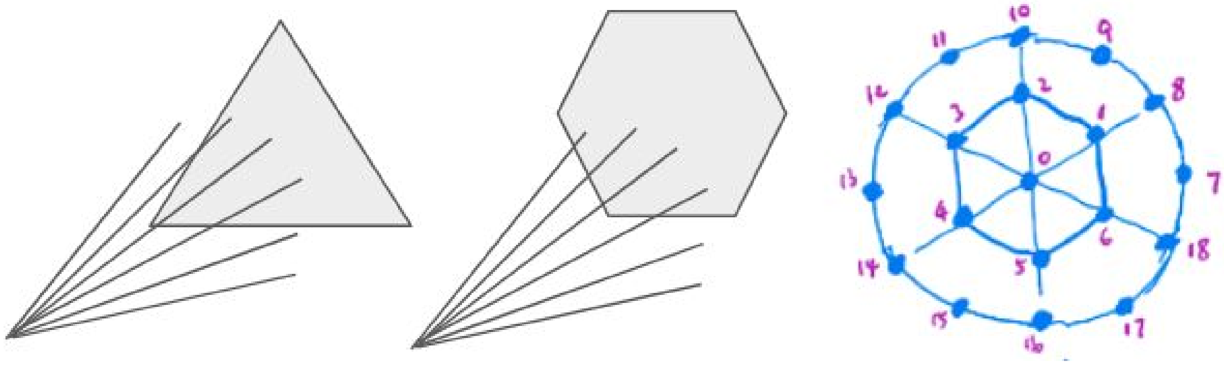
Felix Wang -- Deep Q-Network for Active Whisking

In a 2017 NIPS paper "Toward Goal-Driven Neural Network Models for the Rodent Whisker-Trigeminal System", several deep neural networks that input whisker array signals are trained for shape detection. The best top-5 accuracy is 44.8%, while chance is 0.85%. One potential improvement on this performance is active sensing. In that paper sensing data are obtained from passive sweeping the whisker array against objects, but in real life rats actively control where on the object they whisk against based on past whisking signals. Incorporating a sequential decision-making process in the data acquisition phase can obtain more distinguishing features of the shape than from random passive sweeps, and thus lead to both higher efficiency in terms of the number of whisks needed and higher classification accuracy. In other words, the goal is to learn an optimal controller for a whisker array to collect most representational observations in order to sequentially improve the estimation of the object shape with least whisks.

