Learning Model-based Representations for Reinforcement Learning CS6700: Course Project Report

R S Nikhil Krishna
Department of Mechanical Engg.
IIT Madras
rsnk96@gmail.com

N Sai Kiran
Department of Electrical Engg.
IIT Madras
saikirann94@gmail.com

Santhosh Kumar R
Department of Electrical Engg.
IIT Madras
santhoshkumar.25dec@gmail.com

Abstract

In this project, we focus on learning useful representations for Reinforcement Learning tasks. Some of the useful properties that we try to introduce in these representations include generalizability across different tasks and disentanglement of the different features learned. We also try to ascertain the advantages of these features over features which are learnt from reward-based objectives. We show that such representations are often more generalizable across tasks and are superior to reward-based representations learnt on target tasks. We experiment with the β-VAE as a possible model to learn disentangled representations conducive to control, and investigate the reasons for its poor performance on Atari tasks.

CHAPTER 1: Transferring Model-based Representations for Atari Games

Introduction

In Deep Q-Networks (DQNs), the image representation is learnt from the Q-learning objective which is based on the rewards provided by the environment. Are reward-based representations the best kinds of representations we can learn? In this section, we determine the efficacy of model-based representation learning, i.e, learning representations which aid in predicting the future states given the current state and the action. More specifically, we want to analyze the generalizability of such representations. This is done by transferring the different representations learnt from one atari game (source) to another (target) and measuring the performance of the agent on the target task. Note that more general representations tend to perform poorly on the particular task they are trained on. This is because the agent is unable to overfit to that task.

We attempt different modes of transferring knowledge learnt by an agent from one game to another.

1. Train a vanilla DQN for one task and transfer the representation directly to another task.
2. Train a model for predicting next state from current state and action. Transfer the representation learnt from one game to another.
3. Train a hybrid model consisting of vanilla DQN training with model prediction based regularization over the representations learnt.

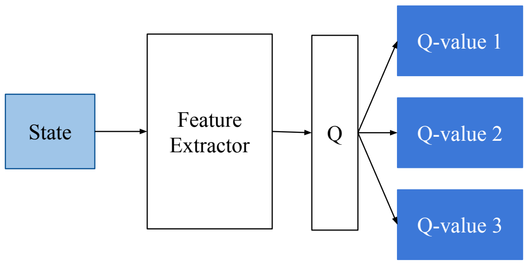

Figure 1: Vanilla DQN architecture

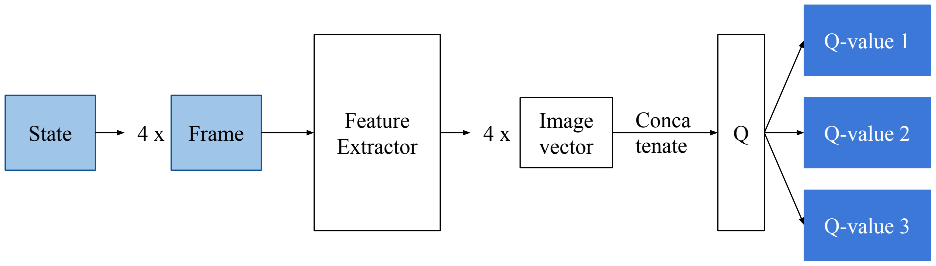

Figure 2: Modified DQN architecture

Learning model-based representations

For learning model-based representations, we initially introduced a model-based regularization cost to a modified version of DQN. Instead of performing a convolution across four consecutive frames and obtaining a state representation, we obtain an image representation for each frame and concatenate them to obtain the joint state representation as shown in Figure 2.

This would enable us to impose a model-based regularization where the regularization cost is the error in predicting the image representation of the 5^{th} frame from the first four frames (current state) and the action taken at that state, i.e,

$$\text{Regularization cost} = ||f_p - f_t||^2$$

where f_p is the predicted image representation and f_t is the true representation of 5^{th} frame. However, the modified DQN did not learn to play the atari games effectively (even without regularization). Since the frames of the state were being processed individually, the joint information from the frames were not captured in the concatenated state representation. As a result, the agent was unable to select the actions correctly, leading to poor performance. In order to test this hypothesis, we added more non-linear decision layers on top of the state representation and trained the new DQN. We observed that the performance improved as expected, supporting our hypothesis. In Figure 3, we demonstrate the above issue on `breakout`. The performance of the modified DQN drops drastically and it improves when you add more non-linear decision layers, indicating that joint modeling of frames in a state is essential.

In order to retain the performance of the vanilla DQN and add a regularization on top of that, we formulated the model prediction as follows: Given a sequence of 5 states ($s_{t:t+4}$) and the corresponding actions taken ($a_{t:t+3}$), the model prediction involves predicting the state representation of s_{t+4} using s_t and $a_{t:t+3}$. Note that each state consists of 4 consecutive game frames and that the state representation is obtained jointly for all the frames using convolution. This avoids the above mentioned problem entirely, and also allows us to impose model-based constraints on DQN.

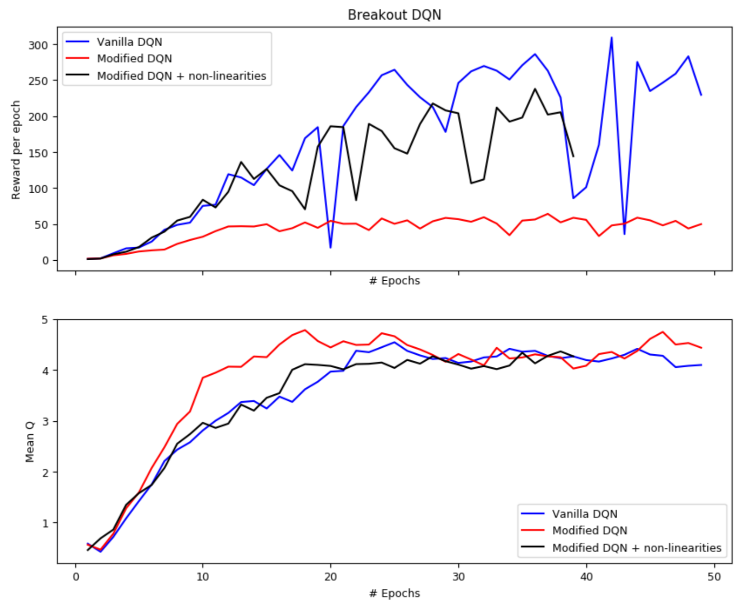

Figure 3: Results for vanilla DQN, modified DQN and modified DQN followed by extra non-linear decision layers. Shows evolution of test rewards and average Q values over `breakout` training epochs.

Experiments

To test the generalizability of different representations, we train the agent on a source task and transfer the learnt representations to target tasks. The different (source, target) pairs we have experimented on are shown in Table 1. For each of the pairs, we compare the performances of four agents.

