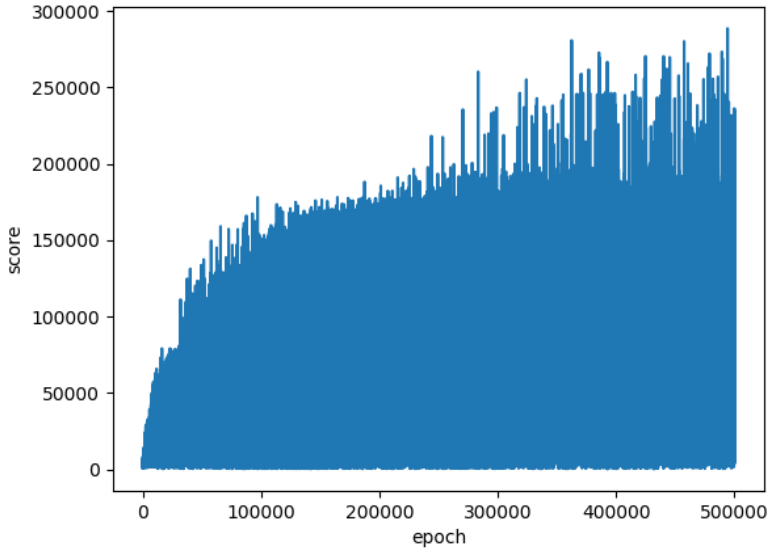
Report (60%)

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



2. Describe the implementation and the usage of n-tuple network. (10%)
   透過sample code的 add_feature將4個6-tuple存入，又由於一個feature形狀經過旋轉
   A:上下左右鏡像對稱後皆是同構的，所以對一個n-tuple我們會透過smaple code已寫好的8種isomorphic的weight合來估計這一個tuple對這個盤面的value 。

   此範例共用了4個6-tuple，因此一共用了4*8=32個weight來估計一個state的value至於更新的時候也一樣 將alhpa*error這個值平均分配給這32個對應的weight做更新。

   使用n-tuple的用途是因為若要紀錄2048的完整盤面需要的空間很大，且有時候只需要透過局部的盤面即可分析這個盤面的好壞，因此利用n-tuple的方式來預估整個state，能有效減少記憶體使用量。

3. Explain the mechanism of TD(0). (5%)
   A:TD(0)是透過TD-target=(Rt+1 + r V(St+1))來預測 V(St)，而這兩者的差又叫做TD-error=V(St)-Rt+1+rV(St+1)，而TD(0)就是透過error*learning rate去對value做更新的，藉由這樣子不斷透過error修正value的方法最終每個state value將會逼近true value，且TD(0)是model-free的，他是透過預測的value來更新預測的value，不需要知道env只需要透過value的估測即更新即可學習，還有一個很大的特色是他不需要等到episode完全做完，即可更新每個state value。

4. Explain the TD-backup diagram of V(after-state). (5%)
   A:after state在此例子代表滑動後還未pop tile的狀態，TD-target是Rnext+V(S'next)，也就是透過下一個after_state(S'next)加上該state所獲得的reward來預估真正的V(S')，透過TD-error(Rnetxt+V(S'next)-V(S')來對V(S')做更新，所以完整的式子才會寫成

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

5. Explain the action selection of V(after-state) in a diagram. (5%)
   A:after state在選擇action的時候較before state簡單，這邊的概念和Q(s,a)相同，透過去計算每個action的Q(s,a)，再去選其中最大的那一個即是選到最好的action，這邊的Q(s,a)算法只要直觀的透過reward+V(S')計算即可得到，也就是透過該action得到的reward加上這個after state的value。
6. Explain the TD-backup diagram of V(state). (5%)
   A:這邊的state亦即before state，在此例子中代表還沒有滑過動作的state，在做更新的時候和after state大同小異，只是所有的state value都改為before state，這邊是透過V(S")加上變成S"這個 before state之前after state所獲得的reward，來預估V(s)，因此透過TD-error的方式做更新完整的式子即可寫成：

