Visualisation of Deep Reinforcement Learning in Artificial Intelligence

Jacob Deasy

University of Cambridge

jd645@cam.ac.uk

Overview

What is Reinforcement Learning?

- Area of machine learning concerned with how *agents* decide which actions to take in an *environment* to maximise cumulative *reward*.
- Actions and environment are formulated as a Markov decision process (MDP).
- No use of correct input/output pairs like supervised learning.
- Trade-off must be struck between *exploring* the environment and possible decisions vs. *exploiting* accumulated knowledge.

Motivating Visualisation of RL

- Understanding how the model work, insights into scene-action associations in the brain.
- Identification of strategies and sub-strategies.
- Debugging of what is wrong in faulty agents.
- Explaining the decision making process to the general public.

Traditional Reinforcement Learning

Formulated as a MDP with set of states S and actions A:

- $P_a(s, s')$ prob. of transitioning state s to s' under action a.
- $R_a(s, s')$ immediate reward after transition from s to s' by a.

Agent's action selection modeled by a map known as a *policy*:

$$\pi : S \times A \to [0, 1], \pi(a|s) = P(a_t = a|s_t = s)$$

For MDPs, the *Bellman equation* is a recursion for expected rewards by taking actions found by policy π:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^\pi(s').$$

Q-Learning

- Q-function to assess quality of state-action pair $Q : S \times A \to \mathbb{R}$.
- At each time step t, agent selects action a_t, observes reward r_t and enters a new state s_{t+1}.
- Uses a *value iteration update* to incorporate a weighted average of old and new value information:

$$Q^{new}(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a))$$

Extension to Deep Q-Learning

Use a deep convolutional neural network to approximate the optimal action-value function:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... | s_t = s, a_t = a, \pi].$$

Q-learning update at iteration i uses loss function:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[\left(r + \gamma \max_{a} Q(s', a'; \theta_i^-) - Q(s, a, ; \theta_i) \right)^2 \right]$$

Use backpropagation of error to update Q-function.

Deep Q-Network Architecture

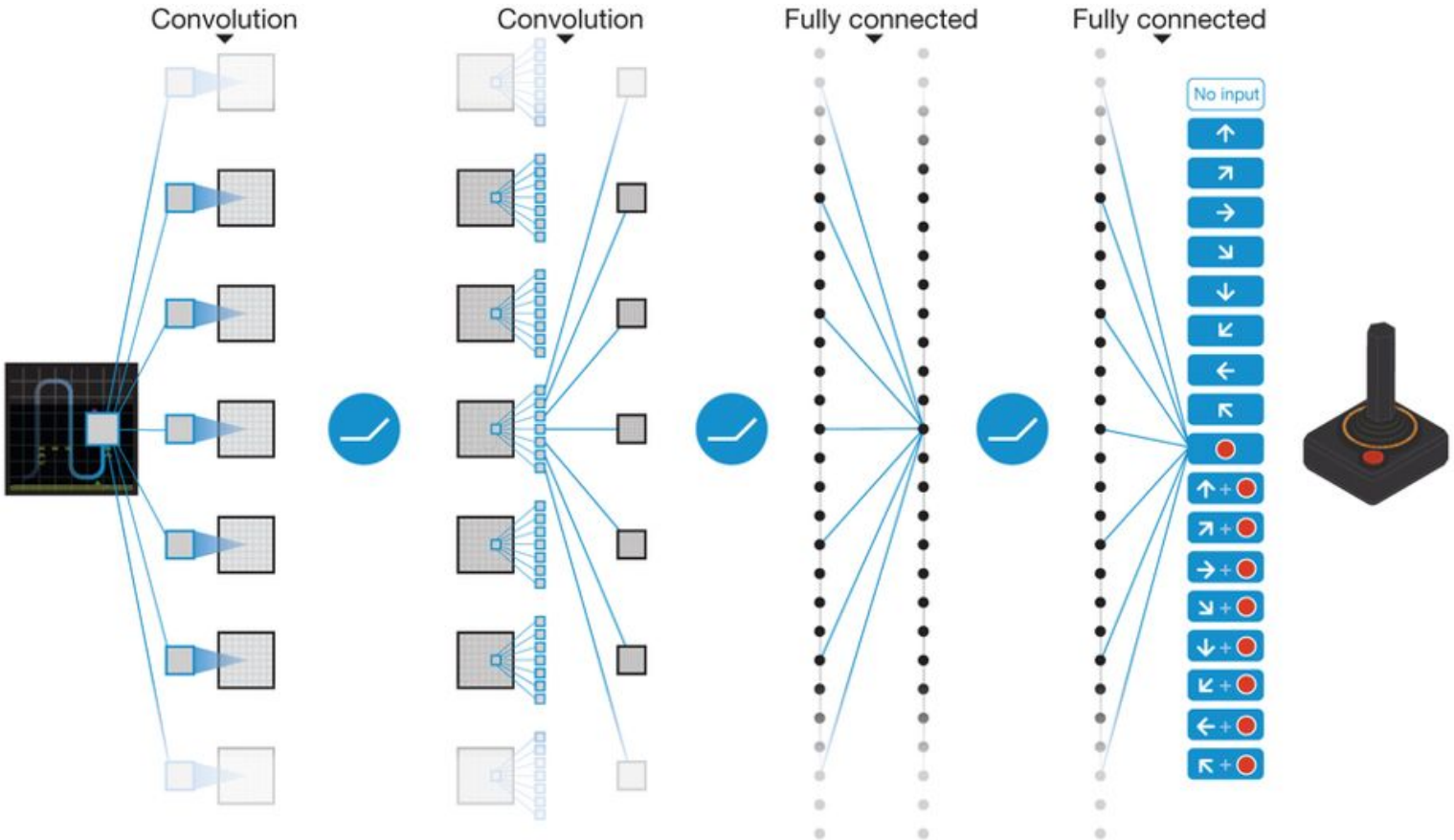


Figure: DeepMind diagram.

Convolutions

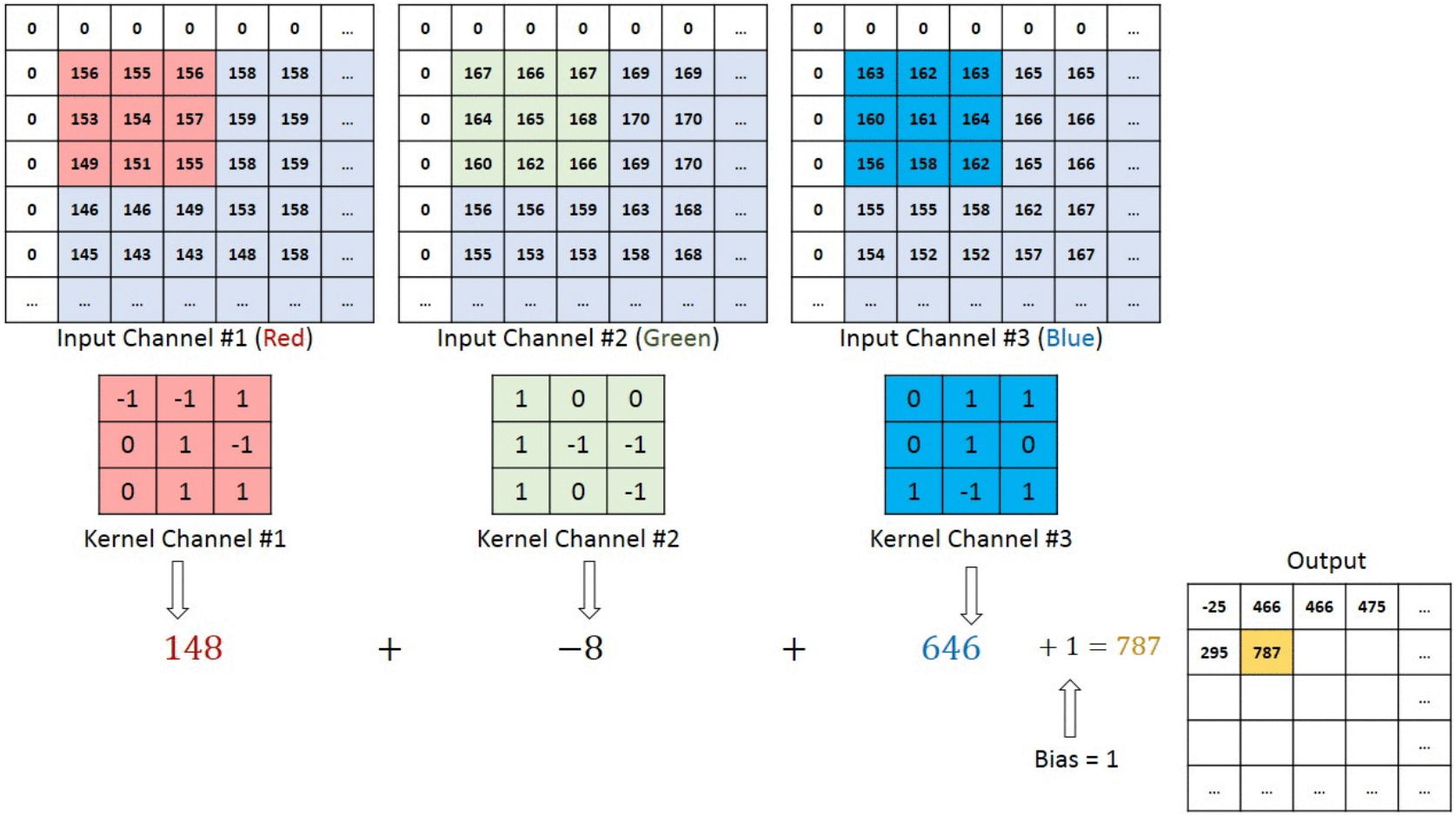


Figure: A single, 3-channel convolutional kernel in action.