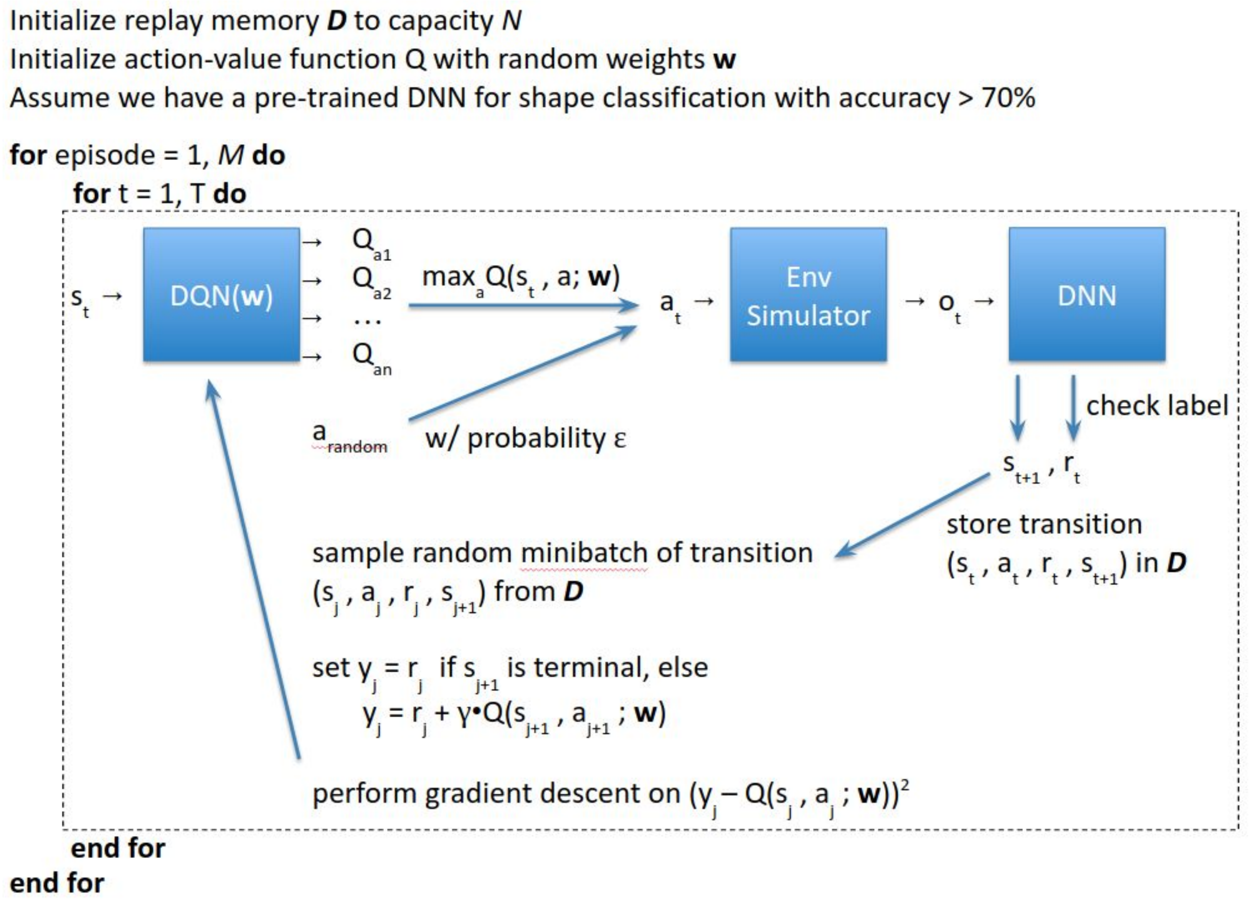
I. Model Abstraction

As a starter, I have simplified the problem to a 2d classification of equilateral triangles and equilateral hexagons. The goal thus translates into training a controller that moves the whisker array to a position that can collect most representational features of the triangles and hexagons, such as corners. I've also simplified the whisker array to a radially outward positioned laser array with fixed angles between lasers, so the controller only changes the head position of the laser array and the data collected are Euclidean distances between contact points and the head of laser array.



There are 19 lasers, 1 aligns with the center axis, 6 in an inner circle with an angle of 30 degree from the center axis, and another 12 in an outer circle with an angle of 45 degree from the center axis. All 19 lasers radiate out from the same laser head and each observation contains 19 distances. A sequence consists of multiple observations for the same shape, ie the same size, position and orientation. Each observation varies by the head position of the laser array. The height of laser head above the 2d shape varies between 1 and 10 discretely, thus the smallest distance in an observation is 1 from along the central axis while the longest distance is 10 * sqrt(2) from the outer circle. Distance measurement off shape uses a value of 255.

II. Algorithm Specification



```
Initialize replay memory $\boldsymbol{D}$ to capacity $N$
Initialize action-value function Q with random weights $\mathbf{w}$
Assume we have a pre-trained DNN for shape classification with accuracy > 70%

for episode = 1, $M$ do
    for t = 1, T do
        $s_t$ → DQN($\mathbf{w}$) → $Q_{a1}$, $Q_{a2}$, …, $Q_{an}$
        $\max_a Q(s_t, a; \mathbf{w})$ → $a_t$
        $a_{random}$ w/ probability ε → $a_t$
        $a_t$ → Env Simulator → $o_t$ → DNN → $s_{t+1}$, $r_t$ (check label)
        store transition $(s_t, a_t, r_t, s_{t+1})$ in $\boldsymbol{D}$
        sample random minibatch of transition $(s_j, a_j, r_j, s_{j+1})$ from $\boldsymbol{D}$
        set $y_j = r_j$ if $s_{j+1}$ is terminal, else
            $y_j = r_j + \gamma \cdot Q(s_{j+1}, a_{j+1}; \mathbf{w})$
        perform gradient descent on $(y_j - Q(s_j, a_j; \mathbf{w}))^2$
    end for
end for
```