$$V(s) \leftarrow V(s) + \alpha(r + V(s'') - V(s))$$

7. Explain the action selection of V(state) in a diagram. (5%)
   A:這邊在選擇action的Q(s,a)比較複雜，為model-base的方法，由於透過next before state估計當前before state的時候會因為環境變量的影響，會random pop tile，且pop出2跟4的機率是9:1，因此我們無法確認下一個before state到底是誰，因此下一個before state的value是透過weighted sum的方式，把所有可能的s"的機率乘上對應的v(s")加起來後，再加上該action獲得的reward來預估當前的Q(s,a)，之後同before state，把所有的action跑過一遍去選最大Q(s,a)的action即可。
8. Describe your implementation in detail. (10%)
   A:一共實做了五個TODO function
   其中estimate update indexof可以直接使用2048 demo after state ver. 的沒有問題
   而其中不同的地方有

```
// find the zero index
int space[16], num = 0;
for (int i = 0; i < 16; i++){
    if (as.at(i) == 0) {
        space[num++] = i;
    }
}
float t = 0; // sum of all possible of next before state
for (int i = 0 ; i<num; i++){
    // random tile 2 is 90%
    as = move->after_state();
    as.set(space[i],1);
    t = t + ((0.9)*estimate(as))/num;
    // random tile 4 is 10%
    as = move->after_state();
    as.set(space[i],2);
    t = t + ((0.1)*estimate(as))/num;
}

move->set_value(move->reward() + t); // v = r + sum of all possible of next before state
```

   1.select_best_move : 由於before state 跟 after state的 evaluate 不同，因此必須把所有可能的next before state給weighted sum 起來，所以再檢測滑動上下左右四個action後的after state時，先透過在board中寫好的at function來找哪幾個index是0，並紀錄它的數量(num)，之後又因為前提 pop出2和4的機率是9:1，所以random pop完before state後，再做weighted sum的時候就以2和4對應的機率除以num來當作random pop出這個數字在這個位置的機率，再透過estimate來計算該before state的value後，把所有的before state做weighted sum即可。

```cpp
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();
        board bs = move.before_state();
        float error = exact -(estimate(bs)); // error = v[s].exact - v[s].predict
        debug << "update error = " << error << " for before state" << std::endl << move.before_state()
        exact = move.reward() + update(move.before_state(), alpha * error); //exact = r + v(s)
    }
}
```

2.update_episode:因為before state的algo.所有參與state更新的都是before state因此這邊所有的state都要是before state的型態，實做時是從尾到頭更新，由於terminal state不需要所以一開始先pop_back掉。根據algo.，透過target-prediction得到error後藉由寫好的update更新value的值，且update後該function會回傳total update value，再藉由total update value +reward當作接著要繼續計算error的TD-target不斷以此迭代由尾到頭做更新。

9. Other discussions or improvements. (5%)

A:實做時透過ofstream把每個episdoe的score傳出來，再執行完sample code後會產生score.txt，透過畫圖操作較簡易的python，plot_graph.py完成score plot graph。
且在make_statistic中從t=11(2^11=2048)抓2048的累積成功率：accumulate accuracy，如此一來在跑大量episode的情況下就不用一筆一筆去人眼搜索找最高的win rate，直接看output結果即可。

```cpp
for (int t = 1, c = 0; c < unit; c += stat[t++]) {
    if (stat[t] == 0) continue;
    int accu = std::accumulate(stat + t, stat + 16, 0);
    if(t==11) {
        if(tmax < (accu * coef)) {
            tmax = accu*coef;
            e    =  n;
        }
    }
    info << "\t" << ((1 << t) & -2u) << "\t" << (accu * coef) << "%";
    info << "\t(" << (stat[t] * coef) << "%)" << std::endl;
}
```

```cpp
cout<<"episode = "<<e<<" , accur = "<<tmax<<endl;
```

下圖為train 50萬次的結果：

```
 500000  mean = 93646.1  max = 236128
         128     100%    (0.2%)
         256     99.8%   (0.4%)
         512     99.4%   (1.9%)
         1024    97.5%   (6.9%)
         2048    90.6%   (11%)
         4096    79.6%   (37.3%)
         8192    42.3%   (42.1%)
         16384   0.2%    (0.2%)
 episode = 497000 , accur = 93
 [1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm}
  0<"/tmp/Microsoft-MIEngine-In-ytrzp3qf.b8e" 1>"/tmp/Microsoft-MIEngine-Out-6vy9a
 x9s.gyi"

 Desktop/DLP_HW2/cp took 2d 15h 22m 20s
```