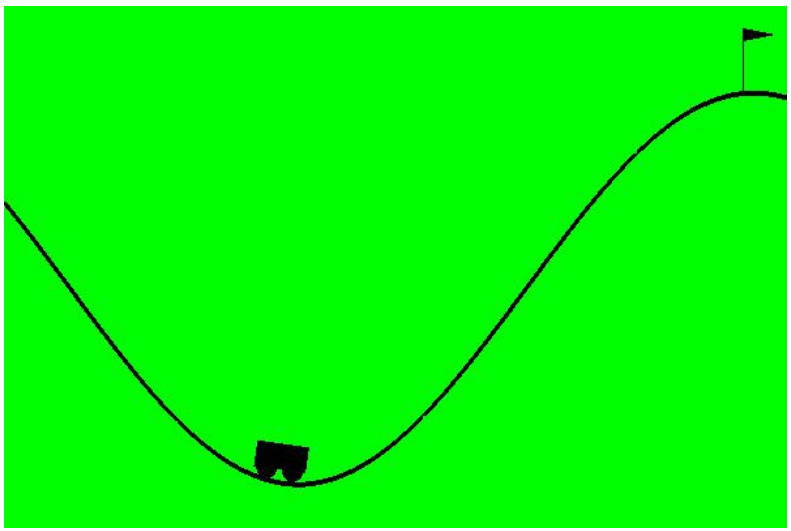
Transfer Matching Networks

Siddharth Nayak Abhishek Nair

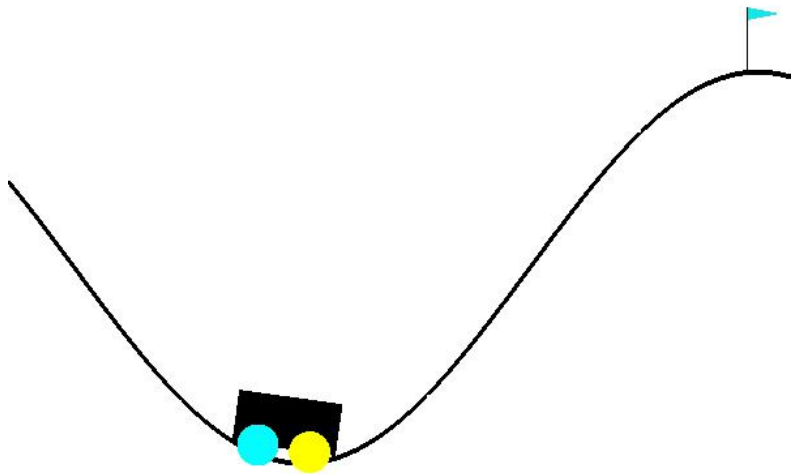
Overview

The Problem Statement

- We aim to transfer the learning of an RL agent trained on one environment to another environment without additional training.
- Achieving this requires finding a common representation between the two environments.



(a) Original Environment



(b) Modified Environment

Figure: Comparison of environments

Figure: The Coinrun environment by OpenAI. Each level can have different backgrounds but the core structure of the levels remain same. The objective is to learn the common representations of the game.

Advantages

- Sparse rewards lead to credit-assignment problem.
- A supervised learning approach for learning common representations can help in leveraging the knowledge acquired.
- The common representation learnt will capture the core structure of the state thus reducing the state representation size.
- Can be used with any RL algorithm without any overhead.

Notation and Terminology

- We will denote the original environment by orig_env and the environment on which we want to transfer the learning by target_env.
- The states from orig_env are denoted by s_1 and that from target_env are denoted by s_2.
- We have two autoencoders: autoencoder 1 ($\mathcal{A}_1$) and autoencoder 2 ($\mathcal{A}_2$)
- The latent representation of the autoencoders (middle layer) are denoted by z_1 and z_2 for $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively.