- **Vanilla**: Vanilla DQN trained on target task with random initialization.
- **Vanilla Transfer**: Transfer representation from Vanilla DQN trained on source task.
- **Model Transfer**: Learn a model to predict next state given current state and action on the source task. Transfer representation to target task.
- **Regularized Transfer**: Learn a hybrid DQN model with both model-free and model-based (regularization) objectives on the source abd transfer to target.

Source	Target
River Raid	Boxing
River Raid	Space Invaders
River Raid	Centipede
Breakout	Pong
Breakout	Freeway
Space Invaders	Pong
Seaquest	Space Invaders
Pong	Breakout
Pong	Space Invaders

Table 1: Set of source-target pairs experimented on

First, we train Vanilla DQN, model-regularized Hybrid DQN and supervised future prediction models on the source tasks. Given in Figure 4 are the plots of the performances of the Vanilla and the Hybrid/Regularized DQNs on each of the source tasks. As we can see, the vanilla agent performs better than or comparably to the regularized agent. This is because the regularized agent is not allowed to overfit on the source task.

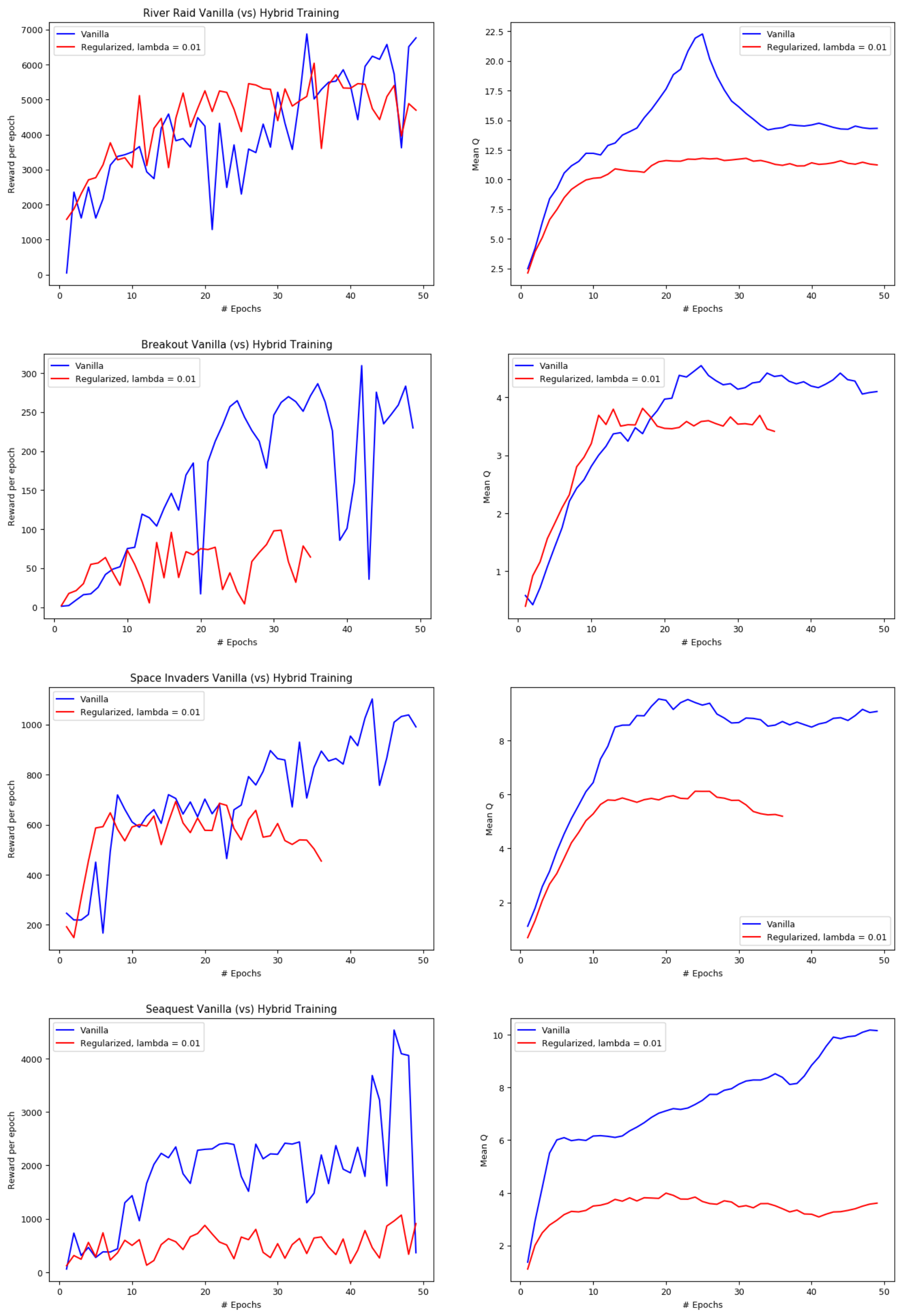

Figure 4: Vanilla DQN and Regularized DQN performance on source tasks of `River Raid`, `Breakout`, `Space Invaders`, `Seaquest` and `Pong`.

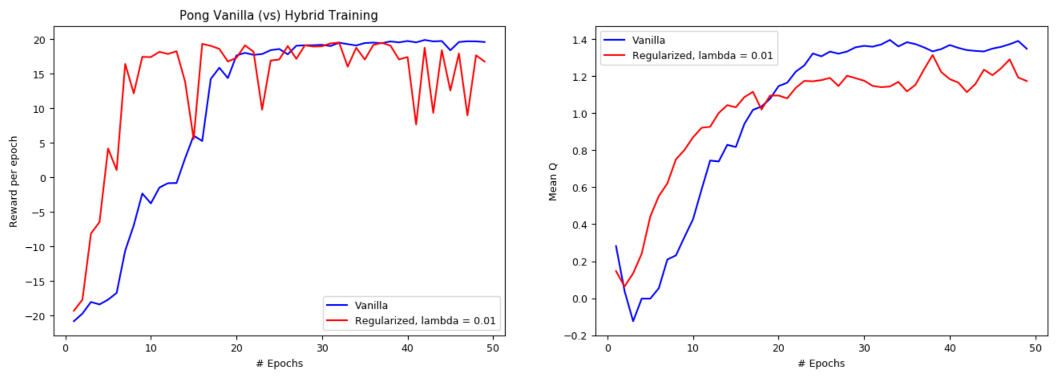

Figure 4: Vanilla DQN and Regularized DQN performance on source tasks of `River Raid`, `Breakout`, `Space Invaders`, `Seaquest` and `Pong`.