Convolutional Architecture

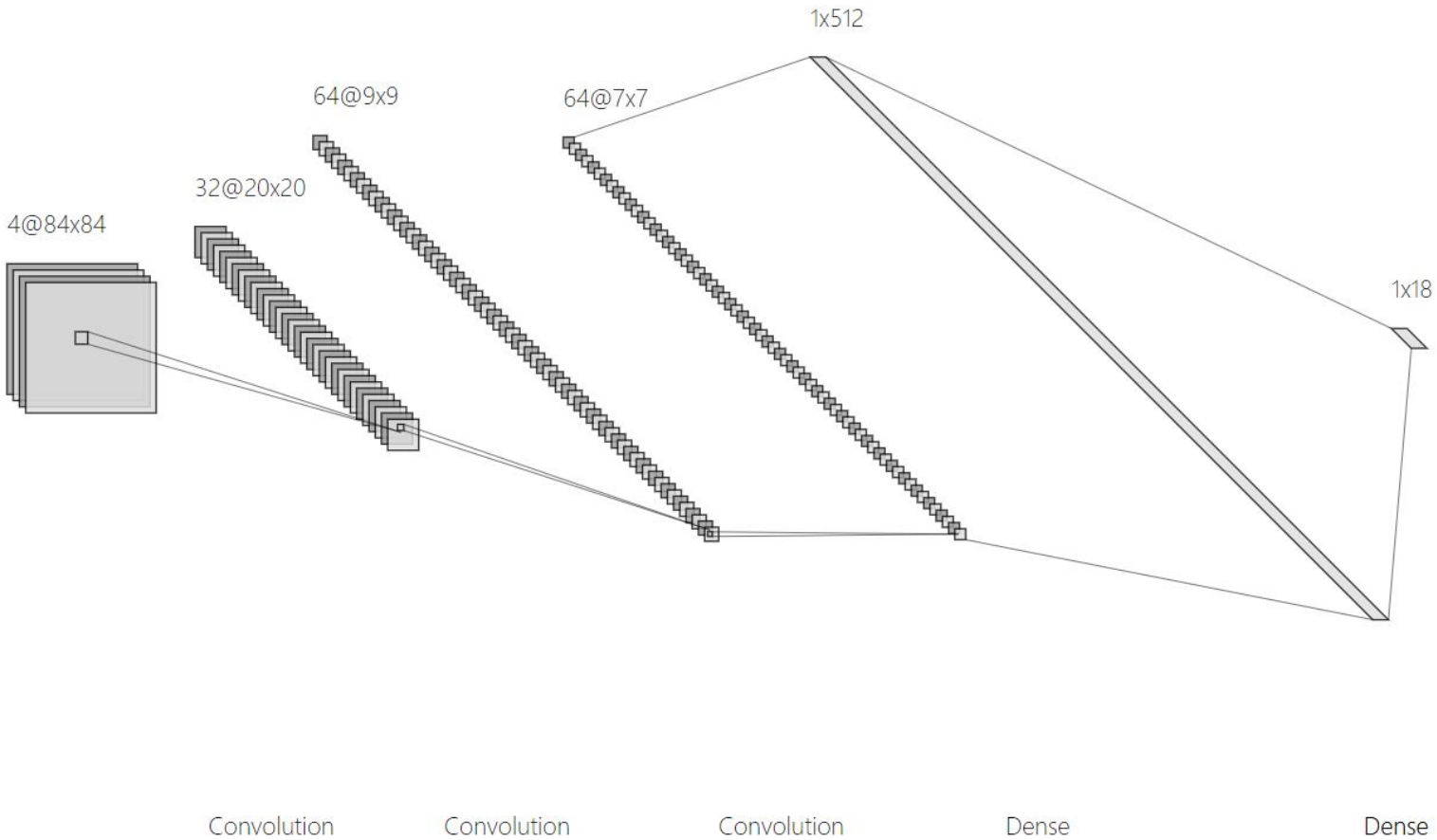
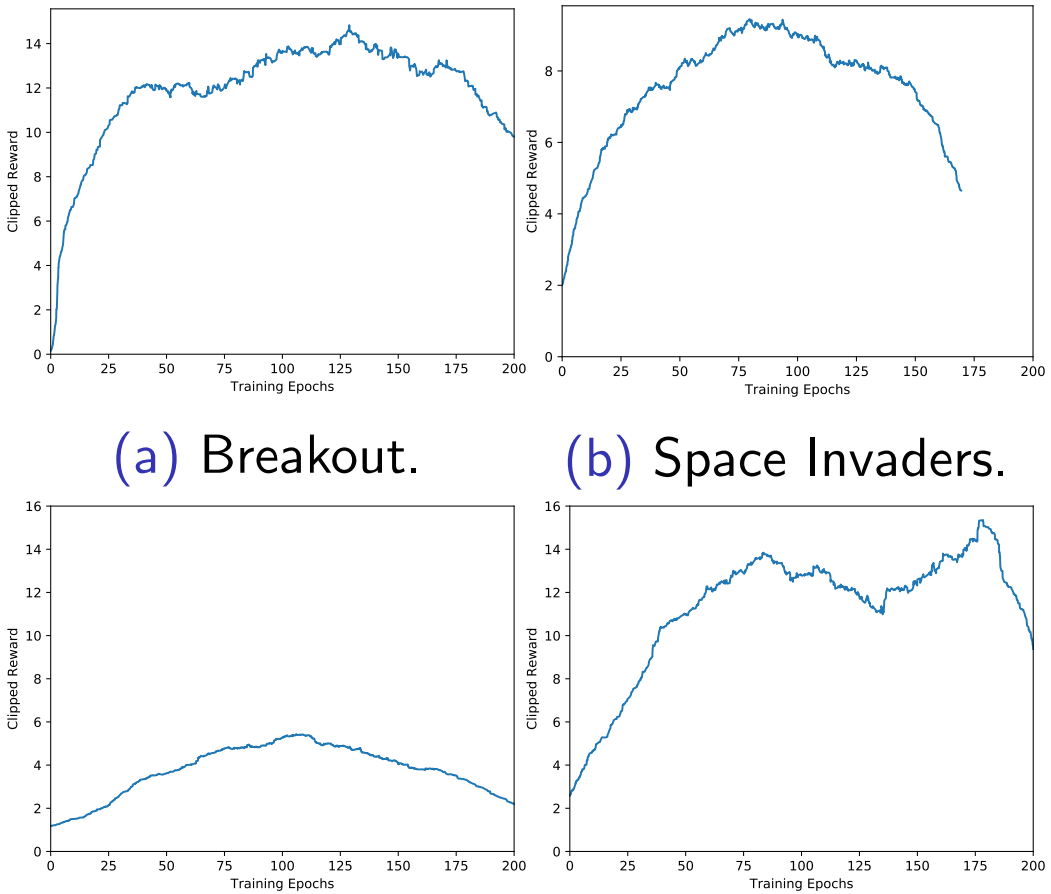


Figure: A more detailed look at the architecture.

Results of Training



(a) Breakout.

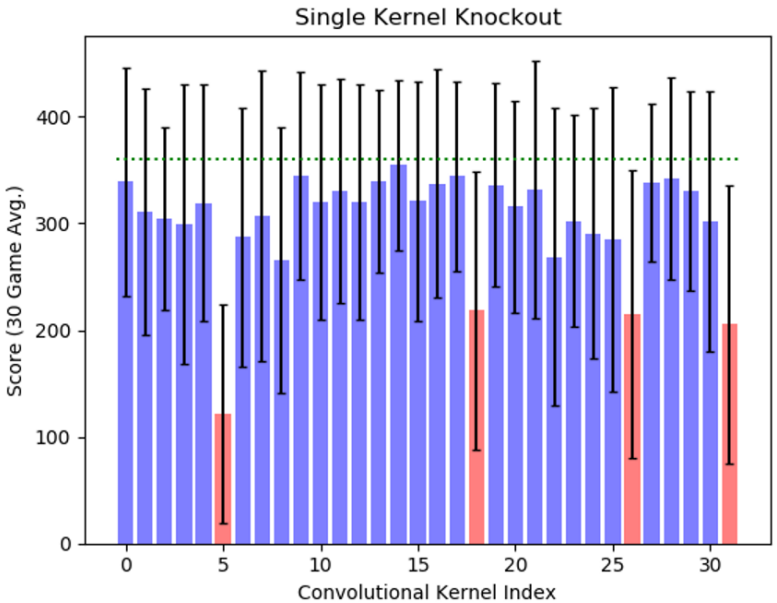
(b) Space Invaders.