Autoencoder

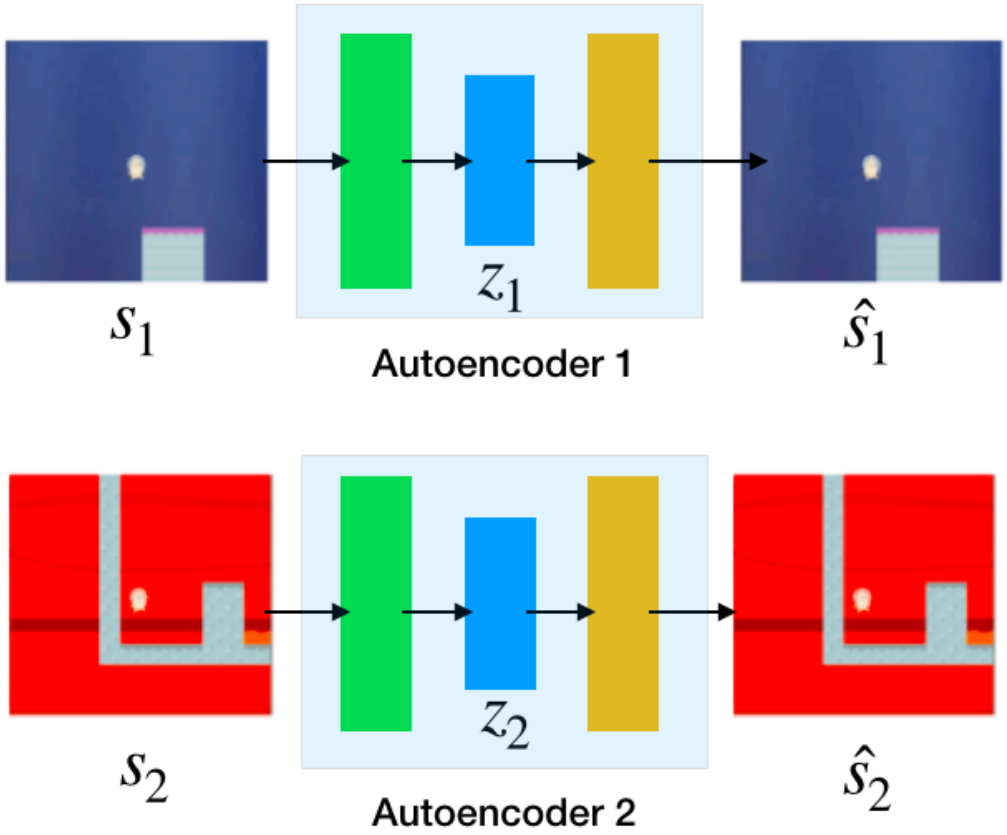


Figure: Autoencoders can be linear as well as convolutional depending on the state representation of the environment

Loss Function

We want to constrain the autoencoders to learn the core structure of the original and target environment.

$$\mathcal{L} = |s_1 - \hat{s_1}|^2 + |s_2 - \hat{s_2}|^2 + |z_1 - z_2|^2 \tag{1}$$

Initial Experiments

We try out our method on the cartpole environment.

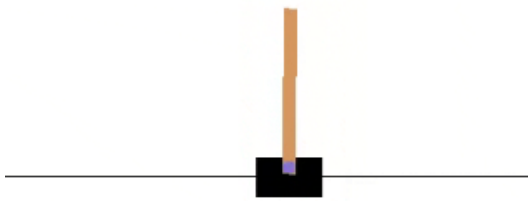


Figure: Cartpole Environment by OpenAI

The state for the Cartpole environment is $[x, \dot{x}, \theta, \dot{\theta}]$ where,

- x: cart position
- $\dot{x}$: cart velocity
- θ: pole angle
- $\dot{\theta}$: pole velocity at tip

- State for the original environment: $(x, \dot{x}, \theta, \dot{\theta})$
- State for the target environment: $(\dot{\theta}, \theta, \dot{x}, x)$