I adopted the algorithm from DeepMind's 2015 "Playing Atari with Deep Reinforcement Learning". There are three main components, presented in blue box: First, a Deep Q-Network that does the decision making. It takes in a state vector, represented by a stack of five most recent observations and output all the possible actions with corresponding probabilities. We choose the action of the highest probability, with a small chance reverting it to a random action to avoid a lack of exploration, and ship it to the environment simulation. The simulation engine then moves the laser head according to the action and returns a new observation. The simulation also resets the shape with different characteristics for each new episode, and keep it the same for all time steps within the same episode. In the third stage, the new observation goes to a simple deep neural network (originally a RNN which did not work that well) that classifies the observation and outputs a probability together with the classification, where a value close to 0 means high probability for triangle and 1 for hexagon. A value corresponding to 90% confidence in the classification is used as a threshold to mark the current step as the terminal step of the episode.

The reward is also generated at this stage in part by calculating the Shannon entropy of the observation. Shannon entropy is a metric that describes the diversity of a population of samples. The laser array should be encouraged to go to places that produce more diverse observations, which correlate with higher entropy, because those regions typically contain more information of the shape. Specifically, a population of 19 whisker observations as illustrated in the first figure contains four species: central whisker No.0 if on shape; inner circle whiskers No.1-6 if on shape; outer circle whiskers No.7-18 if on shape; and lastly all the whisker observations that are off-shape. A high entropy observation will contain all species and have their occurrences distributed evenly. This entropy value is then modified to include a -1 penalty for each time step to encourage finding the terminal state efficiently.

Subsequently, we build a transition pair by bundling together the past state, action, reward and new state, and ship it to the replay buffer. We do not do gradient descent immediately on this transition pair because neighboring time steps are likely to produce highly correlated transition pairs which will make it hard for the DQN training process to converge. Instead, we store in a queue called replay buffer a finite number of most recent transition pairs and random sample from it batches of 32 transitions. In this way not only do we make the training process more stochastic, but also do we use the data generated more efficiently.

III. Data Preparation

In order to train the DQN, we need to populate the replay buffer with some minimally working DNN as classifiers to sample meaningful transitions from agent-environment dynamics. Therefore, we need to generate two dataset. The first one contains 1000 sequences all of 5 observations for randomly positioned triangles and hexagons each. The five observations within each sequence is randomly sampled (random laser head position) and one of these 5 observations is replaced with a corner observation, so that at least one observation within the sequence is significant for classification. This dataset is used to train a network that does shape classification on a single observation. Inputting a corner observation to this network gives a 90% confident guess.

The second dataset contains 5000 sequences and is used to trained the DQN controller network. Each sequence has different shape specification and the number of observations in a sequence depends on how fast the estimation of shape exceeds a confidence threshold. The maximum number allowed is capped at 20.

