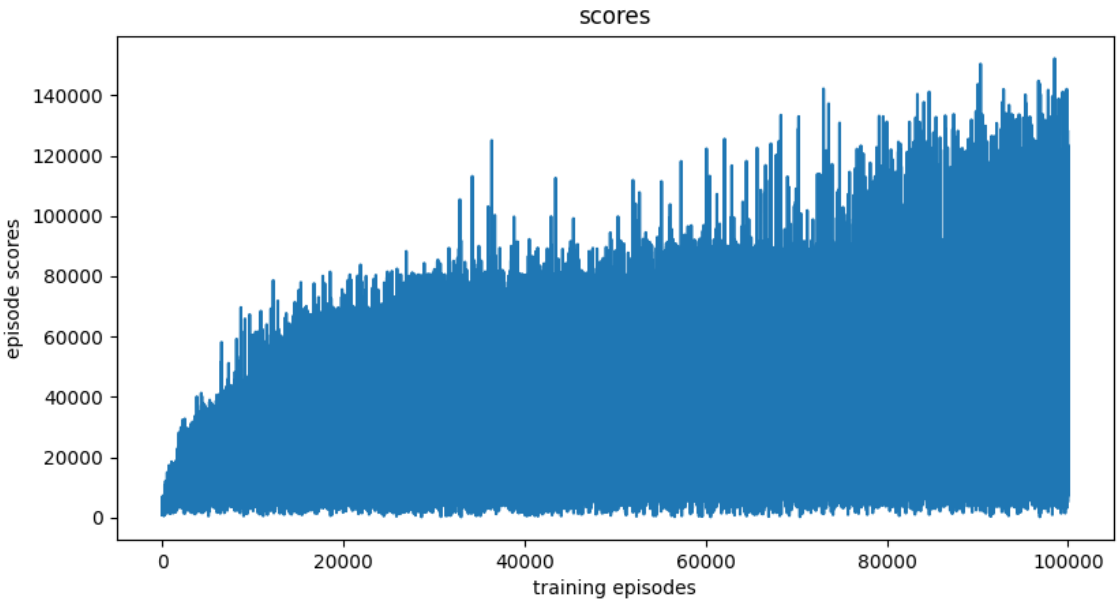
HW2 Report

1. A plot shows episode scores of at least 100,000 training episodes (10%)



2. Describe the implementation and the usage of n-tuple network. (10%)

N-tuple network 是一種快速且有效率的神經網路架構，能有效的近似 pattern 的分類。把 2048 遊戲中的 board 作為當前 state，我們不直接查出現在的 board 幾分，而是查出每一個特徵的分數，再把這些特徵加起來代表盤面的分數。特徵的選擇方式是根據 board 上特定位置的方塊來決定的，可以用一些 tuple 來表示這些特定方塊。下圖示一些 4-tuple 的組合和如何在程式中加入這些 4-tuple。

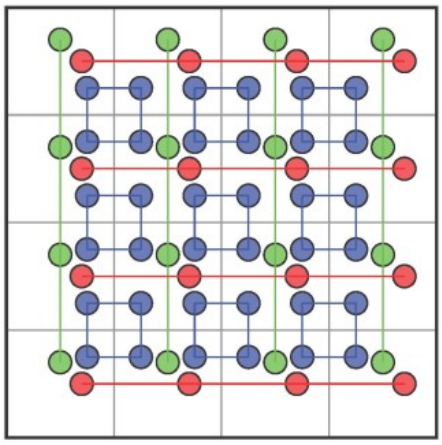


Figure 8: The n-tuple network consisting of all possible horizontal and vertical straight 4-tuples (red and green, respectively), and all possible 2×2 square tuples (blue).

```
for(int i=0;i<4;i++){
  tdl.add_feature(new pattern({ 0+i*4, 1+i*4, 2+i*4, 3+i*4 }));
  printf("%d %d %d %d\n", 0+i*4, 1+i*4, 2+i*4, 3+i*4 );
  tdl.add_feature(new pattern({ 0+i, 4+i, 8+i, 12+i }));
    printf("%d %d %d %d\n", 0+i, 4+i, 8+i, 12+i);
}
for(int i=0;i<3;i++){
  for(int j=0;j<3;j++){
    tdl.add_feature(new pattern({ 0+j+4*i, 1+j+4*i, 4+j+4*i, 5+j+4*i }));
    printf("%d %d %d %d\n", 0+j+4*i, 1+j+4*i, 4+j+4*i, 5+j+4*i);
  }
}
```

