Texture Synthesis and Style Transfer

Aman Kansal
170050027

Ansh Khurana
170050035

Kushagra Juneja
170050041

1. Introduction

refer abstract-intro from paper. Here we solve the problem of synthesizing a large image following the given input texture. Also, transferring the texture to another image.

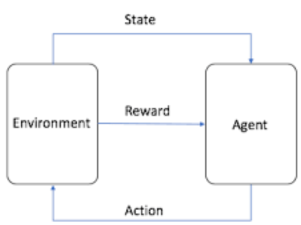


Figure 1: Reinforcement Learning Paradigm

2. Method

2.1. Quilting

Explain algo. Maybe add pseudocode in an algorithm block. Add figure with minimum boundary cut to explain the algo.

Figure 2: Boundary Cuts in Synthesis of Apple image

2.2. Texture Transfer

In the Approximate Q-Learning approach, the model the agent learns weights for features of states, where many states might share the same features. Function approximation learns the weights of different features during training. Using function approximation for parameterizing the problem of Q-Learning significantly reducing training time. It is important to recognize significant features on which the expected gain for the state will depend. For example, for approximating the Q-function for the Pac-Man game would depend upon features like Distance to closest ghost or dot, number of ghosts nearby, proximity to walls etc. Considering a linear model in the features, the update equations for the weights is given by:

2.3. Neural Network Style Transfer

Deep Learning does away with the need to handcraft features for the states. In the Deep Q-Learning based approach, the state is captured by an image of the current situation of the game. Deep Q-Learning is an extension of function approximation for Q-Learning. Deep Q-Learning uses a deep neural network to approximate the Q values for every action, given the current state. Thus, the output layer of the Deep Q-Network has dimensions = $|A|$ (The number of actions). The best action after learning the network can be taken by calculating an argmax over the output layer.

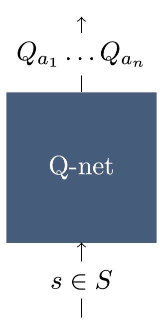


Figure 3: A generic Deep Q-Network

3. Experiments

3.1. Synthesis

Using a python2 based game environment, we implement agents that learn by interacting with the environment. The parameters for the game are shown in Table 1. The average scores obtained by different agents, across different game sizes have been shown in Table 2.

Situation	Reward
Win (finish food)	+500
Lose (eaten by ghost)	-500
Eat food	+10
Consume ghost	+200
Idle/no food	-1

Table 1: Game parameters for Q-Learning

The features chosen for training the Approximate Q-Learning Network are: bias, number_of_ghosts_1_step_away, distance_to_closest_food

Approach	Grid size	Avg Score	#Epochs	Time (train)
AQL	mediumClassic	1202.7	50	5m 45s
AQL	mediumGrid	526.4	50	32s
QL	smallGrid	499.5	2000	1m 24s

Table 2: Some simulations

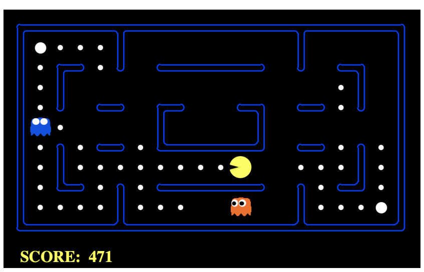


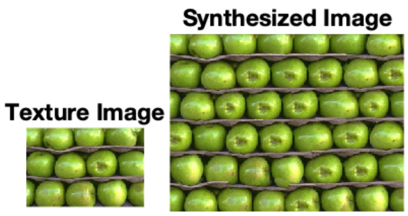
Figure 4: Pac-Man during Training

3.2. Transfer

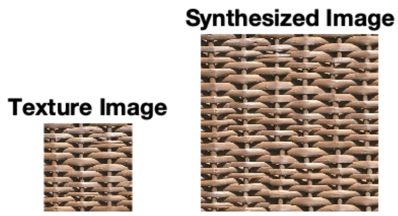
We use OpenAI's MsPacman-v0 environment to train a DQN based agent. Here, we use a neural network to approximately calculate the Q values corresponding to a given state which will be given as an input image.
The first three layers of the architecture are convolutional layers. The first layer has 32 filters of size 8*8 with stride 4. The second layer has 64 filters of size 4*4 with stride 2. The third layer has 64 filters of size 3*3 with stride 1. The fourth layer is fully connected 512 hidden nodes. The last layer is fully connected too with 9 output nodes corresponding to each of the actions possible.
We have used concepts like Replay memory and Epsilon greedy to help our model train better.