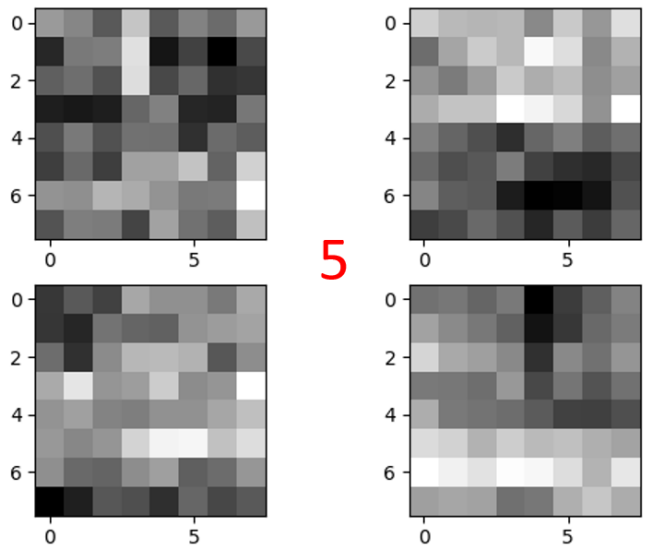
(c) Stargunner.

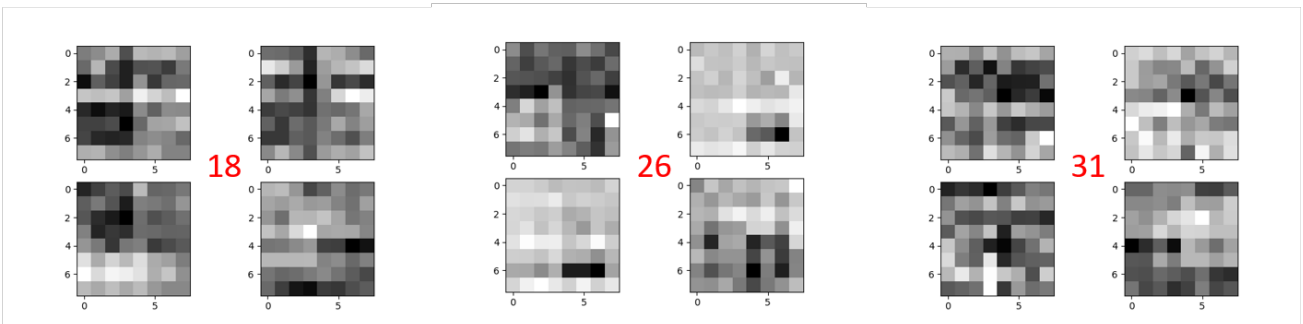
(d) Stargunner v2.

Figure: Training curves tracking the agents average score. One training epoch corresponds to 1,000,000 Atari frames.

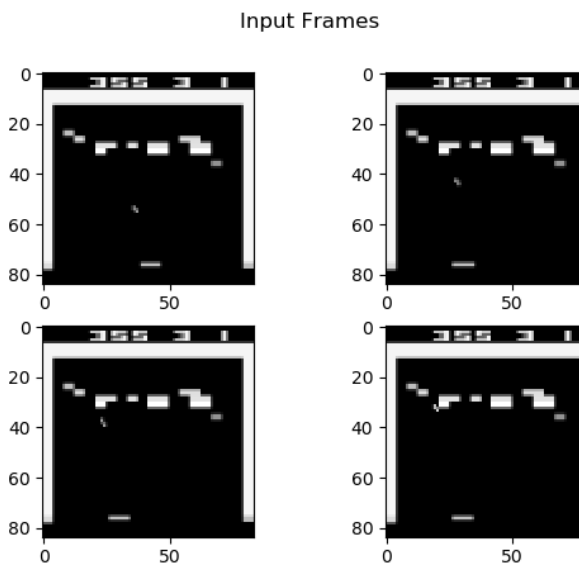
Basic Approach: Knocking Out kernels

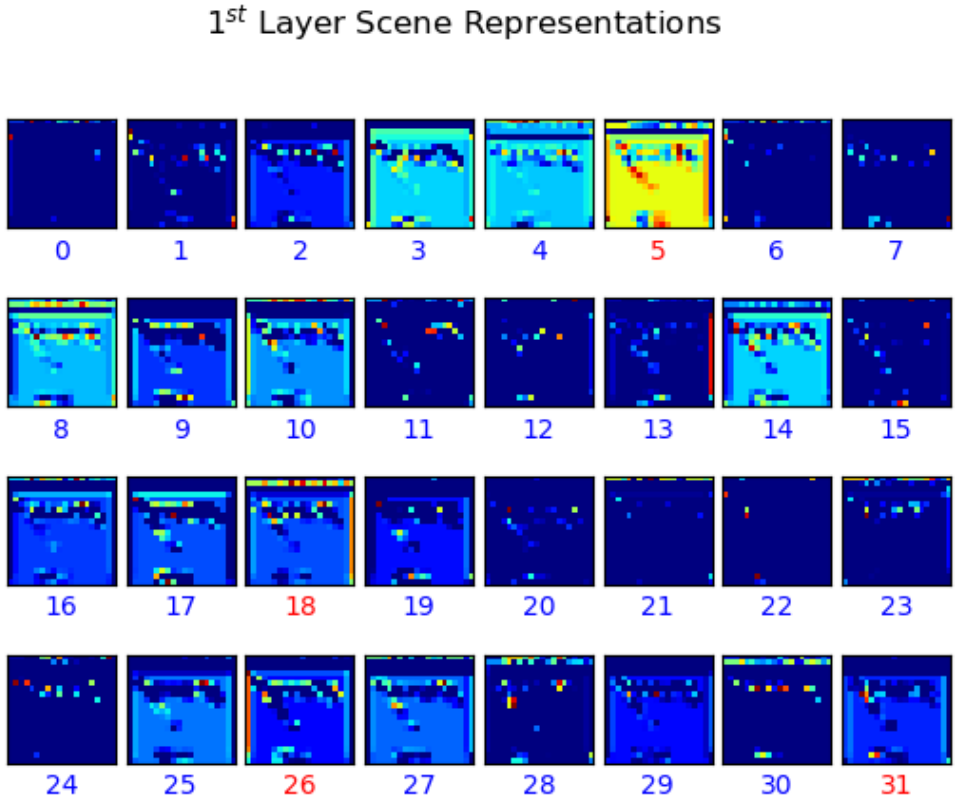




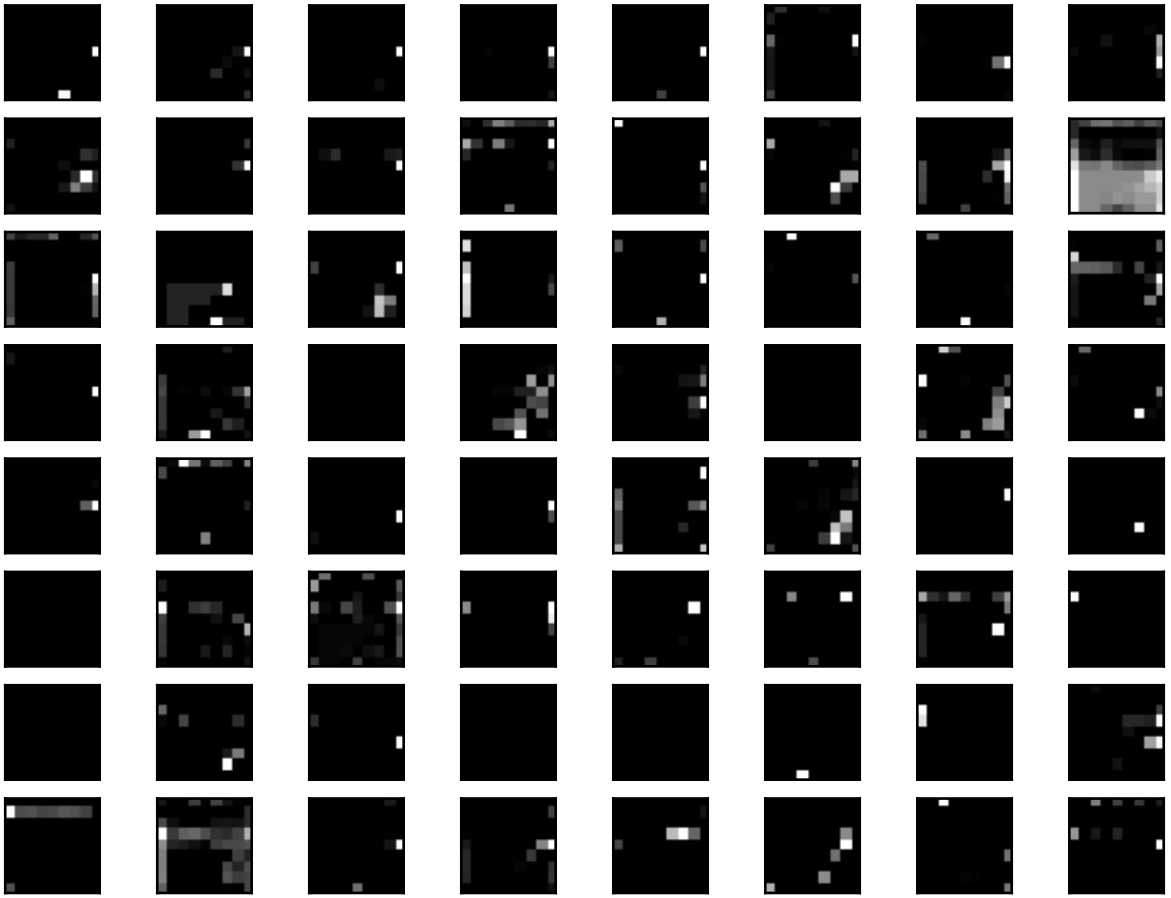


Internal Scene Representation





Deeper Internal Scene Representation



Feature Visualisation

- Use tools from image classification on visual RL networks.
- Is visualising an *action* different to visualising an *image class*?
- *Activation maximisation* uses gradient *ascent* to adjust input according to gradient of deep layers.