3. Explain the mechanism of TD(0). (5%)

Temporal-difference (TD) Learning 是一種強化學習的更新方法，它集成了 Monte Carlo 思想和 Dynamic Programming 思想，像 MC 方法一样，TD 方法不需要環境的 model，它直接從經驗中學習，像 DP 方法一樣，TD 方法不需要等到最終的 outcome 才更新 model。

TD 方法只要等到下一個 time step ，即在时刻 t + 1，TD 方法立刻形成一個 target 並使用觀測到的 reward R(t+1)和估測的 V(t+1)進行更新，最簡單的 TD 方法是 TD(0)：

$$V(S_t) \Leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

4. Explain the TD-backup diagram of V(after-state). (5%)

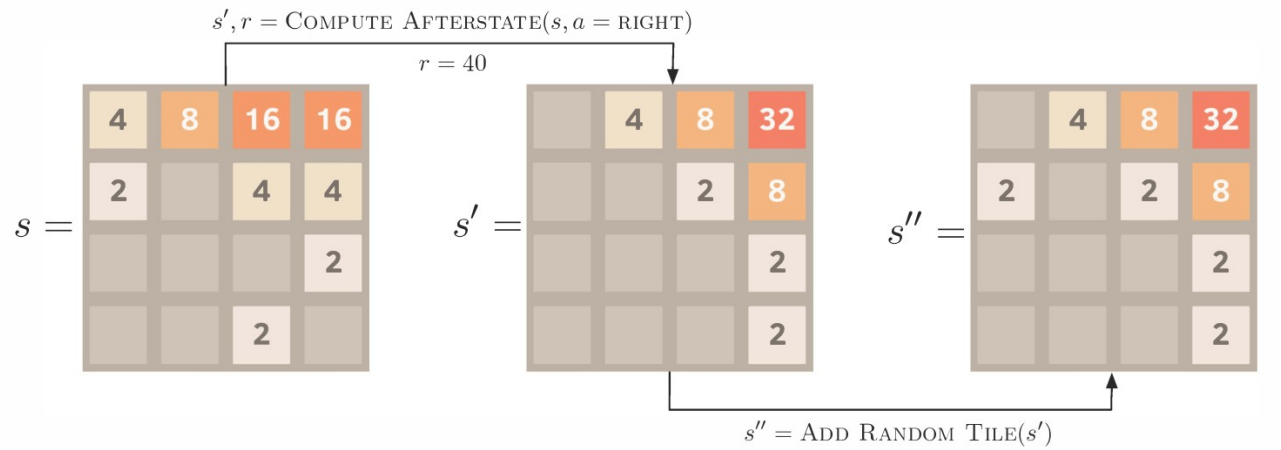


Figure 2: A two-step state transition occurring after taking the action $a = \text{RIGHT}$ in the state s.

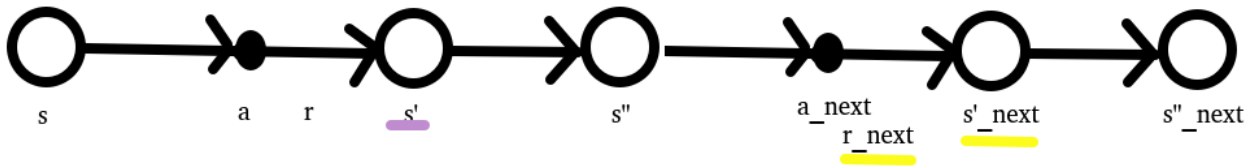
2048 遊戲的 state transition 如上圖，s’是在 s 採取了 a 後的下一個 state，s’’是系統自動 popup 一隨機 tile 之後的 state。而 V(after-state)指的是在 state evaluation 時，它使用了 afterstates 的估測值來做更新。其 TD-backup diagram 如下圖，用畫黃色底線的部份來更新紫色底線的估測值。

function LEARN EVALUATION(s, a, r, s', s'')

