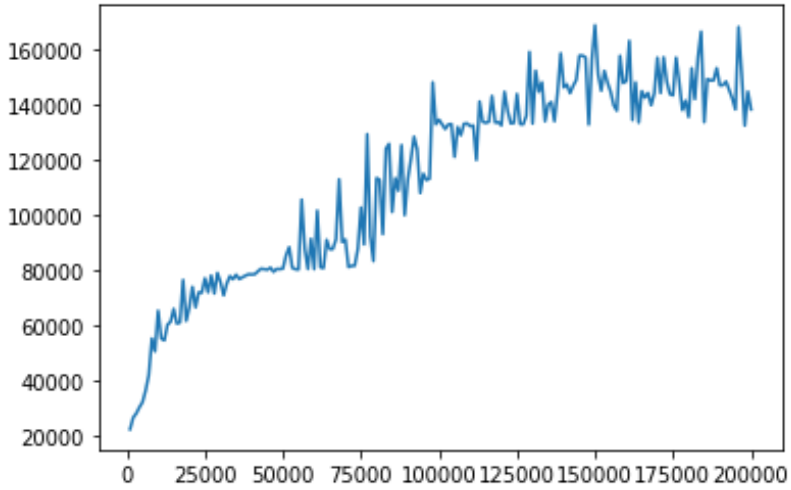
Deep learning
Lab2: 2048-TD

1. A plot shows scores (mean) of at least 100k training episodes



2. Describe the implementation and the usage of nn-tuple network:
因為有可能的 board 組合非常多，計算每一種 board 的估計值會使記憶體用量無法負荷。因此選取一塊位置作為 board 的 feature，而計算估計值時也只會對 board 上的那一小塊 feature 做操作，即 n-tuple network。對 board 取 feature 時，同樣的位置應該要在 rotate*4 及 flip*2，總共八種 isomorphism 中各取一次，相加之後才是完整的估計值。因此相同的 feature 產生相同的估計值，而不同之處讓兩個 board 之間產生差異。

Code 如下:

```
for (int i = 0; i < 8; i++)
{
    board idx = 0xfedcba9876543210ull;
    if (i >= 4)
        idx.mirror();
    idx.rotate(i);
    for (int t : p)
    {
        isomorphic[i].push_back(idx.at(t));
    }
}
```

找出 feature 在 board 上的 isomorphic pattern

```cpp
size_t indexof(const std::vector<int> &patt, const board &b) const
{
    // TODO
    size_t idx = 0;
    for (int i = 0; i < patt.size(); i++)
    {
        idx |= b.at(patt[i]) << (i * 4);
    }
    return idx;
}
```

isomorphic pattern 在這個 board 的 index

```cpp
virtual float update(const board &b, float u)
{
    // TODO
    float u_each = u / iso_last;
    float value = 0;
    size_t idx;
    for (int i = 0; i < 8; i++)
    {
        idx = indexof(isomorphic[i], b);
        operator[](idx) += u_each;
        value += operator[](idx);
    }
    return value;
}
```

根據 idx 找出對應的 weight 再計算這個 board 的估計值

3. Explain the mechanism of TD(0)

從這個 episode 倒數第二個 move 開始，計算這個 move 和下一個 move 的 before state error，用 error 更新 state 的期望值(expect)給上一個 state 使用

For each episode,

```
Initialize (before-)state s
While s is not terminal do
  a ← argmax_a' EVALUATE(s, a')
  r, s', s'' ← MAKE_MOVE(s, a)
  STORE(s, a, r, s', s'')
  s ← s''
End While
For (s, a, r, s', s'') from terminal down to initial do
  LEARN_EVALUATION(s, a, r, s', s'')
End For
```

perform TD backup

4. Describe your implementation in detail including action selection and TD-backup diagram

```
void popup()
{
    int space[16], num = 0;
    for (int i = 0; i < 16; i++)
        if (at(i) == 0)
        {
            space[num++] = i;
        }
    if (num)
        set(space[rand() % num], rand() % 10 ? 1 : 2);
}
```

窮舉所有 2 和 4 出現在 empty tile 的結果

```
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
            int totol = 0;
            board next = move->after_state();
            std::vector<int> pos;
            for (int i = 0; i < 16; i++)
            {
                if (next.at(i) == 0)
                {
                    pos.push_back(i);
                }
            }
            if (pos.size() != 0)
            {
                for (int i = 0; i < pos.size(); i++)
                {
                    board temp = next;
                    temp.set(pos[i], 1);
                    float two = estimate(temp);
                    temp.set(pos[i], 2);
                    float four = estimate(temp);
                    totol += (0.9 * two + 0.1 * four) / float(pos.size());
                }
            }
            else
            {
                totol = estimate(move->after_state());
            }
            move->set_value(move->reward() + totol);
            if (move->value() > best->value())
                best = move;
        }
        else
        {
            move->set_value(-std::numeric_limits<float>::max());
        }
        debug << "test " << *move;
    }
    return *best;
}
```

加總所有可能性的估計值，並計算哪個 move 的結果最好

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

計算 TD-error, 並更新 V(state)的期望值

5. Others

```
tdl.add_feature(new pattern({0, 1, 2, 3, 4, 5}));
tdl.add_feature(new pattern({4, 5, 6, 7, 8, 9}));
tdl.add_feature(new pattern({0, 1, 2, 4, 5, 6}));
tdl.add_feature(new pattern({4, 5, 6, 8, 9, 10}));
tdl.add_feature(new pattern({1, 4, 5, 6, 9})); //十字
tdl.add_feature(new pattern({0, 2, 5, 8, 10})); //X
```

除了原本的 4 個 features,另位新增一個十字的 feature 和一個 X 的 feature