4. Results & Conclusion

In this project, we tested a wide range of common reinforcement learning techniques. The Deep Q-Learning based model, the network learns important features from game-state images and thus, approximates Q values by experience. Given enough training time and model complexity, the network can learn and improve upon the game. However, computational cost for the Deep Neural Network is a major concern. As compared to Approximate Q-Learning, Q-Learning takes a much larger number of epochs to gain similar performance. Thus, Q-Learning is not scale-able for large grids. We find that approximate Q-Learning outperforms the Q-Learning based agent and also trains faster than both the above approaches. Thus, it may be the ideal approach for medium load reinforcement learning tasks like learning how to play Pac-Man on a medium sized grid.

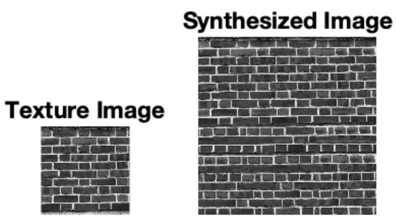


(a) Apples

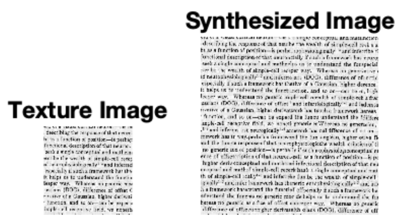


(b) Jute

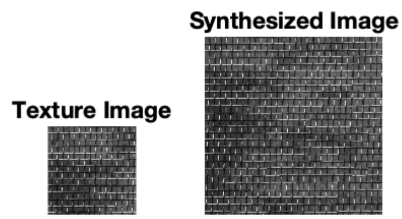
(c) Cans



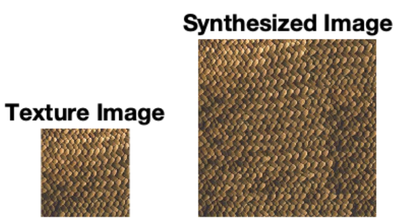
(d) D1



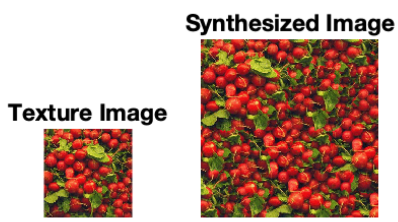
(e) text



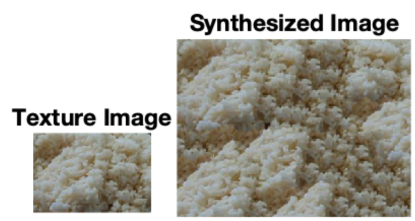
(f) Bricks B/W



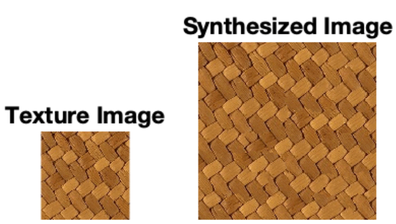
(g) Fabric



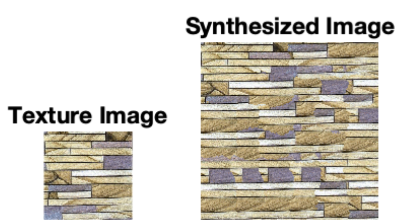
(h) Radishes



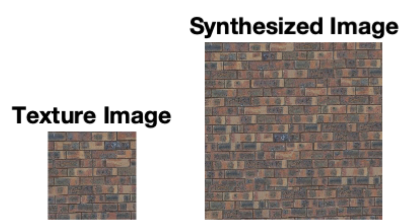
(i) Rice



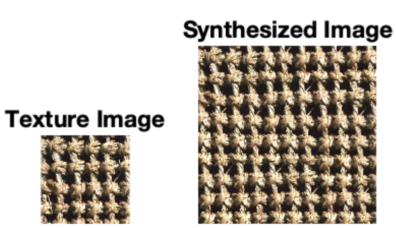
(j) Wood



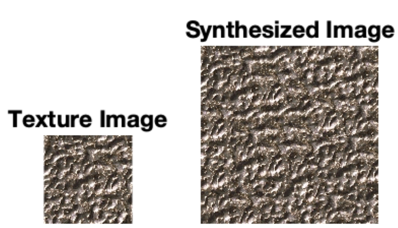
(k) Tiles



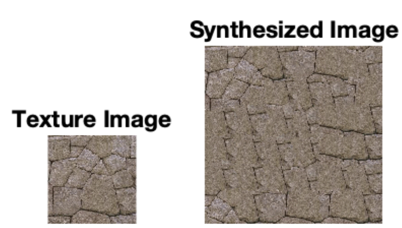
(l) Bricks



(m) Rope



(n) Pattern



(o) Stone

Figure 5: Quilting Results