Next, we transfer the learned representations and train the vanilla, vanilla transfer, model transfer and regularized transfer agents for each of the source-target pairs. The corresponding plots are shown in Figure 5. In many of the cases, the model transfer seems to perform comparable to or better than the vanilla model itself, especially in the cases of `Space Invaders` from `Pong` and `Freeway` from `Pong`. Model transfer also outperforms vanilla transfer on most of the games. This confirms our hypothesis that model-based representations can be more effective amd generalizable in comparison to using randomly initialized representations and transfer from reward-based representations. However, the hybrid learning (regularized DQN objective) failed to obtain positive transfer in most of the cases. It might be possible that the training using both the objectives was not stable and lead to representations that were not generalizable.

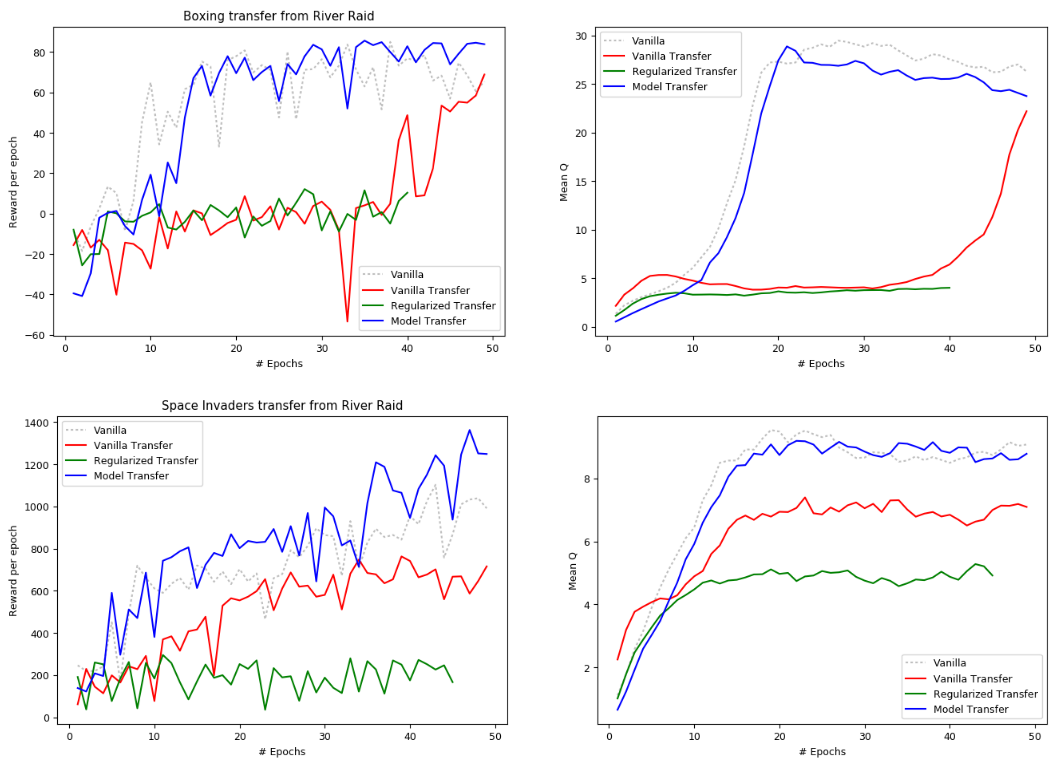

Figure 5: Comparing performances of 4 types of agents on different source-target pairs.

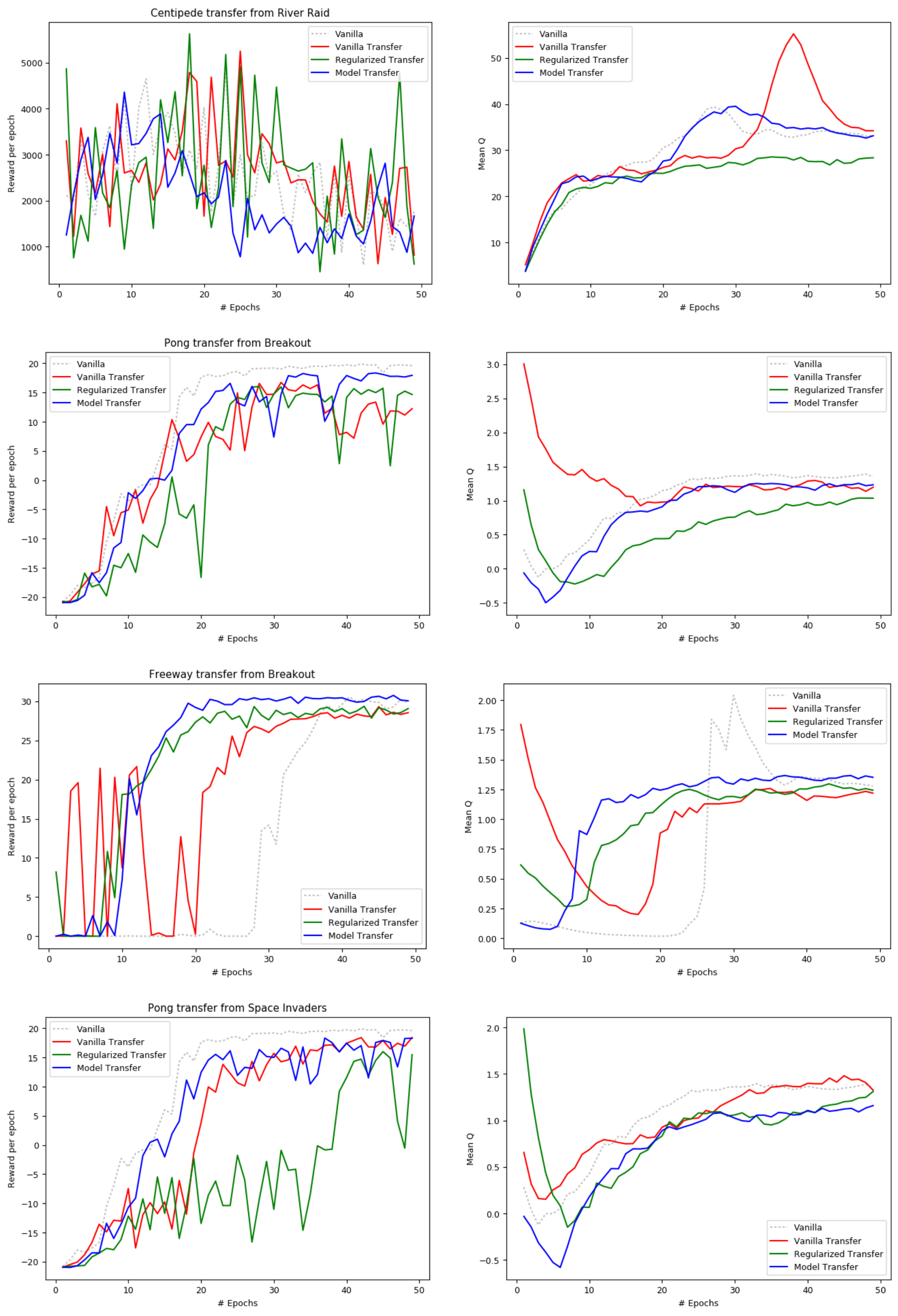

Figure 5: Comparing performances of 4 types of agents on different source-target pairs.