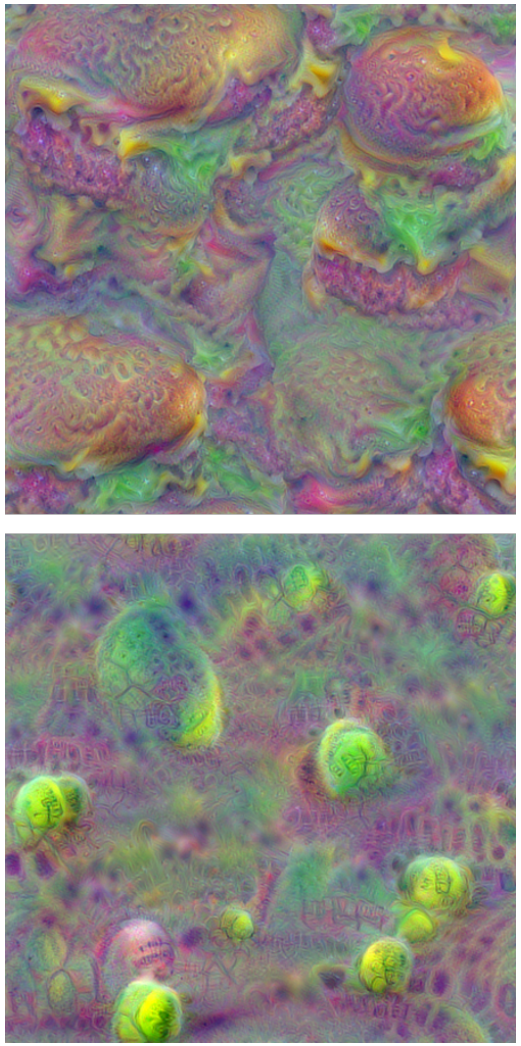


Figure: Cheese burger and tennis ball features from DeepDream.

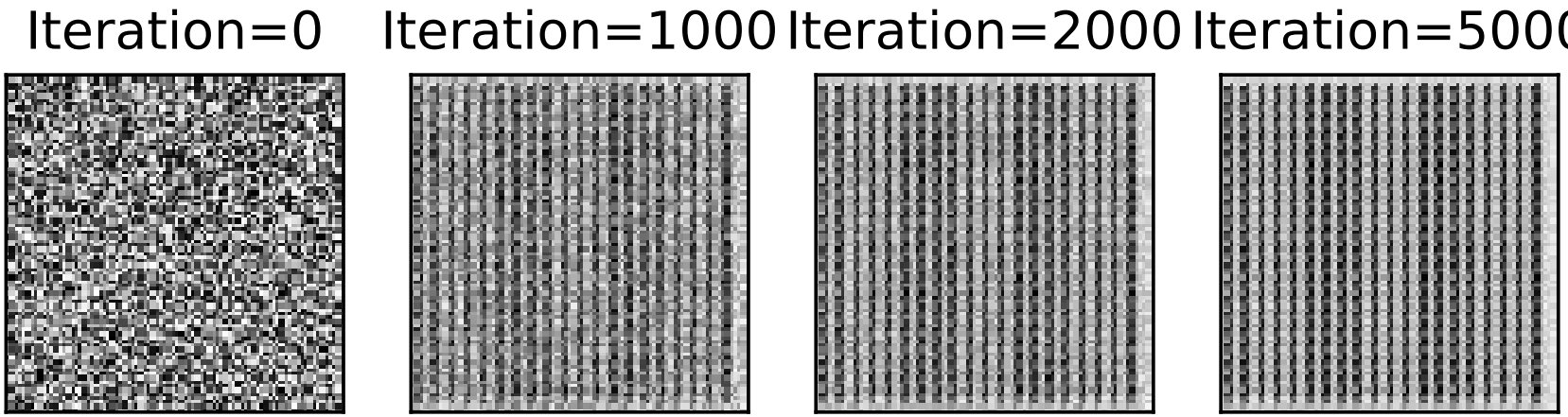


Figure: Progression of activation maximisation.

