A Combination of Deep Learning and Mathematical Models for Addressing the failure of Computer vision in Reinforcement learning.

Abhishek K. Singh : u6411540
Course Code : COMP 8755
Co Supervisor and Supervisor: Ms. Josephine Plested, Dr. Tom Geoden
Index

• Overview
• Roadmap
• Problem Statement
• How an Agent Makes a Decision
• Intermediate Results
• Result
• Continuing Work
Overview

We seek to solve the problem in Reinforcement Learning with respect to failure of agent to transfer its learning from source domain to related domain.

We will first talk about the problem statement and go from there to roadmap till all the way end to the algorithm we suggest.
Roadmap

• Define the Problem.
• Explain the state-of-art.
• Explain the relevant terminologies and concepts to build a background for final algorithm.
• Explain the final algorithm
• Explain its drawbacks
Problem Statement

“An Artificial agent trained on one domain, fails completely on a related domain with identical problem dynamics to the original domain. The two domains, differing only in the unimportant dynamics. (as we shall see later)”
Problem Statement

Breakout Game
Aim of the Game

- Paddle is the AI agent.
- It only has 2 actions it can take: right or left.
- If paddle misses the ball hit, the game is lost.
- The aim is to break all the bricks.
What is the problem then?

“Any visual, unimportant change and the agent fails.”

Breakout Game
Problem Statement

Perturbed state of the games. The lines don’t change the game dynamics and do not interact with the ball or paddle.[6]
Agent trained here.
Failure of the Agent

All the transfer learning techniques perform worse then the full retraining. (For majority of epochs)[6]
Transfer Learning

• A technique used to improve a learner from one domain by transferring information from a related domain\(^1\).

• Simply put, transfer learning involves learning on a related/unrelated dataset and then transferring “that learning” to make predictions about the new target dataset.
Why does the Agent Fail?

Before seeking answers as to how to train an agent that is able to transfer its knowledge successfully across domains. We ask!

“Why does the agent fail at the first place?“
How an Agent Learns to Act.

Before digging into how the answer should look like, we investigate why an Agent fails trained by Reinforcement Learning Techniques.

• All the Reinforcement Learning algorithms are based on training an Artificial agent by making use of image(screen) pixels as input.\[7\]

• The image pixels(screen/frame) representation are fed as inputs to a CNN(internals of RL algorithm) and hence everything on the screen is important.
Break Out game

Breakout Game
How an Agent Learns to Act Contd.

Before digging into how the answer should look like, we investigate why an Agent fails trained by Reinforcement Learning Techniques.

• All the Reinforcement Learning algorithms are based on training an Artificial agent by making use of image(screen) pixels as input.\[^7\]

• The image pixels(screen/frame) representation are fed as inputs to a CNN(internals of RL algorithm) and hence everything on the screen is important.
State of the Art.

The problem has been investigated in the paper Transfer Learning for Related Reinforcement Learning Tasks via Image-to-Image Translation.\[6\]

- Solution includes Generative Adversarial Network.
- How does it work -> Image to Image translation.
Our Approach

Use Saliency maps instead of Image-to-Image Translation.

Saliency(Oxford): “The quality of being particularly noticeable or important; prominence.”
Intermediate results from the Breakout using A3C.\textsuperscript{[10]}
Saliency in context of RL

- What features are important to an RL agent while making a decision?
Saliency map for the Game Breakout

Saliency map for the breakout game.\cite{8}
How does it work

The saliency scores are calculated by masking every pixel, leading to a new perturbed state, on the screen and then calculating the respective value of the Policy function of the state. The perturbed state with the largest difference in Policy function is the one with highest saliency. In equations, it looks like:

\[
S_\pi(t, i, j) = \frac{1}{2} \| \pi_u(I_{1:t}) - \pi_u(\hat{I}_{1:t}') \|
\]

\[
\hat{I}_{1:k} = \begin{cases} 
\phi(I_{k,i,j}) & x \leq 0 \\
I_k & x > 0 
\end{cases}
\]
Proposition

Use Long Short-Term Memory with cell state being fed with the saliency map.

*LSTM has memory gates, to help retain and forget information.*
LSTM Architecture

A basic LSTM Network.[9]
Output of LSTM

Output:

\[
\begin{align*}
    f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
    i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
    o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
    \tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\
    c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
    h_t &= o_t \circ \sigma_h(c_t)
\end{align*}
\]
LSTM with Policy Gradient

Modify the network to look like .
Output of Policy Network

\[ \sigma \left( \begin{pmatrix} w_{h1s1} & w_{h1s2} & 0 & 0 \\ 0 & w_{h2s2} & w_{h1s3} & 0 \\ 0 & 0 & w_{h3s3} & w_{h3s4} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} \right) = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \]

The second set of equations are:

\[ \sigma \left( \begin{pmatrix} w_{m1h1} & w_{m1h2} & 0 & 0 \\ 0 & w_{m2h2} & w_{m2h3} & 0 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \right) = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} \]
Substitute the output in LSTM cell to come up with:
Problems and Future Work:

- Improve the code using GPU.
- Very High training times.
- Dimension issues.
- Revisit the custom saliency score function.
References


