AI_G_AI

Implementation concept:

(1) While the game is running, pass data(observations) to from labview to python using python node.

(2) In python, create a model(neuro network).

(3) Since the game is continuously running, the training must happen simultaneously.

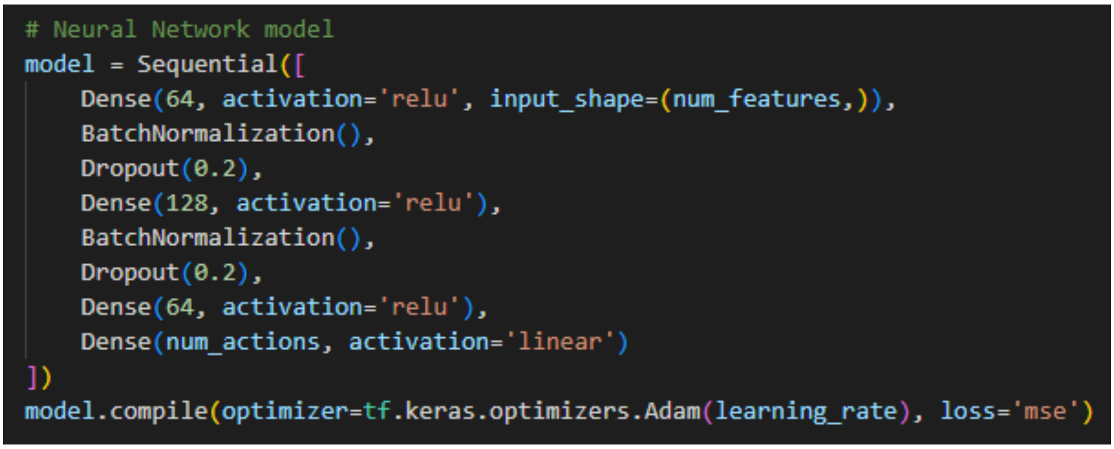
(4) A replay buffer is used to collect the data from previous states.

(5) Only train and save the model after we call the train function ‘steps’ of times.

  (a) To reduce the amount of time to train, since training slows down the game.

(6) Load the trained model and predict the movement of player.

Model and hyperparameters:



```
# Neural Network model
model = Sequential([
    Dense(64, activation='relu', input_shape=(num_features,)),
    BatchNormalization(),
    Dropout(0.2),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dense(num_actions, activation='linear')
])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate), loss='mse')
```

Learning rate = 0.001

Discount_factor = 0.9

Epsilon = 0.2

Decay = 0.95

Replay buffer size = 10000

Batch size = 200

Action/Observation Space:

Observations (5):

(a) Ball_vector[0]

(b) Ball_vector[1]

(c) Ball_position[0]

(d) Ball_position[1]

(e) Player_top_left_y_value

Actions:

(a) 0 (UP)

(b) 1 (DOWN)

Reward Shaping:

(a) Self-Score: +1

(b) Opponent Score: -2

(c) No Score: 0

Note:

To accelerate the training process the data is passed to python every 20 frames.
Training took around 20000 frames (10000 observations to replay buffer)

Environment:
Python 3.10
Tensorflow 2.15.0