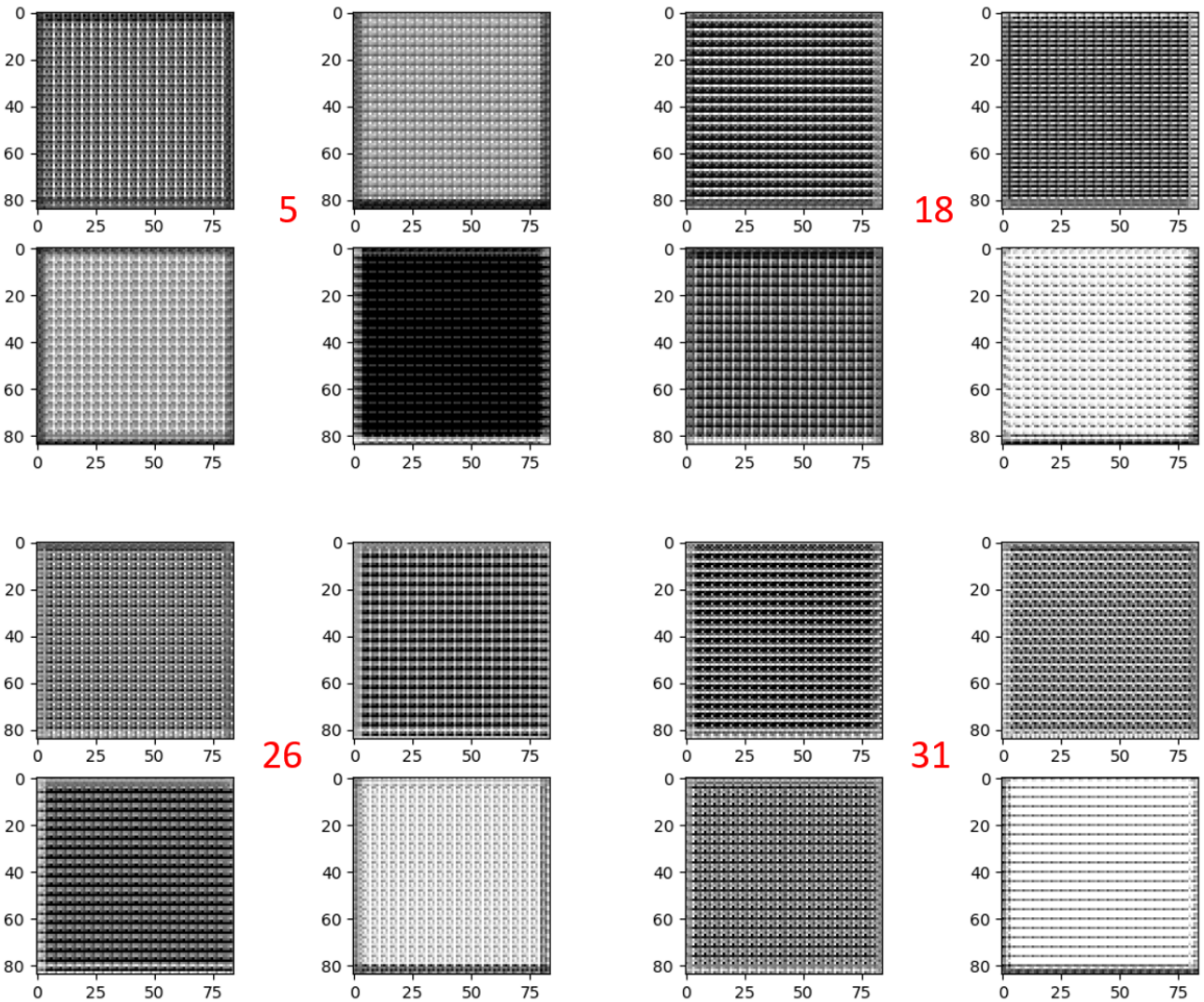


Figure: 1st Layer Features

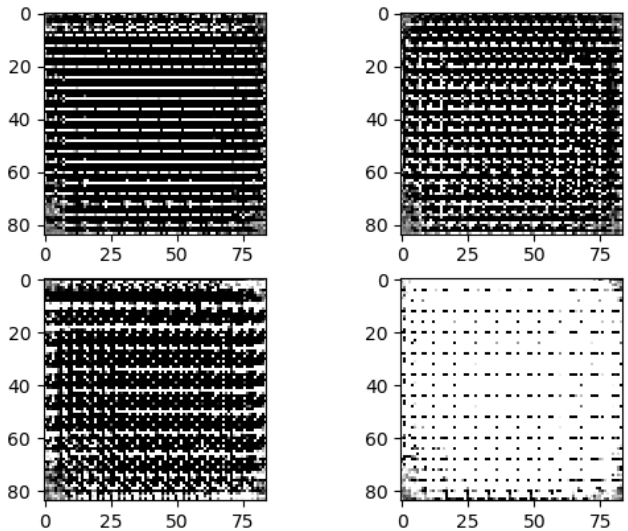


Figure: 2nd Layer Feature

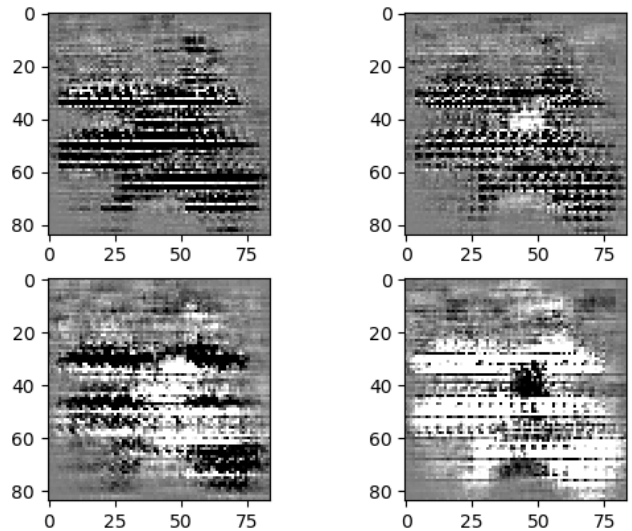
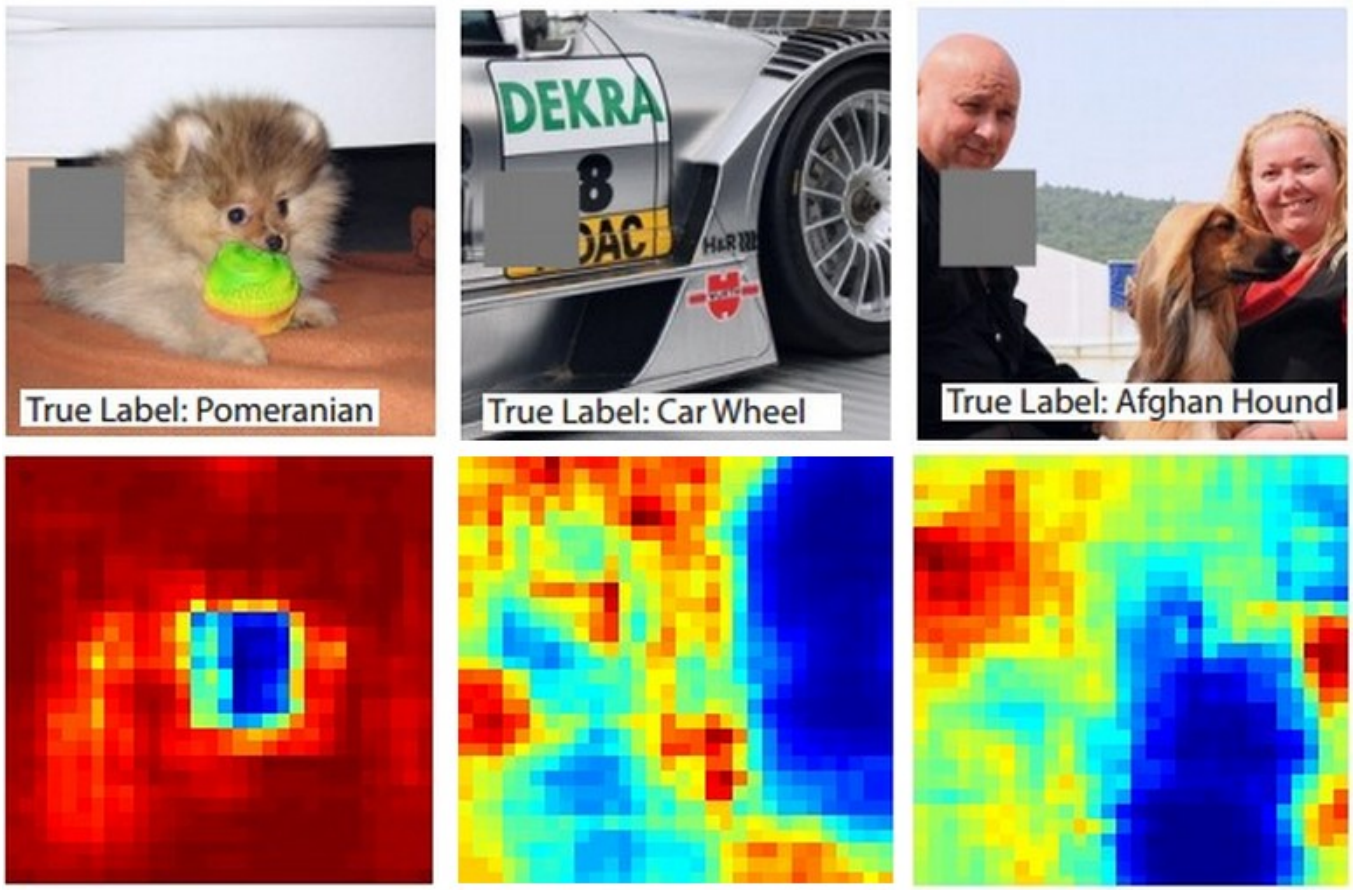


Figure: 'RIGHT' Action Feature

Saliency Maps

- Occlude image sections to discern which features matter.
- Can use perturbation or gradient based methods.



Saliency via Local Blurring

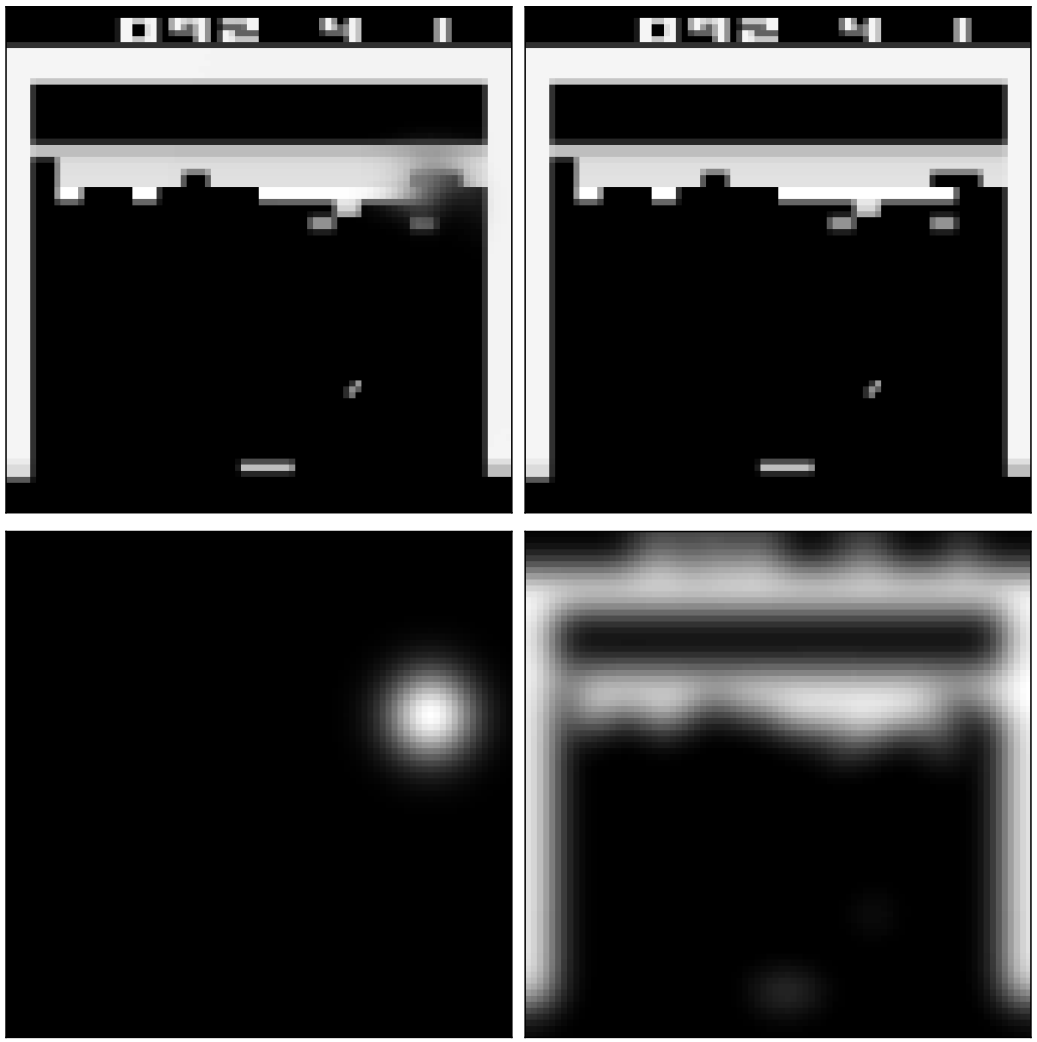


Figure: $\Phi(I_t, i, j) = I_t \odot (1 - M(i,j)) + A(I_t, \sigma_A) \odot M(i,j)$