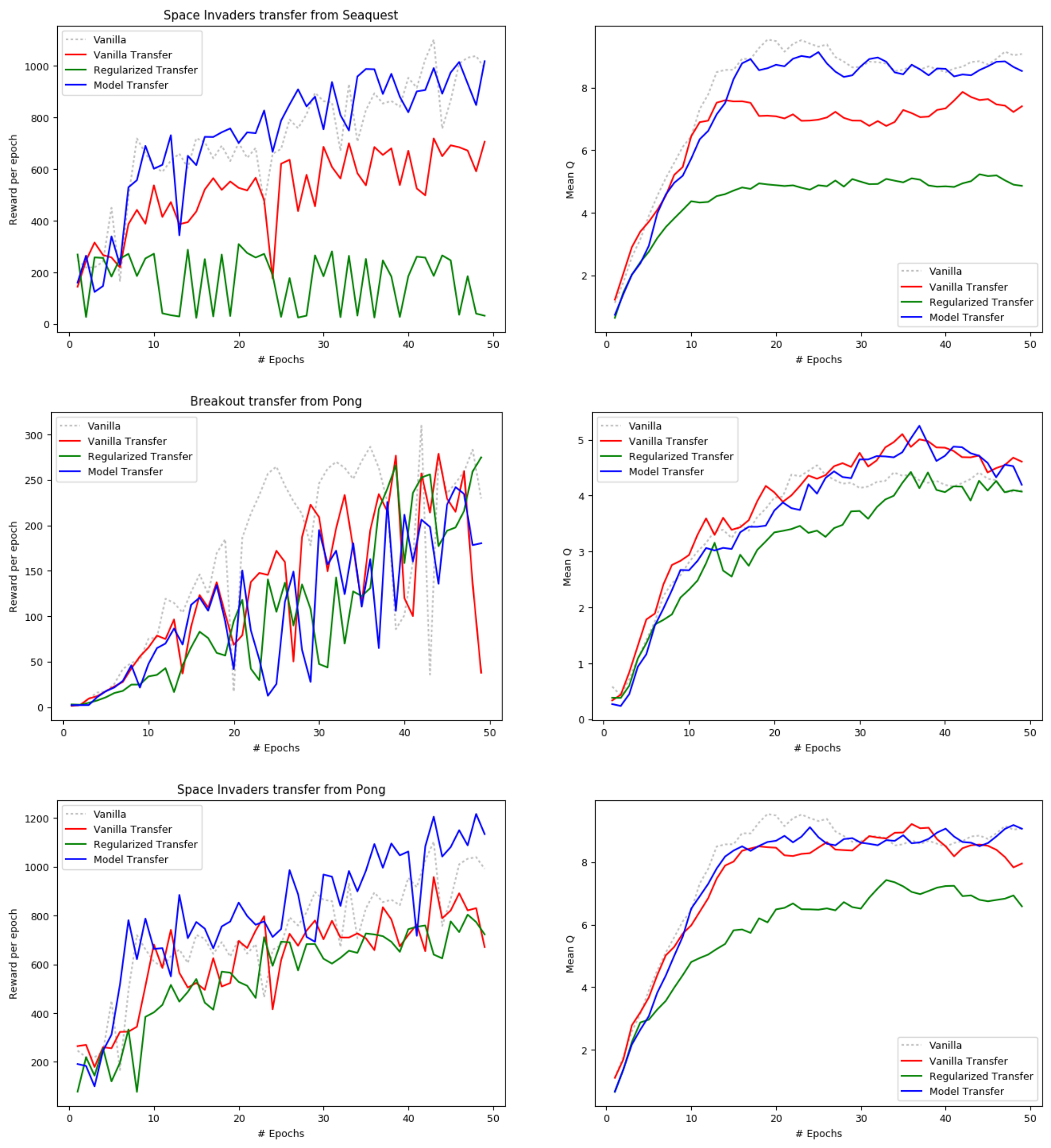

Figure 5: Comparing performances of 4 types of agents on different source-target pairs.

CHAPTER 2: Disentangled representations for control

RL agents in many methods learn end to end, from raw observations to policies or value functions. DQNs for instance take in raw image pixels and learn Q values directly through a deep neural network. These values can be viewed as some learned representation of the observations that are directly usable for control.

In this section, we investigate some possibilities for decoupling the representation learning from the decision making in RL. That is, we attempt to learn general features of the observations that are not specific to the RL problem. Examples of such features would be the coordinates of the ball, or the position of the paddle in the Atari game Breakout. An RL agent can then be made to operate using these features and perform control. The benefits of such a scheme are discussed below.

Advantages

- The learned representation could be reused if the dynamics of the system change. For example, a robot could use the same representation to perform different tasks with different rewards in the same environment. Only the decision layer would need to be learned for each new task.
- Such representations will likely have significantly reduced dimensionality. As a consequence, learning would be faster because simpler models that are easier to train can be used, and data would be used efficiently.

Of course, the representation learning technique would have to produce representations that are amenable to control. [4] uses Deep Spatial Autoencoders to learn features that describe the environment in which a robot is placed, which it uses to learn tasks using efficient RL methods. However, for Atari tasks, we turn our attention to β-Variational Auto Encoders (VAEs) that have been shown to learn disentangled representations from images [5]. Disentangled representations involve latent variables that encode different and independent causal factors that generate the image. They are similar to the kind of representations humans might use, and naturally are expected to be conducive to control.

We test the effectiveness of β-VAEs in learning such representations in the following experiments.

Experiments :

We use a β-VAE with the following architecture for the encoder :

- First convolutional layer with 64 4×4 filters that operates on 84×84 rescaled screen images (in 3 channel RGB format).
- Second convolutional layer with 32 3×3 filters.
- Encoder outputs (mean and covariance of $Q\left(z|x\right)$, the latent distribution) found using a linear layer.

The generator has the following architecture :

- Up projection layer operating on z sampled from the conditional latent distribution, with a ReLU non-linearity.
- Two transposed convolution layers reversing the first two layers of the encoder, with ReLU non-linearities.
- Final sigmoidal output layer that gives the generated image.

The cross-entropy loss function was used to train the β-VAE on frames obtained from an expert DQN agent on Breakout and Pong. $\beta = 1.28$ with 30 dimensional latent space was used for Breakout and $\beta = 1$ with a 5 dimensional latent space was used for Pong. For Pong, the score bar displayed in the top of the screen was blacked out to check if the reduced noise would help focus on the important parts of the image and learn a better representation with a smaller latent space (only 4 generative factors are present in the Pong frames).

