Heuristic Analysis:

Since the neural net could not be prepared in time, I divided the problem into some heuristics and manually tested them. The main group of these can be seen as 'mixins' to the base improved heuristic which aims to maximize number of player moves vs number of opponents moves.
1) Distance of opponent to player- this implies the following options:
1.1 Run away from opponents
1.2 Run toward opponents
2) Number of common moves – implying an taking squares away from opponent.
3) Number of of moves on edge of board as a penalty (with corners being a double penalty)
4) Some abstraction of how far one can see ahead. Since the board is predefined, a quick calculation can be hard coded for this.

In addition, it is probably important that the winning evaluation function is winning atleast 80% of games, and wins against every baseline opponent. The baseline of best is set at AB_improved which has a winrate in the pool of ~75%

	Maximize own score + run toward		Maximize own score +run away +penalize edges		Maximize own score +run away		difference in moves - 2 deep + difference in moves	
Random	10	0	10	0	10	0	10	0
MM_Open	10	0	8	2	7	3	10	0
MM_Center	10	0	10	0	10	0	9	1
MM_Improved	9	1	8	2	7	3	8	2
AB_open	5	5	5	5	4	6	5	5
AB_center	6	4	5	5	7	3	4	6
AB_Improved	5	5	5	5	7	3	6	4
Wins over best	50.00%		50.00%		70.00%		60.00%	
Wins overall	78.57%		72.86%		74.29%		74.29%	

The mixin heuristics can be thought of constraining or expanding one's view of the board. In other words, the heuristic which says 'stay away from edges' limits some options which may be objectively better. A more abstract property like 'access to porous space' would be useful in calculating endgames, where squares are arbitrarily open, and getting to porous squares would be most useful (this was my intent with the neural network value function). This could be normalized to overall board porosity so that the player would not jump into closed spaces during the beginning of the game.

Intermezzo
The final custom heuristic Maximize score and run away held to the less constraining approach of evaluation. (there are more squares farther away from opponent, than closer to them). The additions of various penalties also did not do much.
The next idea is to give some weights to self and to opponents evaluations. It would make sense that having more moves at greater depth is more important than what opponent can do, as during endgames, the number of moves is going to almost always be one. On this the next few rounds I tested biases of self reward vs opponent reward by giving more weight to one or the other as shown in the table:

This results in a view of the world of value functions that favor a bias toward higher evaluation of number of own moves, and to stay away from opponent.

	Maximize own score +run away		**Maximize own score +run away**		**Maximize own score +run away**		**difference in moves – 2 deep**		**difference in moves – 2 deep**		**difference in moves – 2 deep**	
Bias	**self**	**opp**	**self**	**opp**	**self**	**opp**	**self**	**opp**	**self**	**opp**	**self**	**opp**
	2	**1**	**1**	**1**	**1**	**2**	**2**	**1**	**1**	**1**	**1**	**2**
Random	10	0	10	0	10	0	10	0	10	0	10	0
MM_Open	9	1	9	1	9	1	8	2	9	1	8	2
MM_Center	10	0	10	0	9	1	10	0	9	1	10	0
MM_Improved	9	1	8	2	9	1	10	0	8	2	9	1
AB_open	6	4	5	5	3	7	7	3	6	4	5	5
AB_center	6	4	5	5	4	6	4	6	5	5	3	7
AB_Improved	6	4	5	5	5	5	4	6	5	5	4	6
Wins over bes	60.00%		50.00%		50.00%		40.00%		50.00%		50.00%	
Wins overall	80.00%		74.29%		70.00%		75.71%		74.29%		70.00%	

So I keep the final heuristics as distinct versions or running away with with bias to player's own score vs that of opponent for the purposes of the submission. These are probably within some level of statistical error which would be corrected by more trials.