Breakout Strong Policy

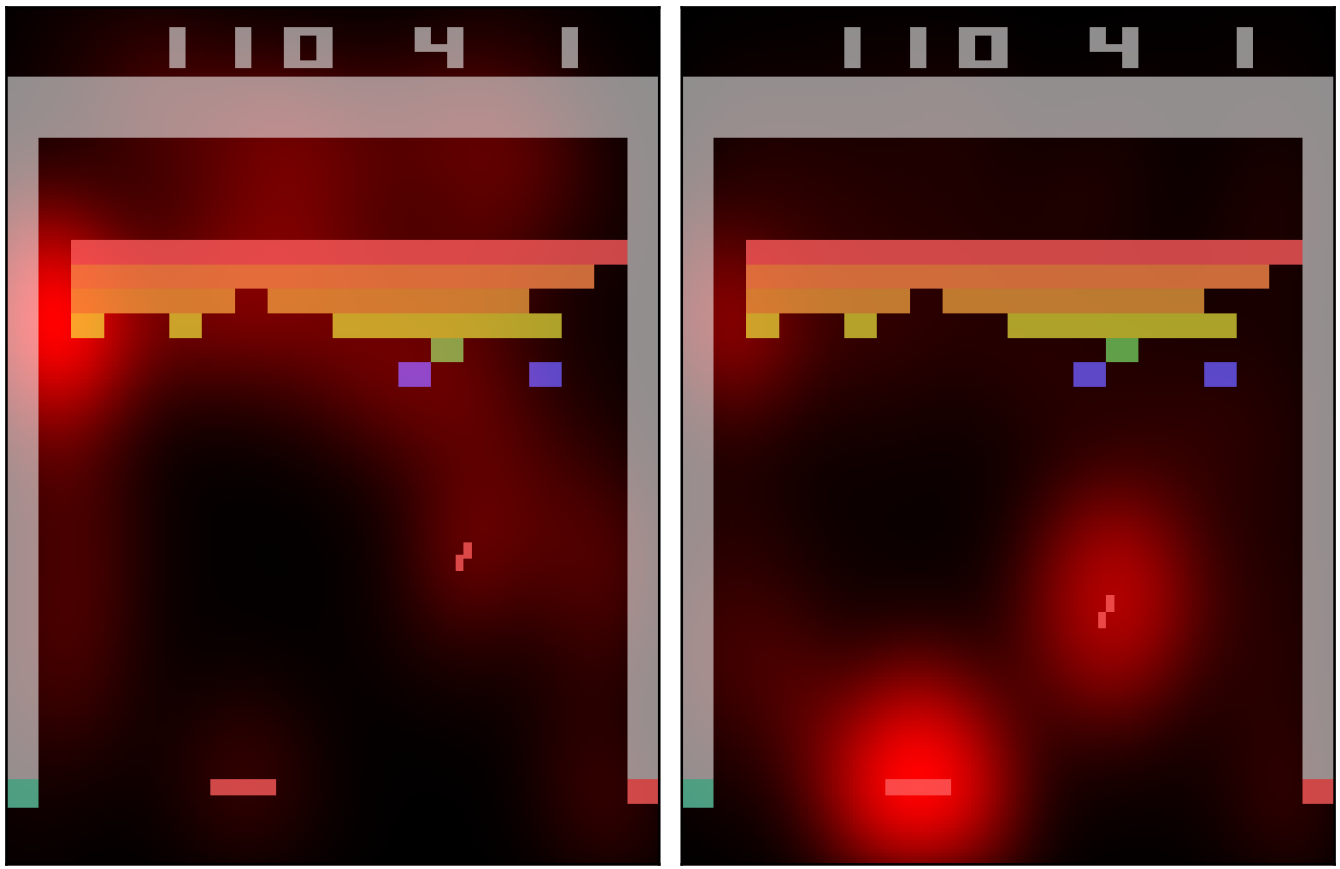


Figure: Agent switches focus between observations.

Space Invaders Strong Policy

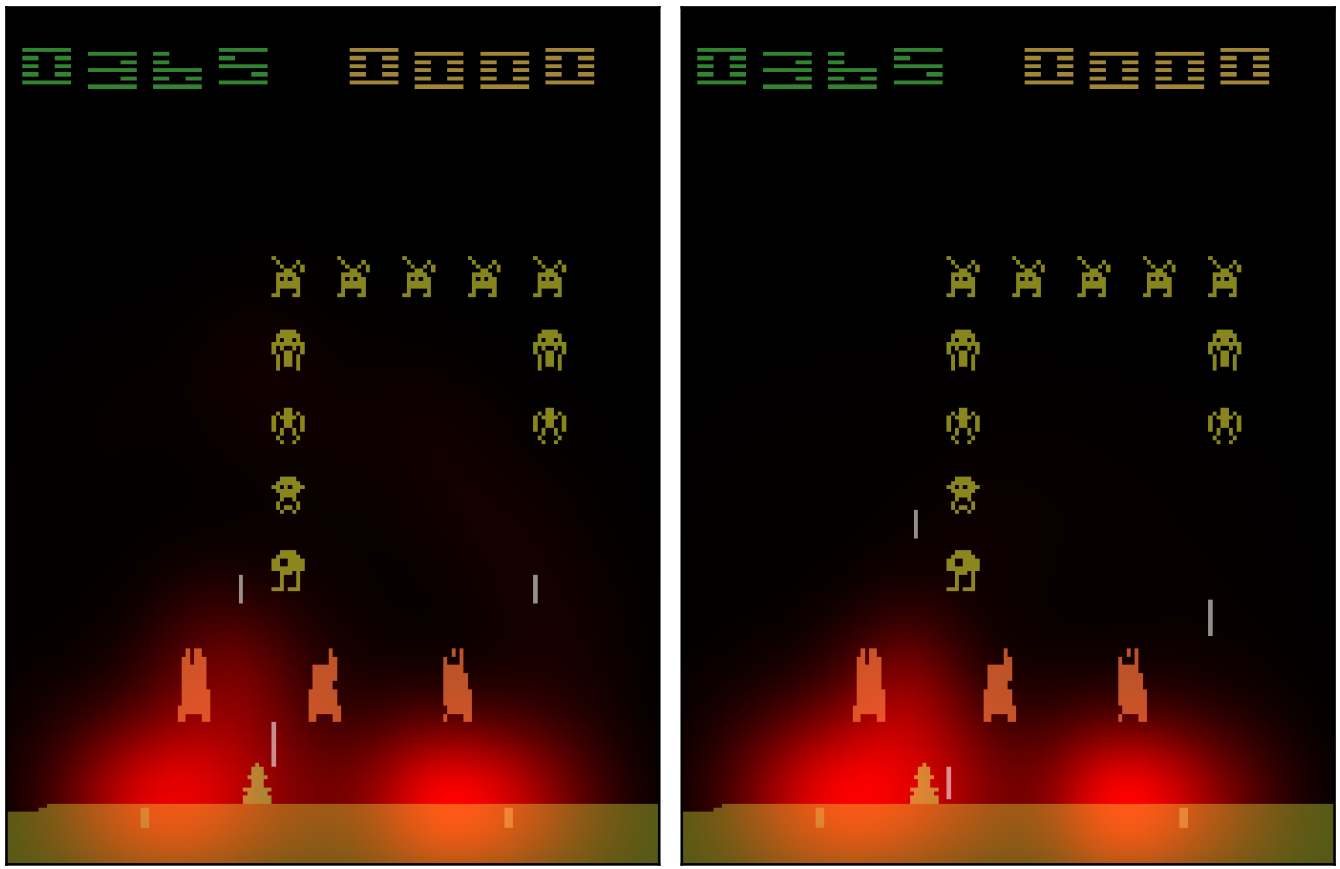


Figure: Focusing on hiding and dodging.

