Rock-Paper-Scissor Team

Tindanai Wongpipattanopas 59130500210

Role: Create CNN, pre-process image, and design the game

Nawapol Limampai 59130500217

Role: Create a hand dataset, set labels and train the data

Phanupong Boonprasop 59130500223

Role: Get the predict data and Implement the game

Peerapat Triyathavornvong 59130500241

Role: Initialize webcam and do computer vision technique

Varunyou Tarnwuttipong 59130500244

Role: Hand gesture detection technique with opencv and Implement the game

Project Introduction

Our group want to create RPS game. Normally we use people to decide who win but in this project. We use computer to be a committee to decide between two players who is the winner by using AI knowledge in field of machine learning. We designed the game to be in 8-bits bike racing game. Overall system is created by python language.

Project Scope

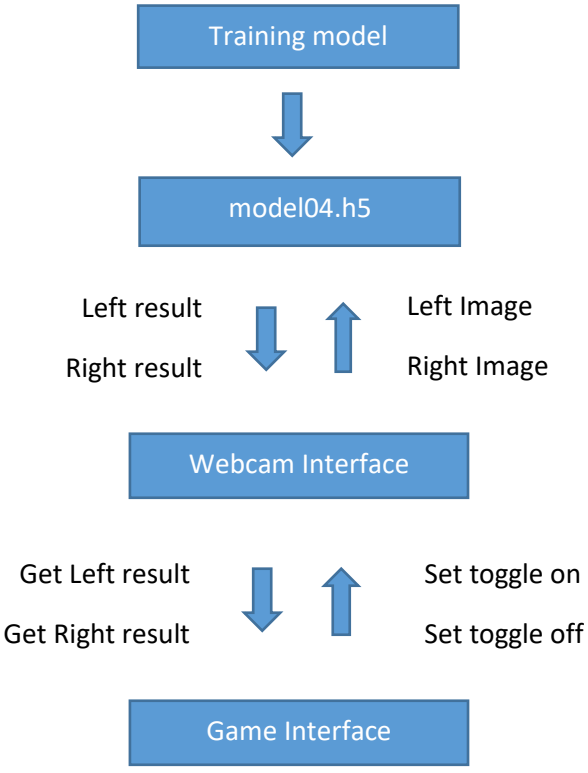
Basic feature

To get the input we use the webcam on laptop to capture the hand gesture which each player will put the hand from left side and right side. Each player must put the hand in the box from their side. Then the computer will show the result of each side from the predict data from the model that we trained. The game will decide which side win by using the rules of the game rock-paper-scissor.

Extra feature (Game)

We create an 8-bit bike racing game style. The winner of each round will move forward, and the loser will stay at the same position. If the player wins 3 time continually, that player will get extra move and show the wheelie. The player who move to the right edge of the game first will be the winner of the game.

Project architecture diagram



Explanation of different component

Image pre-processing

- Data augmentation (with opencv and keras)

  - We capture hand gesture image in many unique ways

  - We resize our image to 28*28 pixels

  - Rotating and flipping image use ImageDataGenerator

- Normalizing (with numpy and keras)

  - We make image to be an array using keras.img_to_array, and then append each image data array to the temp array. We divide 255 to each temp array, and we use that temp array to train

Classification

- Convolutional neural network

  - We use the LeNet CNN architecture for our model. It is small and easy to understand. We set 28x28 input with 3 depths for (RGB).

  - We create 2 sets of layers sets. It contains of convolutional layer which will learn 20 convolution filters, where each filter is of size 5 x 5. Then, we apply the ReLU activation function followed by 2 x 2 max-pooling in both the x and y direction with a stride of 2.

  - Then we define 2 fully connected layers. First layers, we define fully connected contains of 500 units and ReLU activation layer. The second layer, we define the number of dense equal to 3 which is equal to our output. (rps)

GUI system

Webcam interface (using opencv)

- It shows the live video capture frame from the webcam. On the frame, it has the green and red rectangles. The green rectangle is for player from the left, and the red rectangle is for player from the right. When the game countdown finish, this program will start to evaluate the hand. If the hand doesn't show from the both sides, it will show the text on the GUI to inform you to put the hand in the box. If the hand doesn't show from the left sides, it will inform you. For the right side is the same. Input from webcam (images) → model prediction → Result → Game.

Game interface (using pygame and keras)

- Game is created by python using pygame library. It shows the 8-bits bike racing game. The bike will be moved by the result of the predict data that get from the input of the webcam.

Reference

Keras documentation: https://keras.io/

Lenet architecture: https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/

Pygame: https://www.pygame.org/docs/

Python keras tutorial: https://www.youtube.com/watch?v=RznKVRTFkBY&list=PLZbbT5o_s2xrwRnXk_yCPtnqqo4_u2YGL

Image classification with Keras: https://www.pyimagesearch.com/2017/12/11/image-classification-with-keras-and-deep-learning/

OpenCV: https://docs.opencv.org/master/d9/df8/tutorial_root.html

Anaconda: https://conda.io/docs/user-guide/cheatsheet.html