The added code:

```
def _get_emotion(self, img) -> int:
     model = models.load_model('models\model.keras')

             img_resized = cv2.resize(img, dsize=image_size)
             img_resized = cv2.merge((img_resized, img_resized,
     img_resized))

             input_img = np.expand_dims(img_resized, axis=-1)
             input_img = np.expand_dims(input_img, axis=0)

             prediction = model.predict(input_img)
             emotion = np.argmax(prediction)

             return int(emotion)
```

- The interface worked well, the only thing was that the inputs had to be fairly heavily changed for it to fit the input shape of the model
- It didn't seem to be very effective at recognizing our facial expressions, with 'neutral' registering for 'happy' and vice versa. Trying to get 'surprised' to go through was also difficult.
- One of the reasons it might not have worked so well was because the model had overfitted too much while training. Another thing that may have occurred is the classifications were wrong. It was consistently getting happy and neutral reversed, and that may or may not be because of how we trained our model.

Trace of game:
Player X took position (1, 2). (Bot is X) (Player is O)

```
| | | |
| | |X|
| | | |

|O| | |
| | |X|
| | | |

|O| | |
| | |X|
| | | |

|O| | |
|X| |X|
| | | |

|O| | |
|X|O|X|
| | | |

|O| | |
|X|O|X|
|X| | |

|O| | |
|X|O|X|
|X| |O|
```