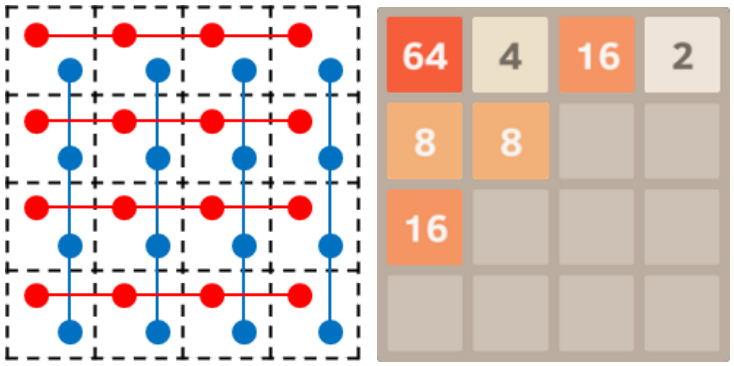
TCG Project 2 Report

1. Network Design

I use default network design, which uses straight lines as tuples. There's one tuple for each column and row to extract the value of the whole board. The pattern is shown in the following graph.



2. TD-learning

The core idea of TD-learning is updating the weight by the difference between estimated value of current state and the estimated value of the next state plus reward. (Since if your network is perfect, the error should be zero).
The main part of the code is shown in the following screenshot.

```
    if(replay_buffer.empty()) return;
    if (alpha == 0) return;
    adjust_value(replay_buffer[replay_buffer.size() - 1].after, 0);
    for(int t = replay_buffer.size() -2; t >= 0; t--) {
        adjust_value(replay_buffer[t].after, replay_buffer[t+1].reward
            + estimate_value(replay_buffer[t+1].after));      You, seco
    }
}
```

```
void adjust_value(const board& after, float target) {
    float current = estimate_value(after);
    float error = target - current;
    float adjust = alpha * error;
    net[0][extract_feature(after, 0, 1, 2, 3)] += adjust;
    net[1][extract_feature(after, 4, 5, 6, 7)] += adjust;
    net[2][extract_feature(after, 8, 9, 10, 11)] += adjust;
    net[3][extract_feature(after, 12, 13, 14, 15)] += adjust;
    net[4][extract_feature(after, 0, 4, 8, 12)] += adjust;
    net[5][extract_feature(after, 1, 5, 9, 13)] += adjust;
    net[6][extract_feature(after, 2, 6, 10, 14)] += adjust;
    net[7][extract_feature(after, 3, 7, 11, 15)] += adjust;
}
```

3. The Training Process

I use the script from README to train the agent. The whole training process took about half a day. I reached a decent (but not amazing) score of 98% 2048 win rate and the max tile value of 28657. Training Time is the key in my case.

Here's the script I use and the screenshot of my result.

```
You, an hour ago | 1 author (You)
# ./2584 --total=0 --play="init save=weights.bin" # generate a clean network
for i in {1..100}; do
    ./2584 --total=10000 --block=1000 --limit=1000 --play="load=weights.bin save=weights.bin alpha=0.005" | tee -a train.log
    ./2584 --total=1000 --play="load=weights.bin alpha=0" --save="stat.txt"
    tar zcvf weights.$(date +%Y%m%d-%H%M%S).tar.gz weights.bin train.log stat.txt
done      You, 6 days ago • Add project 2 code
```

```
> ./2584 --total=1000 --play="load=weights.bin alpha=0" --save="stat.txt"
2048-Demo: ./2584 --total=1000 --play=load=weights.bin alpha=0 --save=stat.txt

1000    avg = 214815, max = 459870, ops = 2831028 (2228353|5260071)
        610     100%    (0.3%)
        987     99.7%   (0.3%)
        1597    99.4%   (1.1%)
        2584    98.3%   (4.2%)
        4181    94.1%   (10.5%)
        6765    83.6%   (19.6%)
        10946   64%     (26.4%)
        17711   37.6%   (37.5%)
        28657   0.1%    (0.1%)
```