For visual inspection of the features learned, the following experiment is performed: a random frame is encoded, and each latent variable is perturbed around its original value. The amount of perturbation is up to three standard deviations of the latent means $\sqrt{\mathrm{var}\left(\mathbf{E}\left[Q\left(z_i|x\right)\right]\right)}$ (estimated from the values of the latent variable for several random images). The resulting latent space "images" are sent through the generator to observe the effect of each latent variables. The results of this experiments on Breakout and Pong models are presented in Figure 6. Only the 5 latent variables with the least expected conditional variance $\mathbf{E}\left(\mathrm{var}\left[Q\left(z_i|x\right)\right]\right)$ are presented as they would have the most informative features as described in [5].

It does indeed manage to encode several generative factors such as the paddle in Pong and the bricks in Breakout (first latent variable).

We tested the effectiveness of these features for control by training a DQN using these features as input. The DQN itself uses a simple network with two decision layers applying ReLU non-linearities. The performance turned out to be as bad as random with no improvement in average reward per episode at all for both Pong and Breakout. We also tested a version of this for Pong where the latent representation is learned over the past 4 frames (though adding history defeats the purpose of being independent of system dynamics). Even here, performance was as bad as random. In conclusion, the vanilla VAE framework appears to give poor feature generation for control.

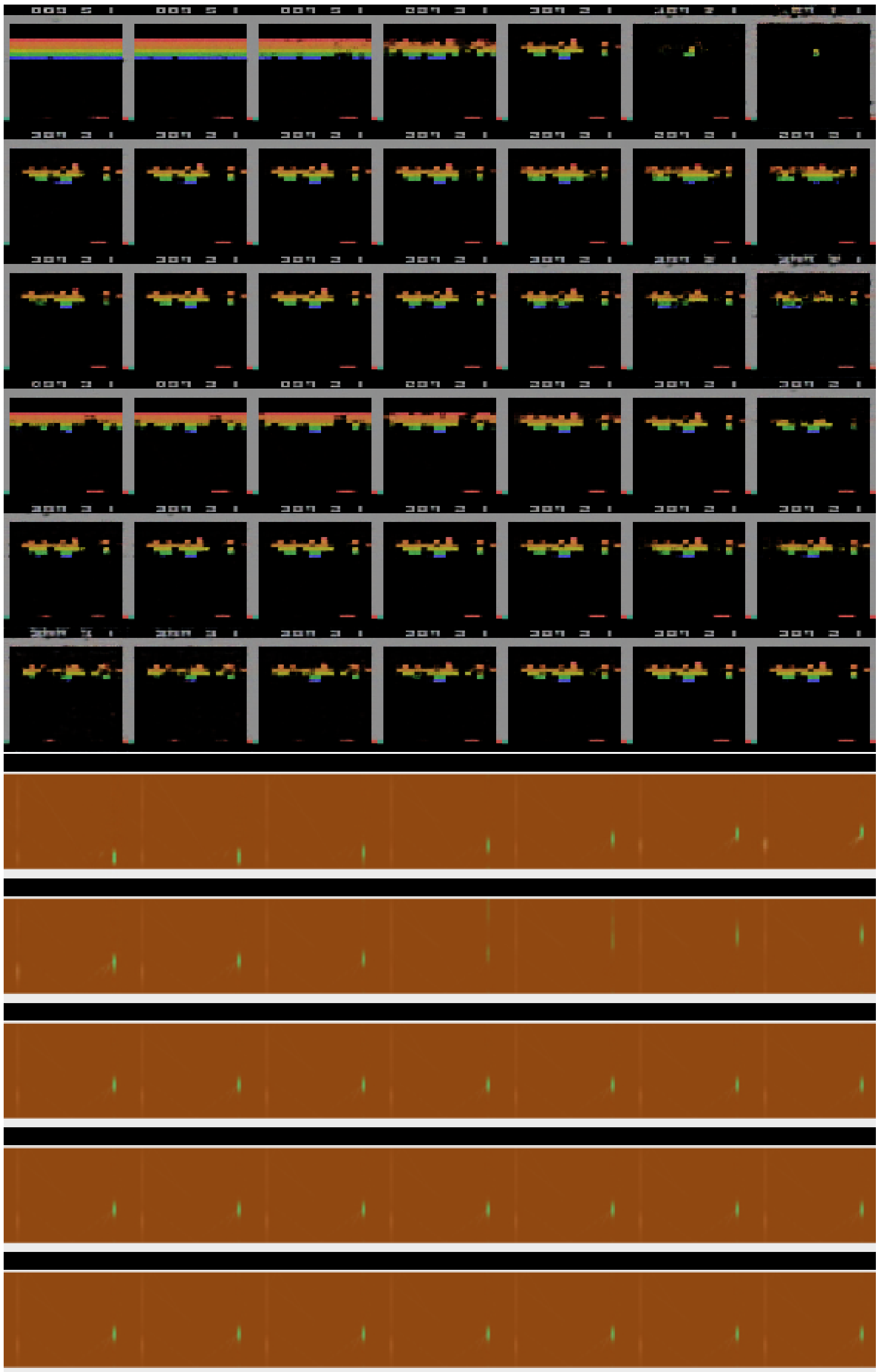

Figure 6: Reconstructions of frames by β-VAEs for Breakout(top) and Pong(bottom). Each row is for a latent variable (increasing variance from top to bottom). In each row, the middle image corresponds to the unperturbed image, and the i^{th} image to the right(left) corresponds to a perturbations of $i(-i)$ standard deviations as described in the text.

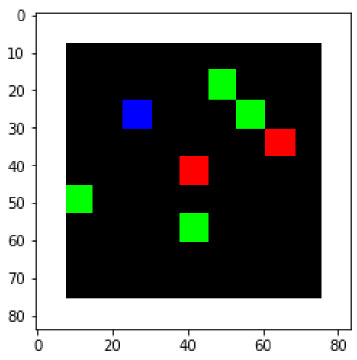

Figure 7: A sample frame from the modified gridworld

CHAPTER 3: Analysing the beta-VAE and Transferability of Model Based Regularization

In Chapter 1, we observed how in most cases, model based regularization on the source task helps in achieving more generalisability of the model learnt. In the first subsection of this Chapter, we test how good the generalisability is on a modified GridWorld, with modifications to the formulation of the network architecture and subject to different types of target tasks

Similarly, in Chapter 2, we observed how sampling on the latent space of the beta VAE performed much worse than expected. In the second subsection of this Chapter, we attempt to look at possible causes for the same, and how they can be fixed.

