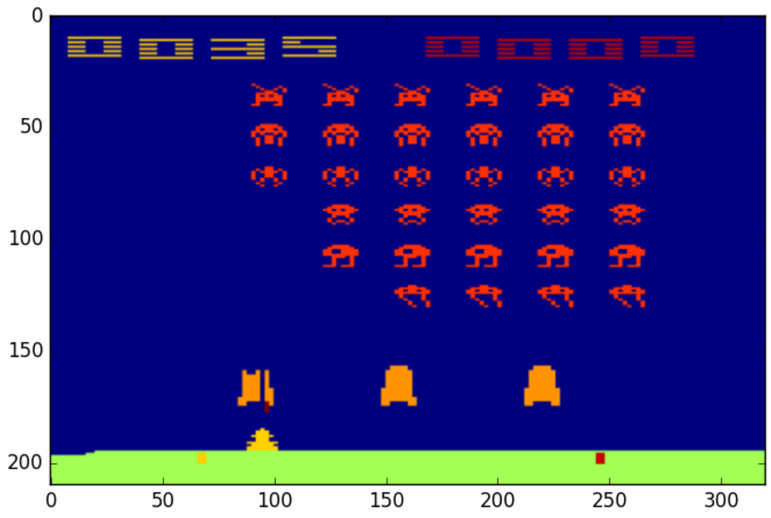


Figure 6: A screenshot from Space Invaders running in the ALE.

4 Project Plan

The rough plan for the project is given in Table (**??**). Since the method we will use to evaluate the disentangled representations is still in development, it's not yet possible to assign specific tasks for its implementation. However, the key milestones not dependent on the evaluation have been included as a skeleton to build on in the coming months.

Table 2: Project plan in terms of remaining months.

	Milestones	**Fallbacks**
February	• Implement β-VAE • Reproduce results from **?** • Finish formal justification of VAE	• Implement β-VAE • Reproduce results from **?**
March	• Disentanglement metric script • Test script with simple data	• Disentanglement metric script • Test script with simple data
April	• Find suitable β for Pong • Find suitable β for Space Invaders • Graph latent vs output	• Find suitable β for Pong • Find suitable β for Space Invaders
May	• Complete report background • Complete report related work	
June	• Complete report results	

The achievements to date are covered in Chapter **??**, and the milestones set out for (at least the immediate months) are a feasible goals. If we are ahead of schedule, the following extensions would be interesting to explore:

- Train an agent on multiple similar games (Pong and Space Invaders, for instance) to demonstrate the ability of a disentangled representation to facilitate transfer learning, as listed in Subsection **??**.
- A *generative adversarial network* could be used as the anomoly detection network (described in Chapter **??**), instead of a regular feed-forward network.
- We could investigate the way the variational autoencoder explores its statespace. To explore its statespace, the autoencoder can endlessly recombine combinations of disentangled factors to produce new samples that may or may not occur in the real worl. For instance, we know not all things that are red and yellow are strawberries and bananas. We can therefore pry apart these two generative factors into a colour and an object. We can then imagine a red banana and a yellow strawberry even though we've never seen either before.
- Using the previous concept, we could experiment with the agent performing offline-exploration or "look ahead" in an Atari game.
- Try using 3D convolutions in encoder and decoder to see how temporal information could be encoded **?**. It may also be worthwhile to see if this improves any elements in reconstruction error or learning rate.

5 Evaluation plan

There are two ways in which the representations found may be evaluated: how realistic the generations seem and the degree in which the latent space is disentangled.

The decoded images from the sampled disentangled latent space should be as realistic as possible. However the task of visually examining multiple decoded images from multiple alternative latent spaces is time consuming. To automate this process, we could implement an anomoly detection network, which is trained on the whole space of input images. This can be used in the following procedure: train a VAE for a given β, then generate unseen samples from the dataset and run each through the anomoly detection network. The more images that pass, the higher the score. Performing exponential search on $\beta > 1$ would find a suitable disentangled representation without the need for time-consuming visual inspection.

As well as using the disentanglement metric introduced in **?**, we may quantify how disentangled the latent space is by the following procedure: create an artificial data set varying one generative factor (the paddle in Pong, for instance), and measure the change in latent variables for each of these as inputs. We'd expect a perfectly disentangled latent space to have exactly one latent variable changing value, and remain constant for other data sets changing other generative factors (the ball in Pong, for instance). The changes may be averaged over the number of inputs to give a final disentanglement score.

The project would be a success if a suitable range of values of β were found such that the generative features in multiple Atari agents were captured, and the benefits of this representation were demonstraed by an agent being able to re-use the same concept in a range of games.