IV. Training Results

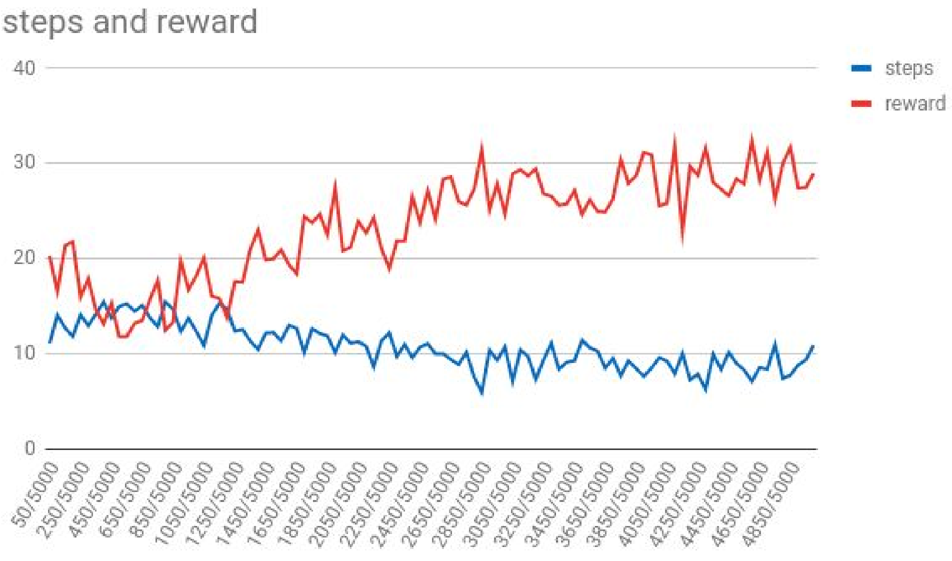
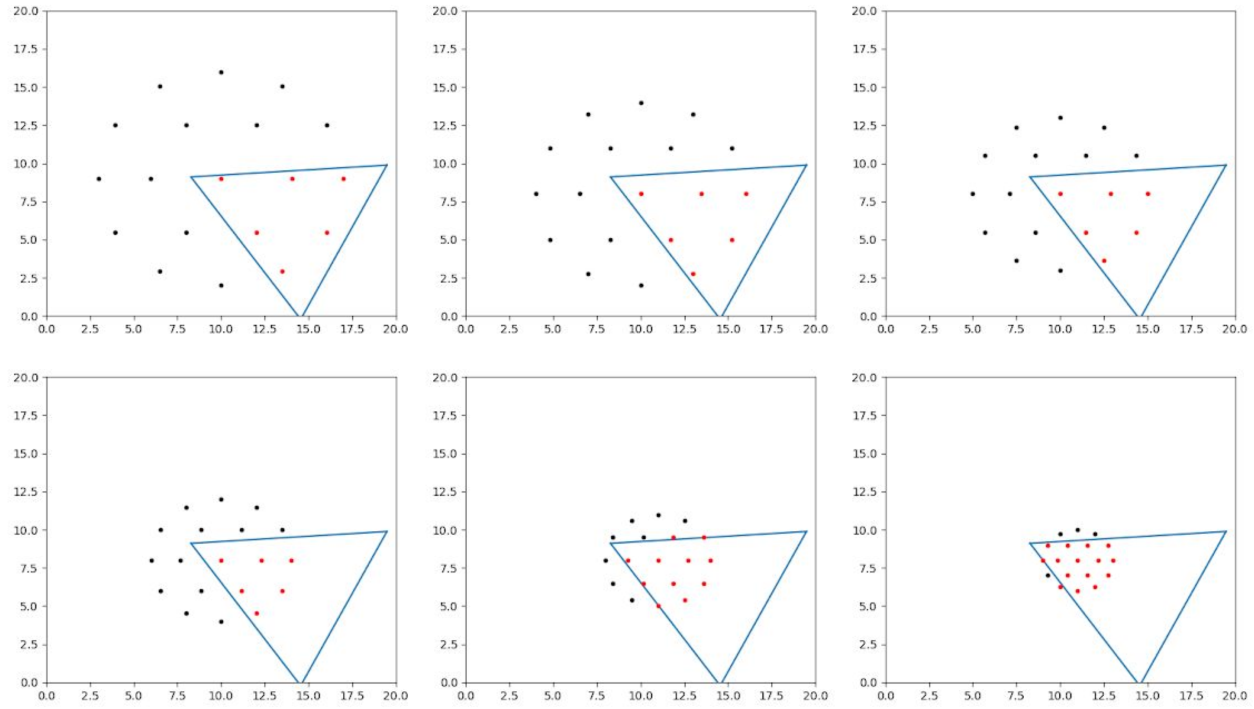
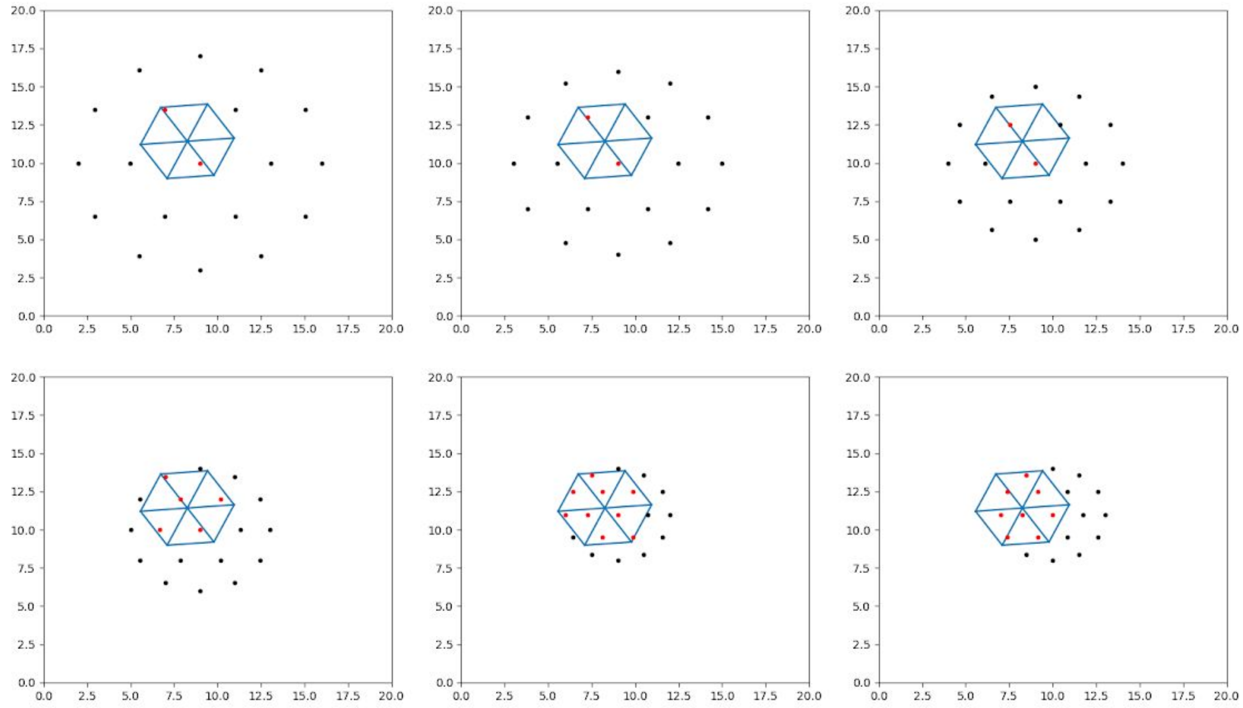


Figure above shows that the network is able to converge to collecting more reward while reducing the time steps it takes to do so. The horizontal axis of the chart labels episodic progress while the vertical axis is reward for red curve and time step number for blue. It took 8 minutes to train 5000 episodes (or sequences), and the resulting DQN is able to collect up to a reward of 30 within 10 time steps. Physically, that means the sequential decision process wastes no steps in driving the laser array to the most representational regions of the shape.





I list two sequences where the laser array identifies each shape correctly with 6 steps. The horizontal and vertical axes are the x and y dimensions of the exploration domain. The red dots are observations on shape and the black dots are observations off shape. Notice the observation dots are becoming closer to each other. That is due to the laser head moving closer to the objects, while the angles between lasers remain the same. The initial laser head position is placed centroid in the domain and most further away from the 2d shape plane. In this way, the initial lasers touch the plane in a most disperse manner and thus cover the most space of the exploration domain, making it easier for the DQN to locate the object. This effectively implements a search sequence where increasingly higher resolution scans follow initially low resolution scan as the laser head closes in on the object corners.

V. Future Work

For future work and code, please refer to my portfolio page at https://github.com/yanweiw/dqnActiveWhisking.