Transferability of Model Regularization on Modified Gridworld

The normal gridworld is too easy a task for a DQN, and hence it might not be suitable for benchmarking the transferability of model regularization using Deep Q Networks. Hence, the gridworld in use for the following set of tasks is a modified gridworld, as seen in Figure 7. In every episode that training is performed upon, the start position of the player, positive, and negative rewarding cells varies. Furthermore, there is not one goal, but 4 goals, each of which give a +1 reward, and 2 cells which give a -1 reward (hereafter referred to as ditches). There is no penalty for a step taken into an empty cell. When the player steps on a goal or a ditch, another corresponding goal/ditch is respawned at a random empty location in the grid. Hence, the objective is to maximize the total reward obtained by going through as many goals as possible, as is the case with many Atari games.

However, unlike the tests performed in Chapter 1, which followed the DQN nature paper's convention of using grayscale images, here the full colour images are used. The player is always blue. In the source task, the goals are marked as green squares, and the ditches as red squares. In all of the tested transfer tasks, the colour scheme for the reward distribution is inverted, i.e., red for positive rewards, and green for negative rewards. The reason for choosing this manner of generating the gridworld is that it should be easy for the model to learn to distinguish the different objects of the game since the rewards lie in different channels, as suggested by Aravind.

DRQN

Initially, a DQN was used for training the same. However, due to the fact that the normal DQNs take a lot of memory, a Deep Recurrent Q Network (hereafter referred to as DRQN) was instead used to enable experimentation on more tasks simultaneously. DRQNs were introduced to solve POMDPs, like Atari games. However, it comes with the added benefit of having a replay memory that is significantly smaller, as the state can be defined with just one frame and the hidden state of the LSTM, rather than by stacking four frames. Hence, a recurrent connection enables training using lesser resources.

The overall regularized loss can be formulated similar to the loss formulation in Chapter 1, except that states in the model loss are not of the 5^{th} frame, but of the hidden state at the next time instant. This lets us formulate the total cost as

$$\text{Loss} = (Q_{target} - Q)^2 + \lambda_{reg}||h_{p|k+1} - h_{k+1}||^2$$

where $h_{p|k+1}$ is the prediction at time t_k of the hidden state for time t_{k+1} and h_{k+1} is the true hidden state at time t_{k+1}

Two values for λ_{reg}, 10^{-2} and 10^{-5} were tested, in addition to using just the model loss. The first 500 episodes of training are used to build up the replay memory, and hence random actions are taken in these 500 episodes. The probability of taking a random action is 1 at the start of training, and this is uniformly decayed to 0.1 at the end of the 5000 episodes of training. As expected, the regularized models perform poorer on the source task, because they do are not overfitting to the source task

DRQN transfer

The transferability of model based regularization is tested by changing the task through multiple methods

1. Changing the formulation of the rewards itself - A colour scheme inverted gridworld is used as a testbed, where positive rewards are obtained for traversing on a red square, and negative rewards for landing upon a green square (quantity of cells giving particular type of rewards is the same, 4 Red Squares and 2 Green Squares). Model based regularization is seen to slightly worsen the performance when transfer is performed, as seen in Figure 8
2. Changing the reward formulation as well as changing the probability of selecting a random action along with corresponding decay - In addition to performing the colour inversion for the rewards of the destination task, the probability of selecting a random action is also changed. In the source task, the probability of choosing a random action begins at 0.5, and with a uniform decay rate, it decays to 0.005 by the end of the 5000 episodes of training. In the destination task, the probability of selecting a random action at the start of the training is 1, and with a uniform decay rate, it reaches 0.1 at the end of the training. Two unique observations can be made in this training from Figure 9
   (a) Model Regularized Training learns on the source task faster
   (b) Model Regularized Transfer outperforms vanilla transfer
3. Change the density of cells returning a certain type of reward - This was planned to be done by changing the number of positive/negative rewarding squares, as well as by making both colours represent the same reward. However, due to a shortage of time, this experiment could not be completed.

The fact that model regularized training works well in Case (2) and not in Case (1) is indicative of how although model based regularization might improve transferability, it is tough to train model regularized tasks such that it becomes beneficial

Sampling on latent space of the beta-VAE

Now we attempt to look at possible reasons sampling on the latent space of a beta-VAE and using it to train a DQN did not perform as well as expected

Verifying if the beta-VAE achieved disentanglement

The first task in this direction, which was performed at the very start of the Project, was verifying if the beta-VAE did indeed achieve disentanglement. This was done by simply training the beta-VAE on a modified MNIST dataset which included the original dataset as well as each element of the original dataset in a set of standard affine transformations. Twenty latent variables were used. To verify if disentanglement was achieved, translated, rotated and scaled versions of test data was fed into the encoder half of the beta-VAE alone, and the corresponding latent variable activations were recorded.