Learning Policies

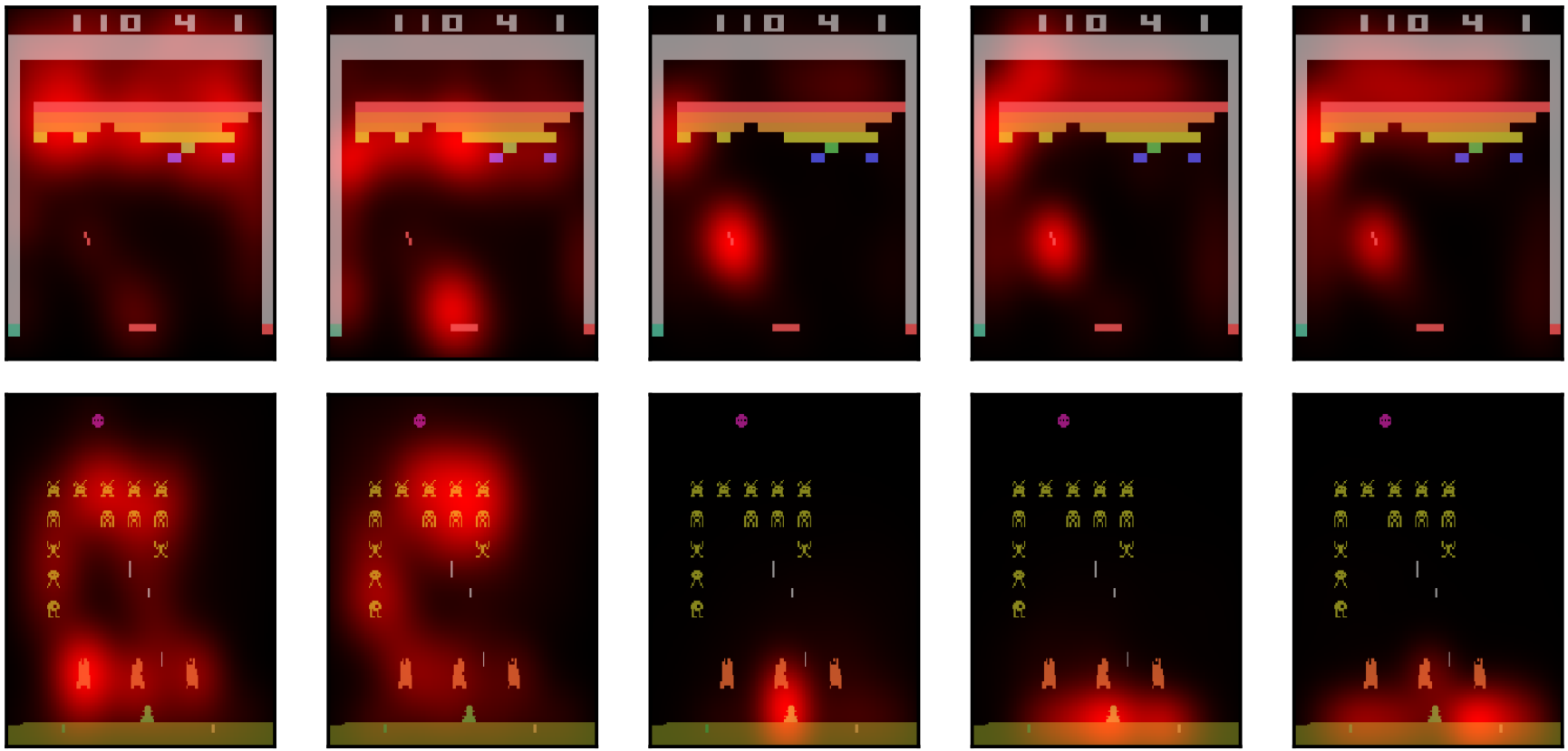


Figure: Saliency maps at 5 different stages of training. From left to right: an untrained network, a network after 3125 parameter updates, and networks after one third, two thirds and complete training.

Learning Policies

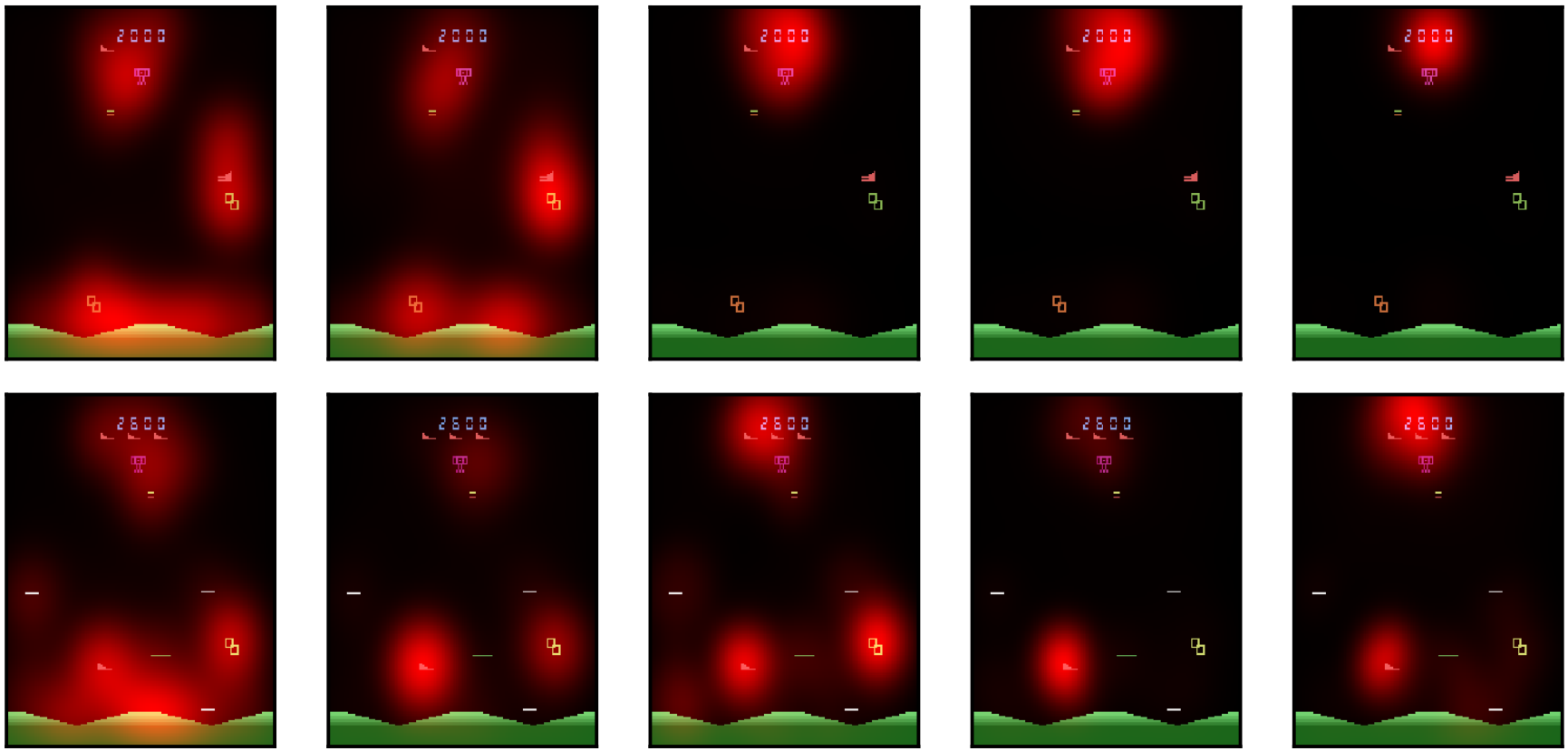


Figure: Improving agent focus and playstyle.

Semi Aggregated Markov Decision Process

- Semi MDPs divide the agents actions into *options* (skills).
- Each skill has an independent policy π^σ, set of states initiating the skill I^σ, and set of termination probabilities β^σ.
- Aggregated MDPs segment the state space using spatial abstractions.

Aggregation

- Use penultimate layer as compressed representation of the input stats.
- 120,000 inputs and penultimate layer activations were recorded.
- $120,000 \times 512$ matrix reduced to $120,000 \times 50$ by principal component analysis (PCA):
  - 99% of variance in Breakout states retained.
  - 95% of variance in Space Invaders states retained.
- t-distributed Stochastic Neighbour Embedding (t-SNE) used to reduce dimensionality further to 2D for visualisation.

Penultimate Layer Representation of Breakout

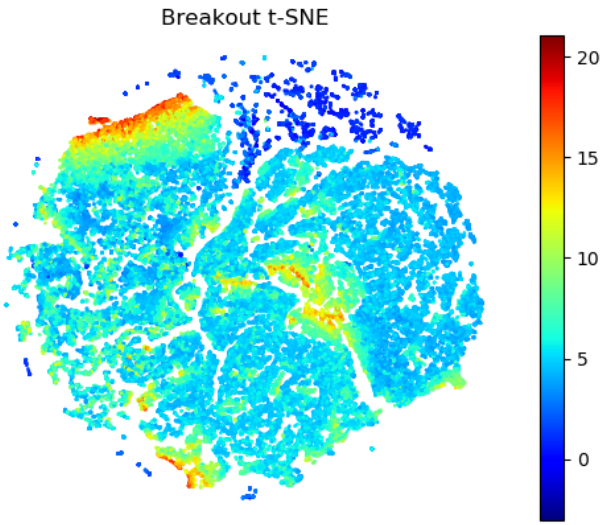


Figure: Colour by max Q value.

