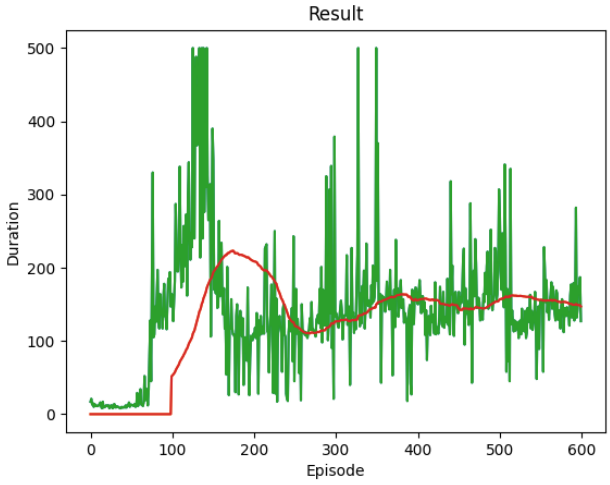
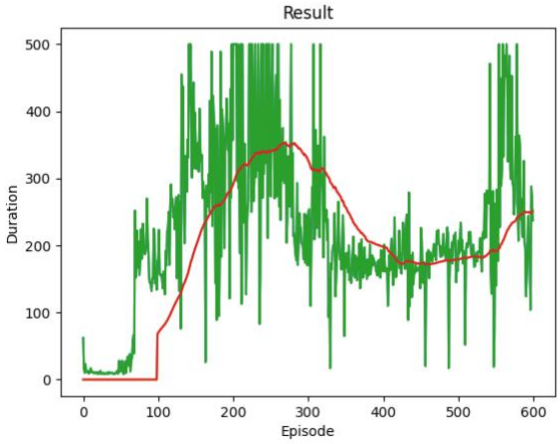
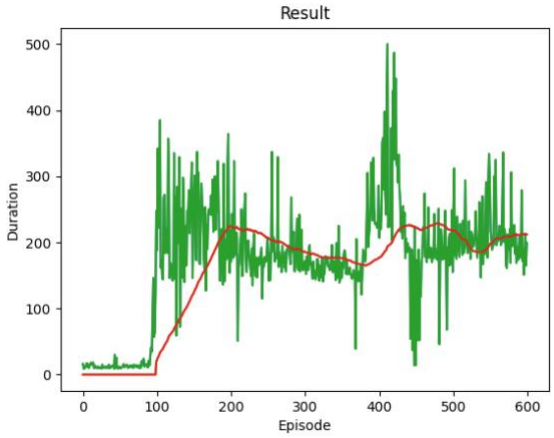
ReLu



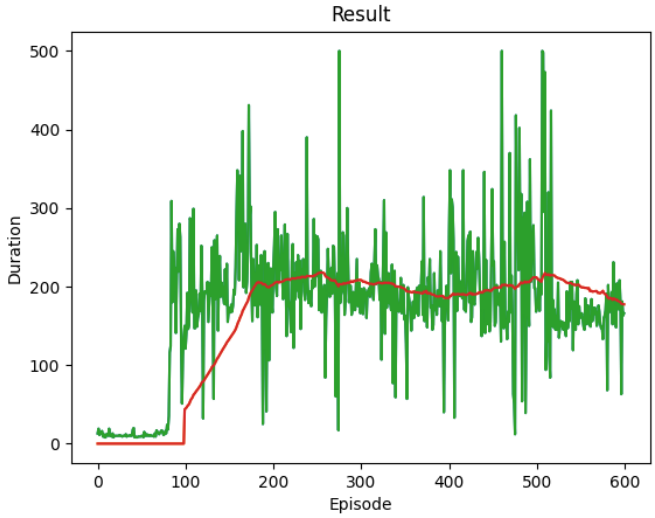
1) Training results for LR: 0.001 ; TAU: 0.005 ; DECAY: 200
    a. Runtime = 92.81044602394104



2) Training results for LR: 0.001 ; TAU: 0.005 ; DECAY: 200
    a. Layer 1 = 64 neurons ; layer 2 = 128
    b. Runtime = 130.63996171951294

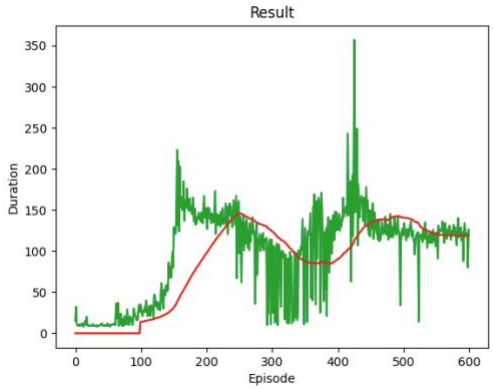


3) Training result for LR: 0.001 ; TAU: 0.005 ; DECAY: 300
    a. 111.84322428703308



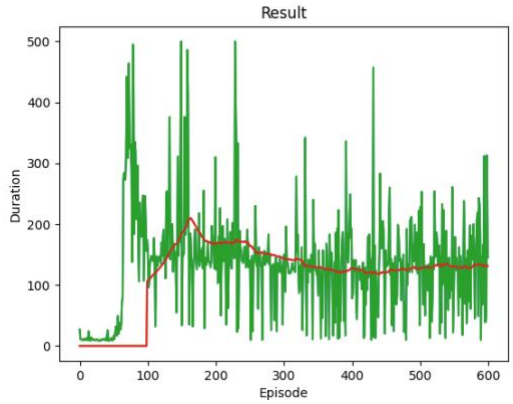
4) Training result for LR: 0.001 ; TAU: 0.005 ; DECAY: 100
    a. 97.27127981185913

Sigmoid



5) Training result for LR: 0.001 ; TAU: 0.005 ; DECAY: 100
    a. 77.81094598770142

Leaky Relu



1) Training result for LR: 0.001 ; TAU: 0.005 ; DECAY: 100
    a. 79.53551506996155

- Architecture of neural net
    - This DQN implementation involved three connected layers with a 128 unit input later that maps the state space, a hidden layer with 128 input units and 64 output units using the ReLu activation function, and a 64 unit output layer.
- Hyperparameters

```
BATCH_SIZE = 128# Batch size
GAMMA =  0.99# Discount factor
EPS_START = 0.9 # Starting value of epsilon
EPS_END = 0.600 # Minimum value of epsilon
EPS_DECAY = 100# Rate of decay for epsilon
TAU = 0.0050# Target network update rate
LR = 0.001 # Learning rate
```

- Training progression and experimentation

- I ended up spending quite a while manipulating the hyperparameters and activation functions beyond what is displayed above and had difficulty finding an optimal function that performed consistently. When running the program several times, the combinations which had maxed out the duration failed to repeat success, particularly in the case of the inverted architecture with the 64 unit layer preceding the 128 bit layer. The ReLu activation function appeared to provide the most consistency in its results and decided to stick with the activation function, along with the 128 – 64 unit progression in the three layer architecture. The optimization of the parameters and architecture certainly appears tedious as I easily spent over an hour experimenting with combinations beyond what is included above and am interested to learn how these variables could be optimized for learning. After the first few times executing the program, it also became apparent the time and computational power needed to execute these programs as the runtimes lasted over 2 minutes in some cases. Very interesting to see it play out in action and is thought provoking when considering how neural nets can apply to larger and more complex scenarios.