$a_{next} \leftarrow \arg\max_{a' \in A(s'')} \text{EVALUATE}(s'', a')$

$s'_{next}, r_{next} \leftarrow \text{COMPUTE AFTERSTATE}(s'', a_{next})$

$V(s') \leftarrow V(s') + \alpha(r_{next} + V(s'_{next}) - V(s'))$

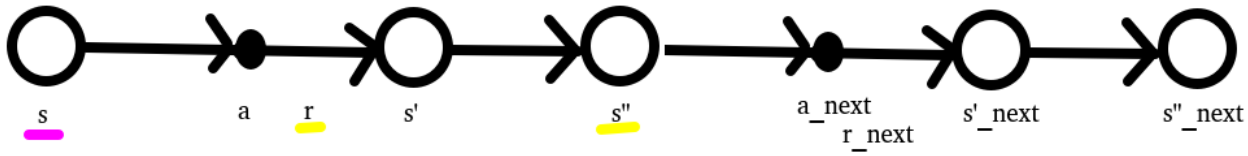


5. Explain the action selection of V(after-state) in a diagram. (5%)
V(after-state)方法，agent 在選擇 action 時會根據以下 policy。其中 T(s, a)代表的是，在 state s 時採取 action a，會得到 state s’這個結果，這個過程的映射關係。

$$\pi(s) = \arg\max_{a \in A(s)} [R(s, a) + V(T(s, a))]$$

6. Explain the TD-backup diagram of V(state). (5%)
V(state)指的是在 state evaluation 時，它使用了 state 的估測值來做更新。其 TD-backup diagram 如下圖，用畫黃色底線的部份來更新紫色底線的估測值。

$$\textbf{function } \text{LEARN EVALUATION}(s, a, r, s', s'')$$
$$V(s) \leftarrow V(s) + \alpha(r + V(s'') - V(s))$$



7. Explain the action selection of V(state) in a diagram. (5%)
V(state)方法，agent 在選擇 action 時會根據以下 policy。這個 agent 需要了解環境 model 的狀態轉移機率 P。每一次 agent 做決策時，它需要計算在採取了這個 action 後，所有可能 states 的估測值。並據此選擇出期望值最大的 action。因這個窮舉的步驟，它執行效率比較慢。

$$\pi(s) = \arg\max_{a \in A(s)} \left[R(s,a) + \sum_{s'' \in S} P(s,a,s'')V(s'') \right]$$

8. Describe your implementation in detail. (10%)
只有修改參考程式碼中的兩個部份。
(1) 在做 action 選擇時，將 policy 中，窮舉所有可能性，並計算期望值的部份實作出來。

```
state select_best_move(const board& b) const {
  state after[4] = { 0, 1, 2, 3 }; // up, right, down, left
  state* best = after;
  for (state* move = after; move != after + 4; move++) {
    if (move->assign(b)) {
      // TODO
      board next = move->after_state();
      int count = 0;
      double sigma_P_V = 0;
      for (int i = 0; i < 16; i++){ //All Possibile Next State
        if (next.at(i) == 0) {
          count ++ ;
          next.set(i,2);
          float V1 = estimate(next);
          next.set(i,0); //clear
          next.set(i,1);
          float V9 = estimate(next);
          next.set(i,0); //clear

          sigma_P_V += (V1*0.1 + V9*0.9);
        }
      }
      if(count!=0){
        move->set_value(move->reward() + sigma_P_V/count);
      }
      if(count==0){
        move->set_value(move->reward() + 0);
      }
      // ~TODO
```

(2) 把迭代更新的地方也改成 state 的方法。

```
void update_episode(std::vector<state>& path, float alpha = 0.1) const {
  // TODO
  float exact = 0;
  for (path.pop_back() /* terminal state */; path.size(); path.pop_back()) {
    state& move = path.back();
    float error = move.reward() + exact - estimate(move.before_state());
    debug << "update error = " << error << " for after state" << std::endl << move.after_state();
    exact = update(move.before_state(), alpha * error);
  }
  // ~TODO
}
```

9. Other discussions or improvements. (5%)
參考以下文章，使用 T 字形狀的 5-tuple。

https://www.mxeduc.org.tw/scienceaward/history/projectDoc/18th/doc/SA18-065_final.pdf?fbclid=IwAR2ogNjtCRuspYBkydsLYJJ1orCYKAZ5k3jBQN6NcpmdSNb0MgHjPGft0Qk