Case study 2 (option A):

1. Approach:

A literature study was followed by developing a reinforcement learning (RL) Pong environment using Pygame and OpenAI gym. Gym was chosen for a standardised interface.

Since learning from pixels was not explicitly specified, it was decided to reduce the complexity of the task by using the positions of the paddle and ball respectively as observations to the agent. The speed of the ball was later added to the observation. Next the actions were simplified to discrete binary enabling the agent to move the paddle left or right at constant speed.

Since the environment is similar to the fruit fly RL experiment, Cartpole, an exploratory exercise could be done using code that was successful for Cartpole.

The following approaches were tried:

- Gradient-based
  - Deep Q-Networks (DQN)
  - Policy Gradients (PG)
- Population-based
  - Random search
  - Evolution strategies (ES)
    - Simple ES
    - Covariance-matrix-adaptation ES (CMA-ES)
  - Genetic algorithm

As expected the problem could be solved with a simple brute-force random search, using only 3 coordinates and a linear function, as could be done with Cartpole.

However, to deliver a higher quality solution CMA-ES was chosen. From the population-based approaches simple ES did the worst. DQN outperformed PG, but the population-based algorithms were substantially superior.

2. Libraries used

pygame
Pygame is a Python wrapper module for the SDL multimedia library. It contains python functions and classes that will allow you to use SDL's support. SDL (Simple DirectMedia Layer) was created by Sam Lantinga. SDL is a cross-platform C library for controlling multimedia, comparable to DirectX. It has been used for hundreds of commercial and open source games. From examples found on the internet this was the easiest approach to create a Pong game.

gym
OpenAI Gym is a toolkit for developing and comparing RL algorithms. It is a nice standardised approach of defining a reinforcement learning environment compatible with higher level RL libraries.

numpy
NumPy is the fundamental package for array computing with Python.

cma
A stochastic numerical optimization algorithm for difficult (non-convex, ill-conditioned, multi-modal, rugged, noisy) optimization problems in continuous search spaces, implemented in Python.

3. Model architecture
The observations are mapped to actions via a multivariate linear function. This function contains 5 parameters (one for each observation) and an IF statement. If the dot product of the parameters and observation returns a positive value, it is mapped to the discrete action 1 else the function returns the discrete action 0.

4. Training
Since the requirement of the case study is to train an agent from zero-knowledge, RL was used. RL lies somewhere between supervised and unsupervised learning. On the one hand, it uses many well-established methods of supervised learning, such as deep neural networks for function approximation, stochastic gradient descent, and back-propagation, to learn data representation. On the other hand, it usually applies them in a different way.

The following diagram illustrates the process:

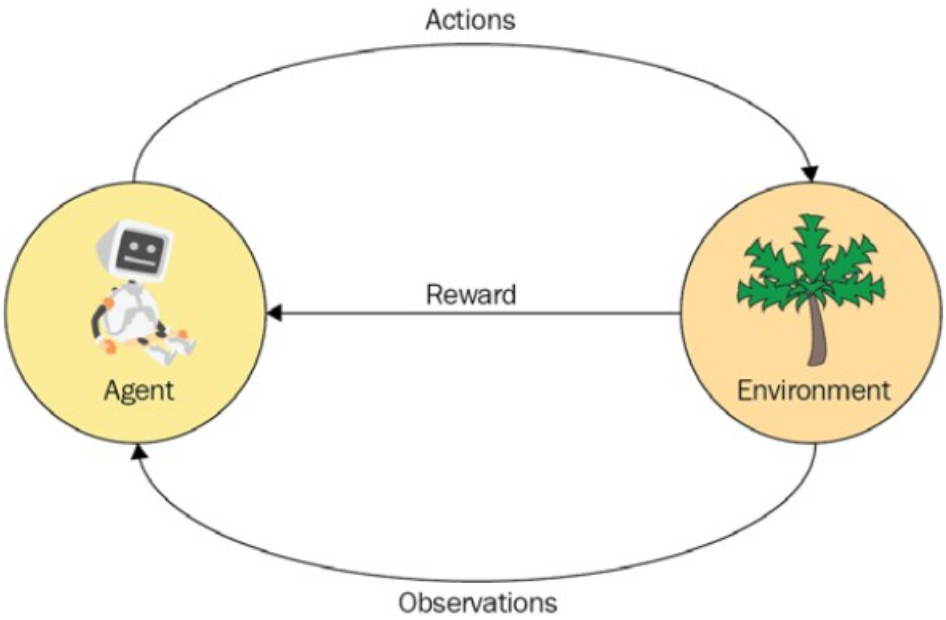

Figure 1: RL entities and communication

An agent interacts with an environment via actions. It receives information from the environment in the form of rewards and observations. In the case of Pong, the agent is the linear function, the environment is the Pong game, the actions are discrete values of 0 and 1 moving the paddle left and right and the rewards, the score for hitting the ball.

The reward signal directs the agent towards the objective. In the case of Pong receiving the maximum score.

Refer to pong_cmaes.py

As mentioned the CMA-ES was used for optimisation. Please refer to:
https://blog.otoro.net/2017/10/29/visual-evolution-strategies/
for a visual explanation and theory.

Training of the agent was done by playing the game a few times to get an average value for the fitness since the environment is initialised with random ball velocity to ensure that the game changes. Otherwise with the constant speed and angle of the ball (as specified) the game will always be the same and that is not interesting. The averaging ensures that the agent can play variations of the game. There is also a step limit imposed, to speed up the process.

A population of 5 individuals was chosen. Each individual plays the game 5 times and the results are averaged. Since a reward per environment step worked well for Cartpole, the same was used. There is facility to use the score for experimentation, but this was not pursued. A step limit of 10000 was imposed, since it proved to produce a proficient agent. The CMA-ES solver was initialised with 0.1 standard deviation.

Performance is printed to screen every 10 steps. Sometimes the random initialisation slows down the learning, therefore if the steps increase beyond 100, the solver is re-initialised. It is capable of finding the optimum solution before 100 steps are reached. The approach was chosen as a better illustration of the solution effectiveness than choosing a random seed.

The code runs so fast that training and testing can be done together, but if only the trained agent want to be tested (as specified), a solution array is provided. Line 97 and 98 can be commented out and line 99 re-instated. The game will continue until 10000 steps are reached or the agent drops the ball, which could be awhile.

5. Bonus Questions

1. From the population based family a genetic algorithm, simple ES or random search would have worked, by just substituting the solver.

From the gradient-based options DQN could work, by using the same approach with deep neural networks for function approximation, stochastic gradient descent, and back-propagation. The vanilla DQN uses a replay-buffer to sample experiences from in order to try to enforce independent identically distributed experiences for the agent. Q-learning is applied via the Bellman equation.

2. In supervised learning observations are mapped to labels. The basic question is, how do you automatically build a function that maps some input into some output when given a set of example pairs. In unsupervised learning the there are no labels and the main objective is to learn some hidden structure of the data set.

3. In classification, a discrete class is predicted, e.g. 0 or 1. Is the mail spam or ham? In regression a continuous value is predicted, e.g. what is the temperature?

4. Class imbalance put accuracy as a metric out of business. Standard accuracy no longer reliably measures performance, which makes model training trickier.

5. Things to try to improve the data set:

- Balance the data set by either up-sampling the minority class or down-sampling the majority class.
- Change your performance metric, Area Under Receiver Operating Characteristic Curve (AUROC) is recommended as a general purpose metric.
- Penalize algorithms (Cost-Sensitive training). A popular one is Penalized-SVM from sklearn.
- Use Tree-based algorithms
- Create synthetic samples (data augmentation)
- Combine minority classes into a single larger class if they can be grouped together.
- Reframe the problem from classification to anomaly detection.

6. The term ‘deep learning’ from a machine learning perspective refers to the use of multi-layered neural networks with many hidden layers. The key differences between these algorithms and non-deep learning are:

- many more hyper-parameters
- much higher computational requirement
- many more parameters
- much higher complexity for optimisation
- many pitfalls (vanishing/exploding gradients problems)
- more prone to over-fitting
- performance improves beyond non-deep learning approaches with more data

7. As the score increases the speed of the ball increases. As the time increases more surfaces change from constant angle bouncing to random angles. All these changes can easily by done with Pygame, as access to all the parameters is available.