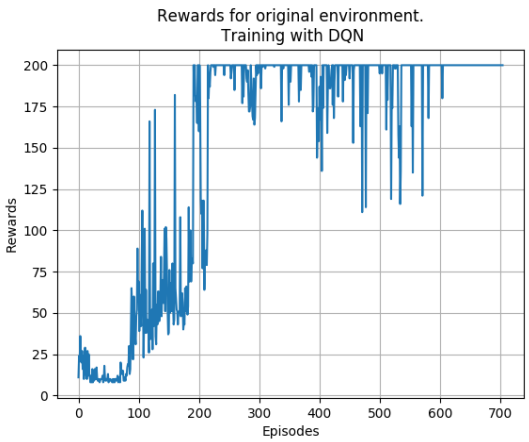


Figure: Rewards while training on the orig_env

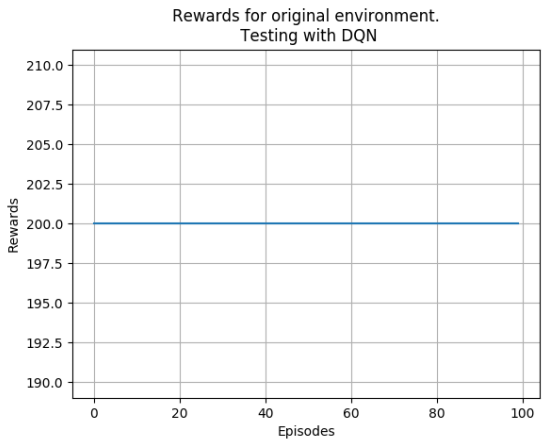


Figure: Rewards while testing on the orig_env

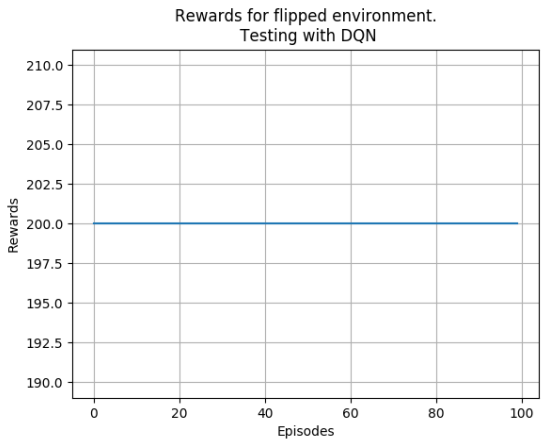


Figure: Rewards while testing on the target_env

Mountain Car Environment

State for the Mountain Car environment is: $[x, v]$ where,

- x: car position
- v: car velocity

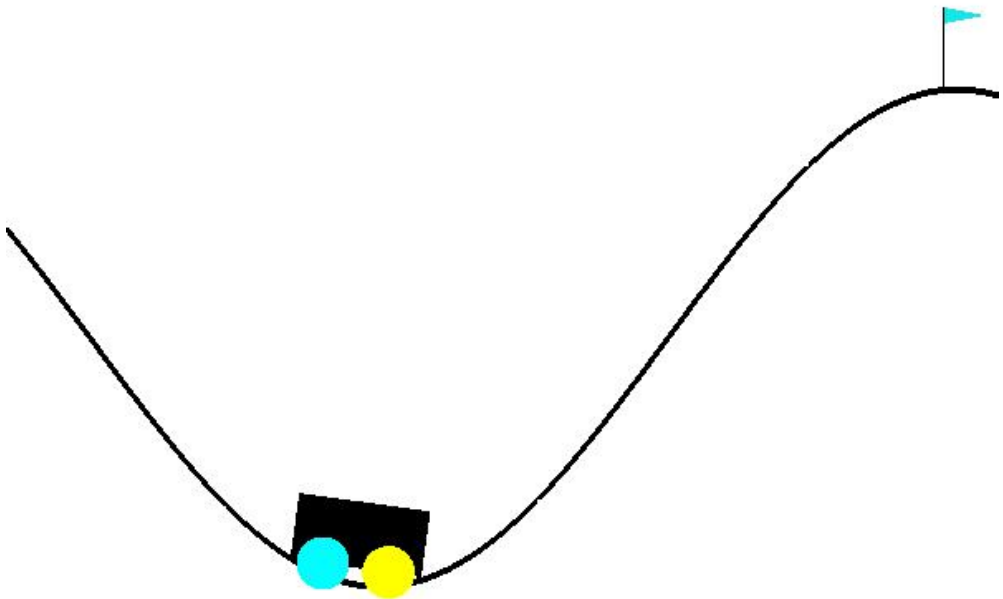
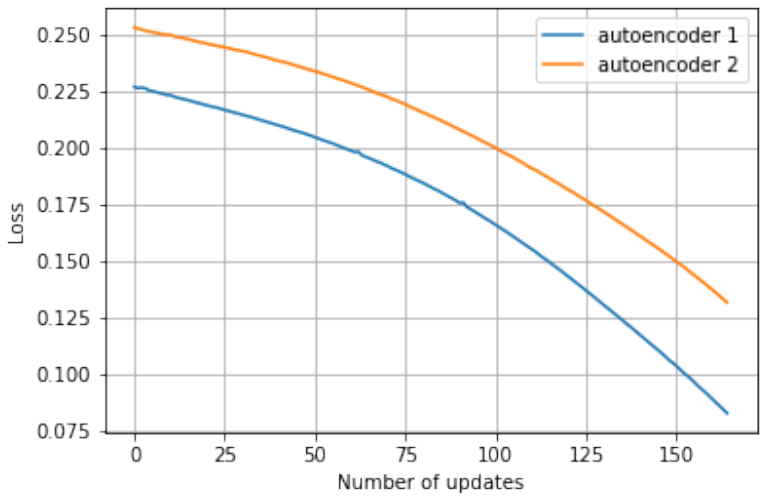


Figure: Mountain Car Environment

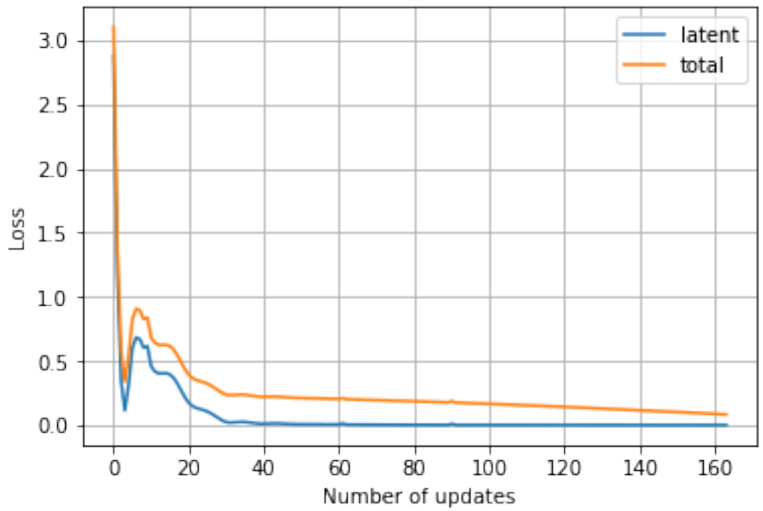
Mountain Car Environment

- We extract the frames and use the images obtained as inputs to autoencoder.
- We modify the background color, color of individual wheels, and the color of the flag.
- We use convolutional autoencoder.

Results with Mountain Car



(a) Autoencoder loss



(b) Latent and total losses

Figure: Have to train more! This is just with one episode of information

Future Work

- Experiment with environment with major structural changes.
- Run further experiments with Mountain Car and Atari environments.