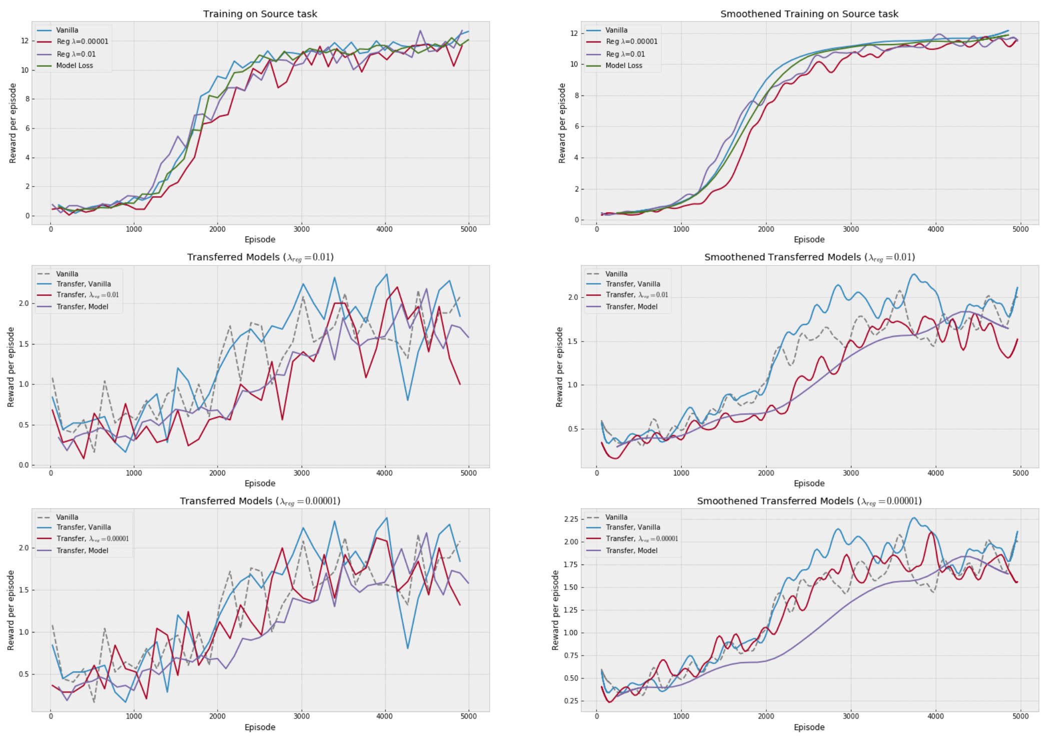

Figure 8: Transfer on the modified gridworld by inversion of colour scheme

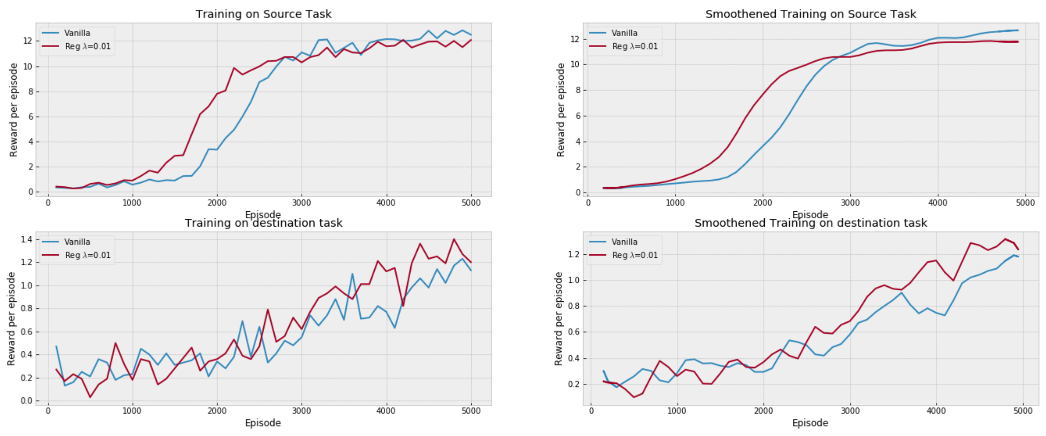

Figure 9: Transfer on the modified gridworld by inversion of colour scheme and by changing the probability of selecting a random action

It was observed that only a few variables for each type of transformation received significant activation, and the number of variables showing any activation also decreased as beta was increased, proving the hypothesis that using the KL loss enforces disentanglement, at least to the extent of accounting for basic transformations.

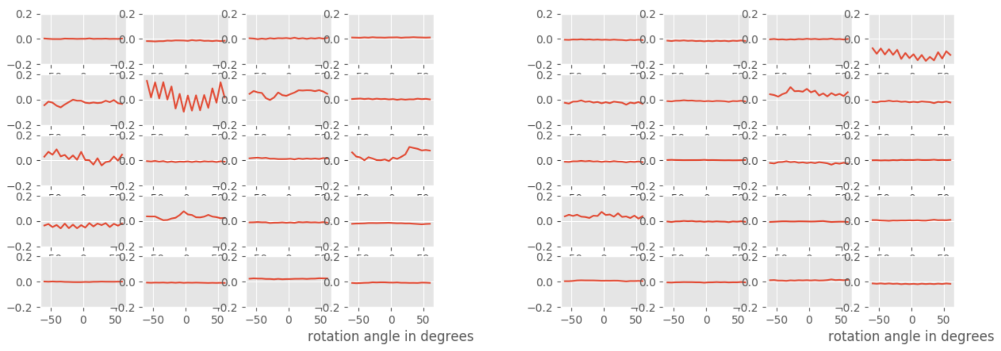

Figure 10: Activation of latent variables when the same image is fed in at different rotations at $\beta = 4.0$ and $\beta = 9.0$

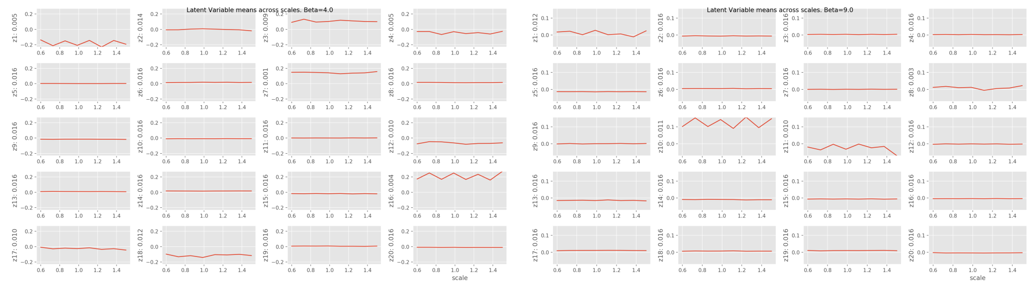

Figure 11: Activation of latent variables when the same image is fed in at different scales, cropped to the dimensions of the unscaled image at $\beta = 4.0$ and $\beta = 9.0$

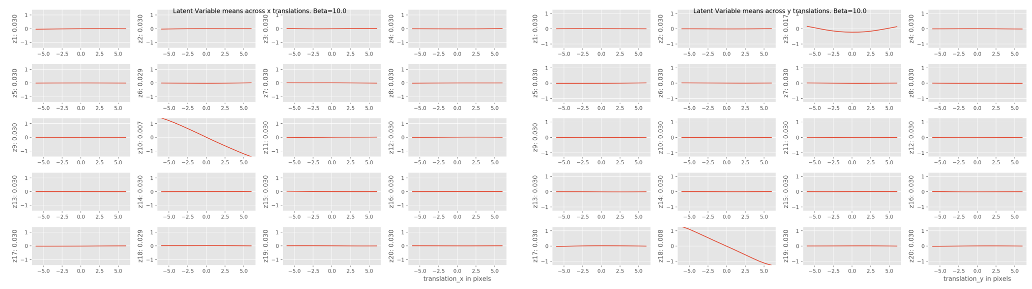

Figure 12: Activation of latent variables when the same image is fed in at different positions of x and y at $\beta = 10.0$

Similarly, the beta-VAE was also trained on expert Atari frames, and the generated outputs were observed when the latent space variables were perturbed. The results obtained through this experiment have already been covered in Chapter 2

Problems with the beta VAE being used for Model fitting

Upon manual inspection of the reconstructions generated by passing Atari frames through the beta-VAE, it was observed that the VAE had a tendency to encode only a particular set of features, neglecting others. This is significantly visible and consequential in cases like Pong, where the beta-VAE fails in encoding the ball even after achieving convergence in training. This can be attributed to the small size of the ball. When the Pong frames are resized to dimensions of 84x84, it was observed that in most cases, the ball occupied only a pixel, stretching to a 2x2 at the best. This is problematic as the filters encoding the features themselves have a minimum local receptive field of 3x3. Hence, we tried different changes to the architecture of the decoder in order to get the beta-VAE to encode the ball

Resolutions and scales of beta-VAE

Different methods of making the beta-VAE model the ball in Pong were attempted

- We first checked whether the problem was one that arose due to the generator's last convolutional layers not being learnt properly, as that too could potentially produce a blurred output, removing the ball from visibility. This was done by simply using a low resolution decoder, with the final transposed convolutional layer being removed so that it produces a half scaled output. However, this did not provide any useful information as the ball was absent in the images generated through the low resolution decoder too
- We simply expanded the latent space of the beta VAE itself, along with varying the value of beta, in hopes that it learns to encode the ball. This showed promising results, as the PSNR value of the reconstruction generated by passing an image through the beta-VAE was higher, as seen in the Figure below

  However, this came at the cost of poorer quality of samples generated by random sampling, as seen in Figure 13. This could easily be solved by adding a prior to the sampling procedure for the latent space. However, we did not follow through with this idea as we realized the very idea of expanding the encoder's output into a larger dimensional space beats the purpose of trying to achieve a smaller description length for our frames.

