Annie McKay
Yazmyn Sims

TicTacToe Report

A trace of moves in that game:
Game 1
Bot went 2,1
I attempted for neutral two times and it worked both times. We move 0,0
Bot moves 2,0
I attempted surprised to stop their win, but got neutral. Second attempt I tried happy and it worked. We move 0,1
Bot moves 2,2. I lose

Game 2
Bot moves 2,0
Surprise worked: 2, Happy interpreted as neutral, but numbers were very close. but the spot is filled. Happy worked the second time. But then the surprise was detected as neutral. We move 1,0
Bot moves 0,0
Both my next faces were misinterpreted. We move 0,2.
Bot moves 1,1
Attempting to move 2,2. But once again it was wrong, I moved 0,1.
Bot took 1,2
I attempted surprise twice and it worked. We move 2,2
Bot moves 2,1
Game ended in a draw.

● **Answers to these questions:**
- **How well did your interface work?** After lots of tweaks the interface worked surprisingly well. It loaded fast, and returned its predicted face almost immediately with decently high accuracy.
- **Did it recognize your facial expressions with the same accuracy as it achieved against the test set?** No, It was a bit lower than the validation and test accuracy. I'd say the TicTacToe game had about a ~40% accuracy, and a ~80% accuracy if you knew the kind of face you needed to make for the program to correctly recognize it.
- **If not, why not?** I believe there's lots of reasons why the model isn't as accurate as we'd want it to be. There were the limited parameters that we had to use, the small size of images we used to train the model, and the already low accuracy had a huge impact on the accuracy of the game itself. Our faces with our expressions were also not trained in the model, so it could've made the model harder to recognize our emotions.

● The lines of code you added in the _get_emotion function to use the model for emotion detection

```
        res = cv2.resize(img, dsize=image_size)
        rgb = cv2.cvtColor(res, cv2.COLOR_GRAY2RGB)
        sampleList = np.array([rgb])
        predictions = self.model.predict(sampleList)
        predictions[0][1] -= .2    #happy is usually higher, so im manually decreasing it so it has to be significantly bigger to be chosen.
        print(predictions)
        prediction = predictions.argmax(axis=1)[0]
        return int(prediction)
```