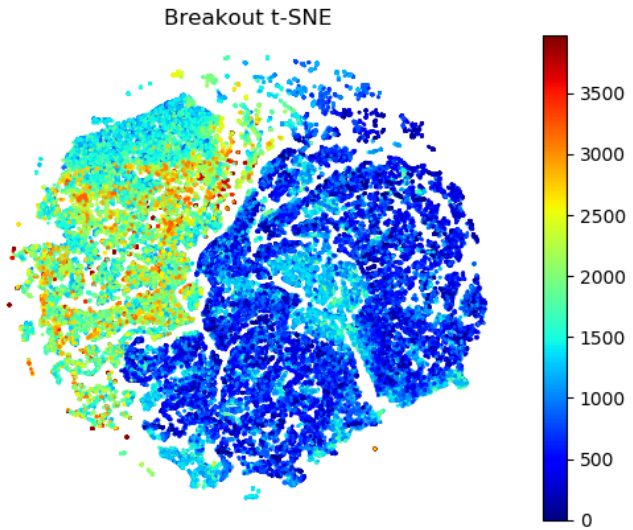
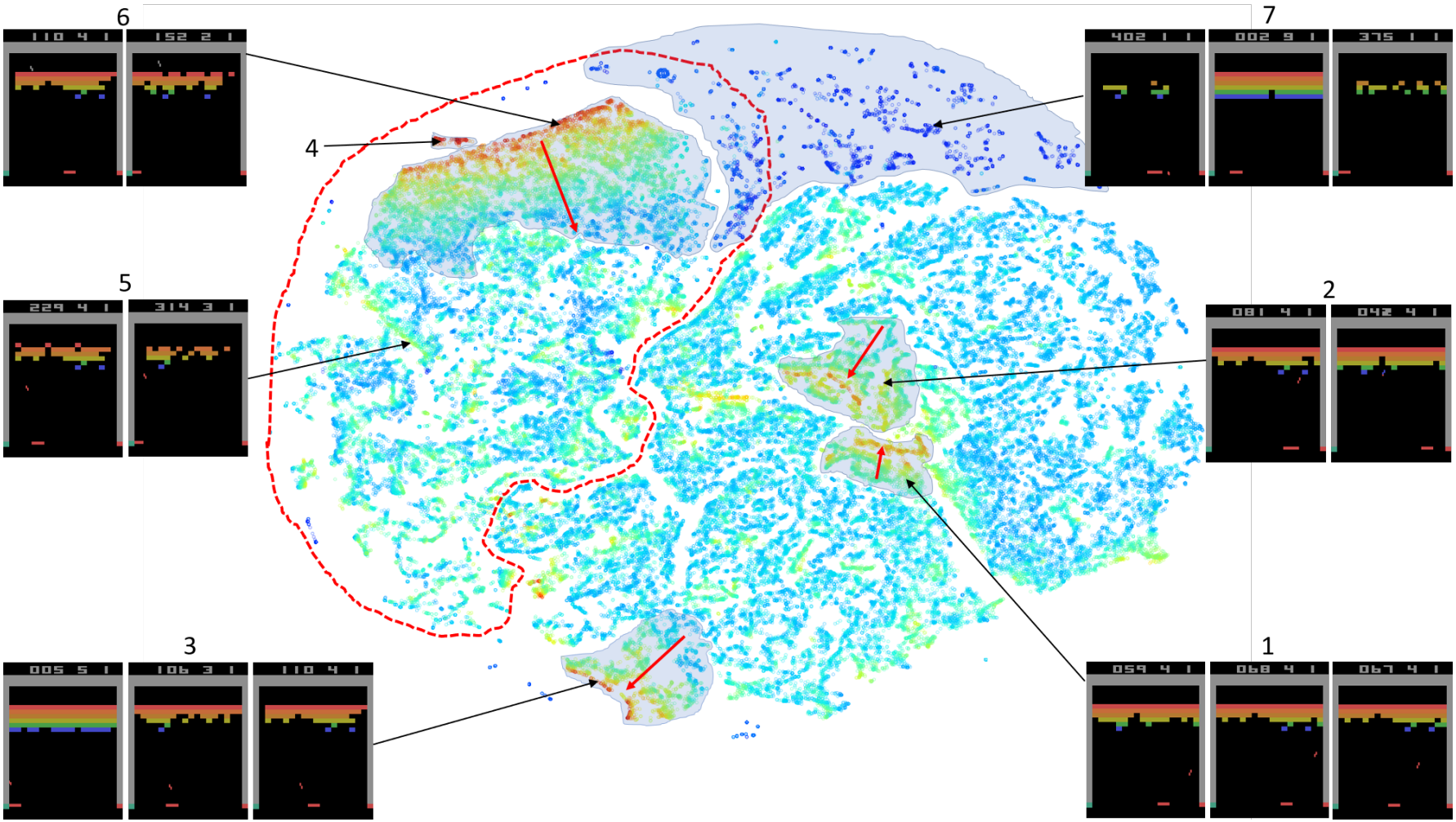


Figure: Colour by time point.

Labelled Breakout t-SNE



Penultimate Layer Representation of Space Invaders

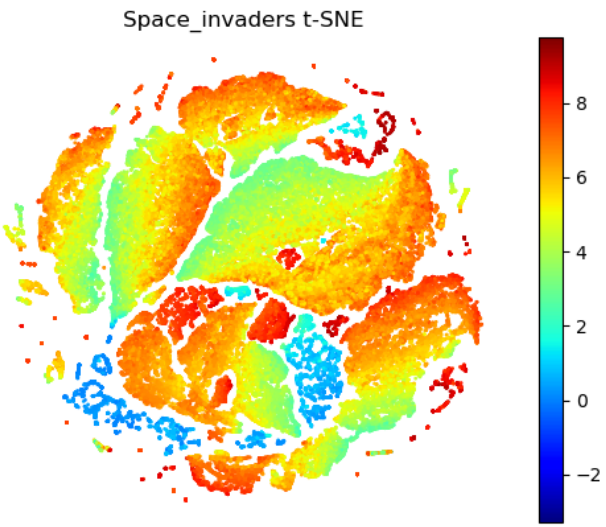


Figure: Colour by max Q value.

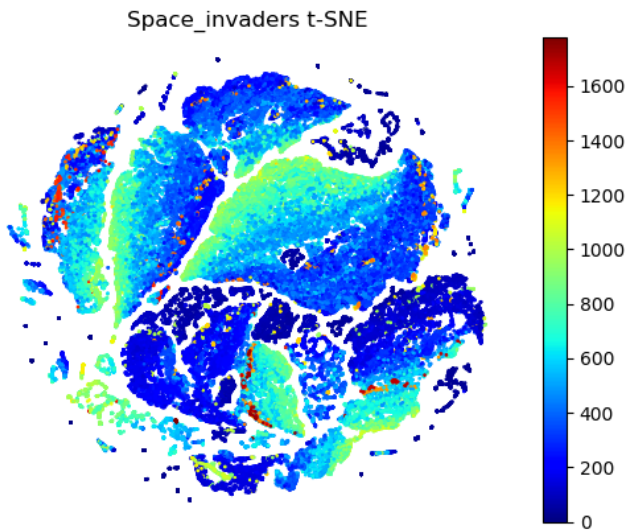
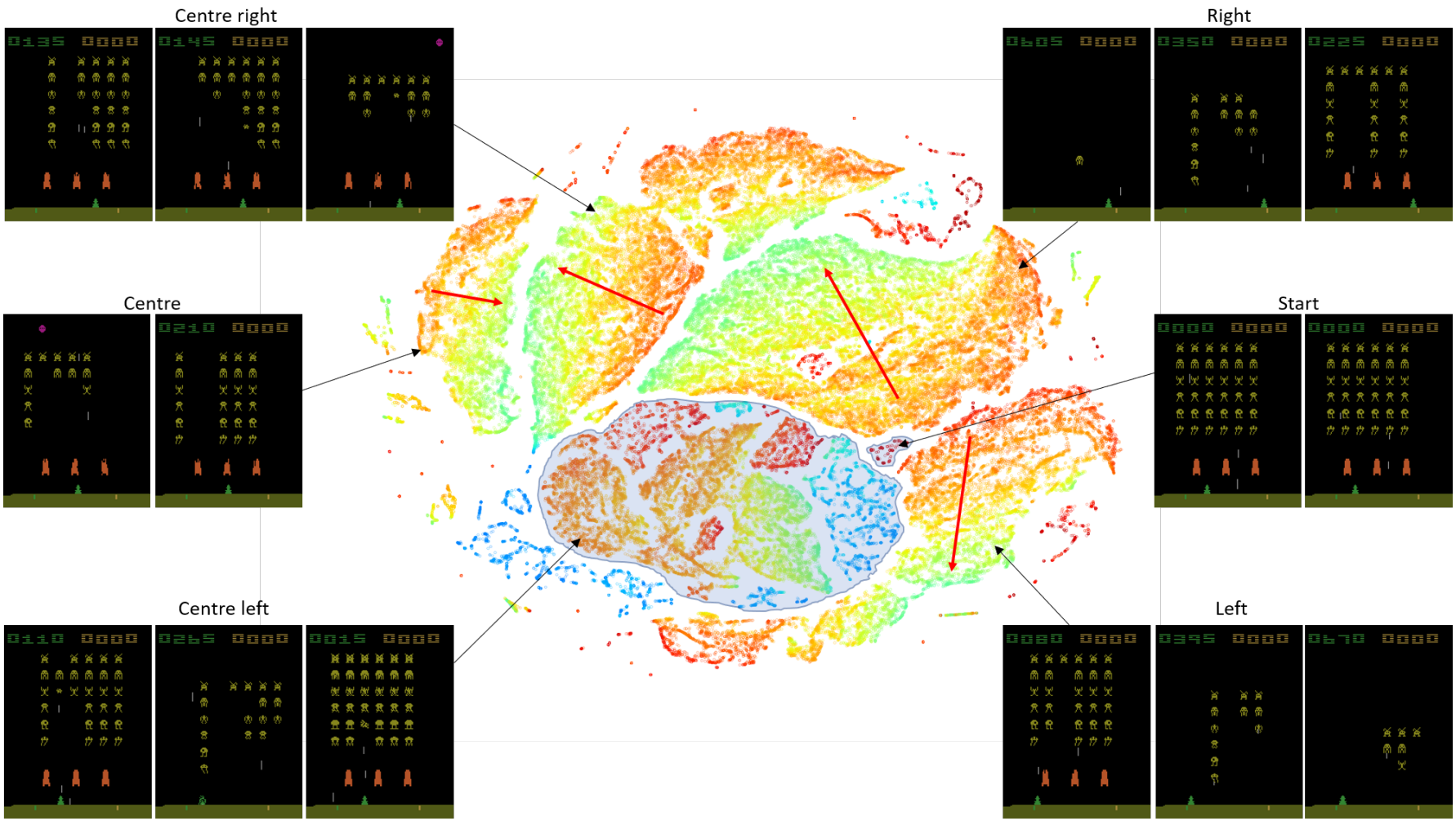


Figure: Colour by time point.

Labelled Space Invaders t-SNE



Summary and Outlook

- Visualised *where* the agent is looking using saliency maps.
- Visualised *what* features the agent is looking for using activation maximisation.
- Visualised *how* the agent associates these structures by dimensionality reduction of higher-level features.
- Future work:
  - Automate the SAMDP analysis process.
  - Use recurrent neural networks (better for continuous tasks).
  - Use generative attention networks to build saliency into network.

Thank you for listening.