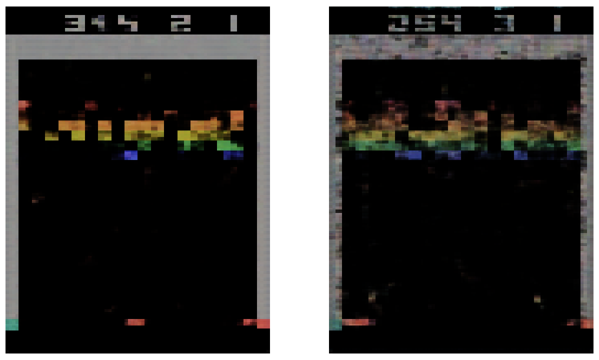

Figure 13: Frames generated by random sampling on latent space of beta-VAEs with 75 and 225 latent variables

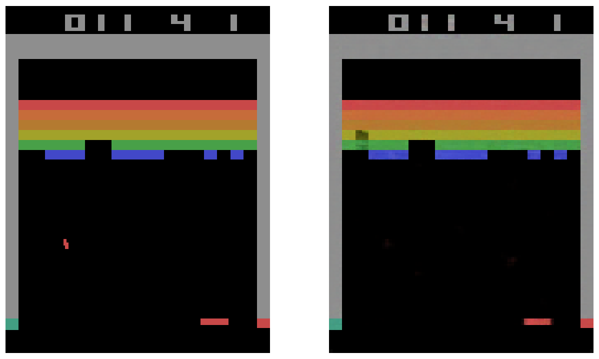

Figure 14: The true input frame vs its reconstruction by a beta-VAE with 75 latent variables

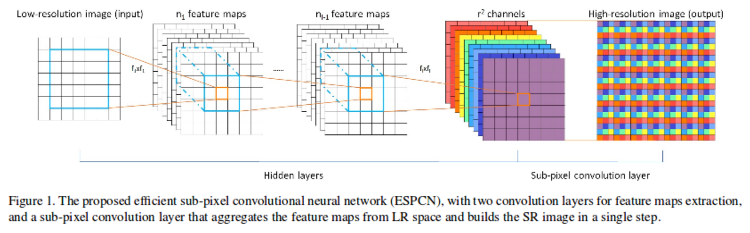

Figure 1. The proposed efficient sub-pixel convolutional neural network (ESPCN), with two convolution layers for feature maps extraction, and a sub-pixel convolution layer that aggregates the feature maps from LR space and builds the SR image in a single step.

Figure 16: The Efficient Sub-pixel Convolutional layer used for the generator, with the r^2 channels being replaced with $w \times h$ channels

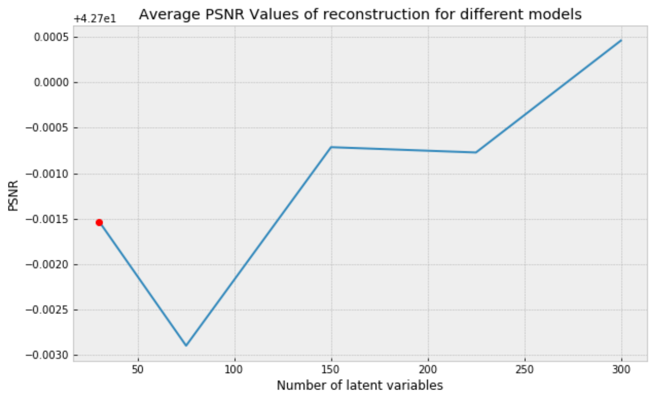

Figure 17: PSNRs (+dB) of the beta-VAE with different number of latent variables. The red point represents generation via ESPCNN. All other points use normal transposed convolutions for generation

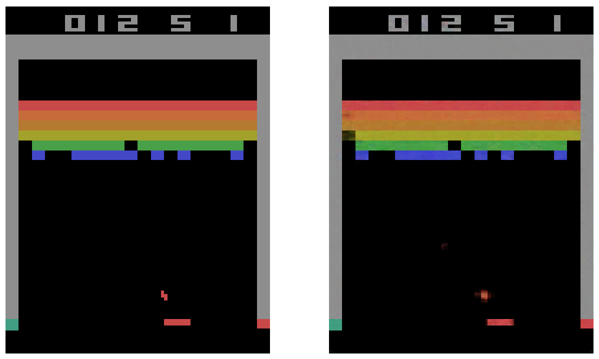

Figure 15: The true input frame vs its reconstruction by a beta-VAE with 225 latent variables

- We also tried simply changing the generator function being used to reconstruct the images from the latent space. For this, phase shift efficient super resolution, as seen in Figure 16 was implemented to replace the transposed convolutions that were being used in the generator. Although this improved the PSNR values of reconstruction, it was implemented too late to be integrated with the other sections of this project.

CONCLUSION

Although it was thought that sampling on a disentangled latent space for training DQNs would help in coming up with a generative method to train DQNs, the same is not reliable enough to replace training through expert frames. Hence, it is necessary to come up with better ways of both creating a latent space representation as well as sampling in order to effectively do the same.

It is also shown that the idea that reducing the dependence of the model learnt on the rewards obtained through a particular task could improve the generalisability and transferability of the learnt model. However, better methods of training need to be though of in order to make a model that can effectively work using this idea.

Contributions

- Santosh Kumar R - Chapter 1
- N Sai Kiran - Chapter 2
- R S Nikhil Krishna - Chapter 3

References

[1] Matthew Hausknecht, Peter Stone (2015) Deep Recurrent Q-Learning for Partially Observable MDPs

[2] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, Zehan Wang (2016) Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network

[3] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529-533.

[4] Finn et al. "Learning Visual Feature Spaces for Robotic Manipulation with Deep Spatial Autoencoders", CoRR abs/1509.06113, 2015 (http://arxiv.org/abs/1509.06113)

[5] Higgins et al. "Early Visual Concept Learning with Unsupervised Deep Learning", arXiv:1606.05579, 2016 (https://arxiv.org/abs/1606.05579)