Rock Paper Scissors

A image classification model created by Michael Shepherd and Connor Rogers

Hypothesis:

Using image classification, it is possible to determine the hand gestures a user is showing to play a game of rock, paper, scissors against a computer.

Rock =

Data Set: Rock Paper Scissors

- 700+ images for each sign.
- Images are of resolution 300 x 200.
- Images were taken with consistent Lighting and Background.
- Dataset taken from a similar project, although our model uses the full resolution images, not ½ scale versions (https://www.kaggle.com/drgfreeman/rockpaperscissors)

Classification Model:

- Keras/TensorFlow Neural Network.
- Uses 2D Convolutions to exaggerate the feature from the background.
- 2D Pooling is used to downsample output for subsequent operations.
- Dropouts provide randomizations, to reduce overfit scenarios.

Model Training:

- Augmentations of each image were created (Flips, Rotations, Etc.) using Keras' ImageDataGenerator.flow_from_directory() to avoid having to manually load and augment each image.
- 80% of data was used for training, 20% was used for validation steps during training.
- Training terminated when model met 98% accuracy during validation to avoid overfitting.

```
generated_data = ImageDataGenerator(rescale=1. / 255, rotation_range=20, vertical_flip=True,
horizontal_flip=True, shear_range=0.2, fill_mode='wrap', validation_split=0.2)

train_data = generated_data.flow_from_directory(base_dir, target_size=image_size,
class_mode='categorical', subset='training', shuffle=True)

val_data = generated_data.flow_from_directory(base_dir, target_size=image_size, class_mode='categorical',
subset='validation', shuffle=True)
```

Deployment: Server

- Deployed as a Flask application running Gunicorn, a WSGI HTTP server for UNIX, hosted by NginX.
- OpenCV for handling input from client.
- Image is processed by Flask server, classified using the pre-trained Keras model, and the result (with a random choice of sign by the CPU) is sent back to the client.

```python
def predict(img_source):
    img = expand_dims(img_source, axis=0)
    classes = model.predict(img, batch_size=10)[0]
    if classes[0] > classes[1] and classes[0] > classes[2]:
        result = "paper"
    elif classes[1] > classes[0] and classes[1] > classes[2]:
        result = "rock"
    elif classes[2] > classes[0] and classes[2] > classes[1]:
        result = "scissors"
    return result

def decide_winner(player_choice):
    cpu_choice = choice(['rock', 'paper', 'scissors'])
    if player_choice == cpu_choice:
        return [player_choice, cpu_choice, "tied"]
    elif player_choice == "scissors" and cpu_choice == "rock":
        return [player_choice, cpu_choice, "lost"]
    elif player_choice == "rock" and cpu_choice == "paper":
        return [player_choice, cpu_choice, "lost"]
    elif player_choice == "paper" and cpu_choice == "scissors":
        return [player_choice, cpu_choice, "lost"]
    else:
        return [player_choice, cpu_choice, "won"]
```

Development: Client

- Live viewfinder created using JavaScript's mediaDevices.getUserMedia() to work with most modern browsers
- Captured image is converted from HTML canvas to dataURL in base64 format, then sent to Flask backend using ajax
- Response is received from server using ajax and HTML is updated using JavaScript's document.getElementById()

Capture Function to get image from user webcam:

```
async function capture() {
      ...
       var canvas = document.getElementById('c');
       canvas.width = video.videoWidth;
       canvas.height = video.videoHeight;
       canvas.getContext('2d').drawImage(video, 0, 0,
video.videoWidth, video.videoHeight);
       var dataURL =
canvas.toDataURL().split(';base64,')[1];
```

Ajax request used to send image to server and receive response:

```
$.ajax({
  type: "POST",
   url: "/get_image",
   data:{imageBase64: dataURL},
   success: function (res) {
document.getElementById('message').innerHTML = res },
   error: function (error) { console.log(error) }
})
```

Conclusion:

- Using Keras, it was possible to create a Neural Network to determine with accuracy what hand sign (rock, paper, or scissors) a user is showing their camera
- Using this and a Flask based web server, a user can play rock, paper, scissors with a computer simply by visiting a webpage
- Deployed as a desktop and mobile-friendly front end

(not working on IOS devices for unknown reasons)

Visit https://www.CodeSmith.link if you would like to play!