Data Science: NLP (2021) 11:25
https://www.societyofai.in/

# Rock Paper Scissor Game with a machine learning algorithm in python using OpenCV

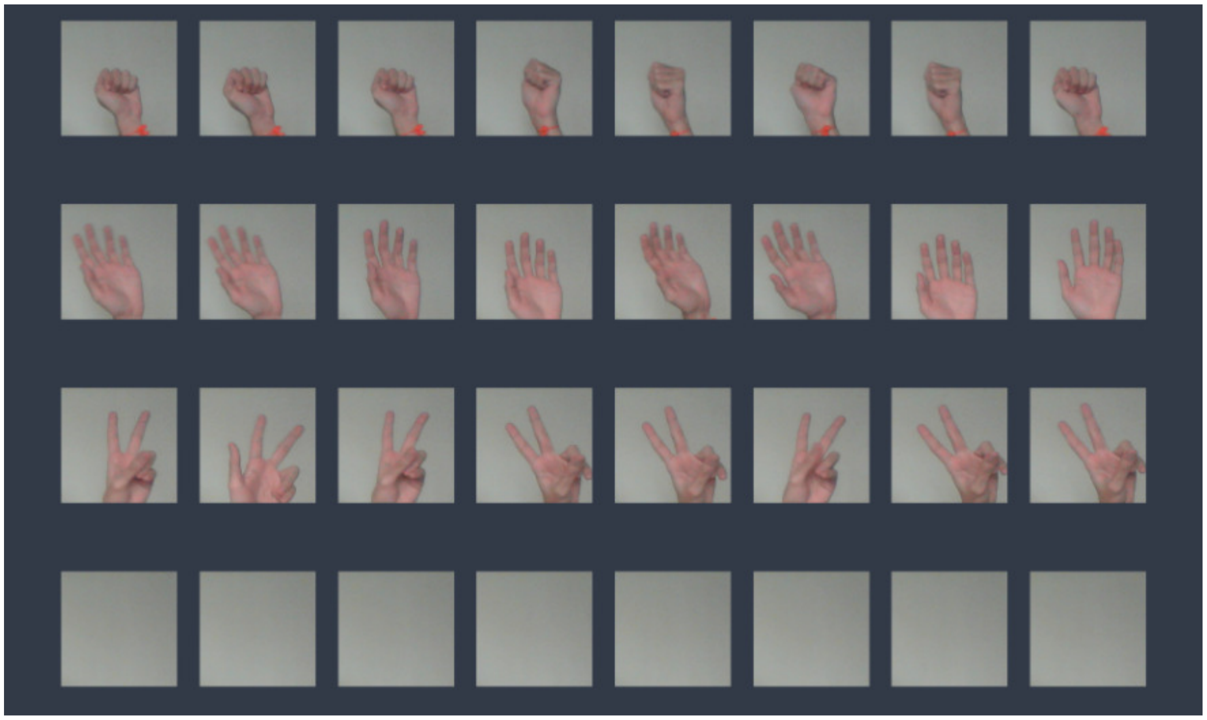
## 1 Introduction

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. This project Rock, paper scissors is based on random moves so I can't model a winning AI system as there is no strategy involved. The only intelligence our system would have been in visual recognition of my hand signs. We have fine-tuned NASNETMobile model to recognize my hand signs when it's inside the box, so when the model predicts my hand signs, the AI randomly generates its own move. And then the winner of that move is decided. The application is designed in a way that you can decide whether to go for the best of 5, best of 10, or any number for that number.

**Project Structure:**
Here's a breakdown of our application in steps.
Step 1: Gather Data, for rock, paper scissor classes.
Step 2: (Optional) Visualize the Data.
Step 3: Preprocess Data and Split it.
Step 4: Prepare Our Model for Transfer Learning.
Step 5: Train Our Model.
Step 6: Check our Accuracy, Loss graphs & save the model.
Step 7: Test on Live Webcam Feed.
Step 8: Create the Final Application.

## 2 Related Work

We have gathered the data from webcam using python code. Our dataset is look like as below



## 3 Technical Approach

We had used the TensorFlow as a backend to train deep learning model for Image Classification to identify whether it's a Rock, Paper or Scissor game. We had used "Keras" as a backend API. We had collected 100 frames each for Rock, Paper and Scissors using "OpenCV" package enabling the Video capturing with a bounding box. If anyone press the "q" key then video capturing frames process would be terminated. If anyone press the "s" key that means he is ordering to capture the "scissors" images, if press the "p" key that means ordering to capture the "paper" images, if press the "r" key that means is ordering to capture the "rock" images and if press the "n" key that means ordering to capture the "background" images.

We had used the "OneHotEncoding" to label the corresponding images. We had used 75% data for the training and 25% data for testing.

We had the 4 labels and images against each label captured is 100. So, total labels would be 400 and 400 images.

```
# Now the convert the integer labels into one hot format. i.e. 0 = [1,0,0,0]  etc.
one_hot_labels = to_categorical(Int_labels, 4)

# Now we're splitting the data, 75% for training and 25% for testing.
(trainX, testX, trainY, testY) = train_test_split(images, one_hot_labels, test_size=0.25, random_state=50)

# Empty memory from RAM
images = []


# This can further free up memory from RAM but be careful, if you won't be able to change split % after this.
# rock, paper, scissor = [], [], []
```

```
Total images: 400 , Total Labels: 400
```

Image size used is 224. We had Loaded a pre-trained NASNETMobile Model without the head by doing include top = False because we are using a "Network Transfer" model.

```
# Check the number of layers in the final Model
print ("Number of Layers in Model: {}".format(len(model.layers[:])))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/nasnet/NASNet-mobile-no-top.h5
19996672/19993432 [==============================] - 9s 0us/step
Number of Layers in Model: 773
```

We had also done the data augmentation to add more data into our dataset.

```
# Adding transformations that I know would help, you can feel free to add more.
# I'm doing horizontal_flip = False, in case you aren't sure which hand you would be using you can make that True.

augment = ImageDataGenerator(
    |
        rotation_range=30,
        zoom_range=0.25,
        width_shift_range=0.10,
        height_shift_range=0.10,
        shear_range=0.10,
        horizontal_flip=False,
        fill_mode="nearest"
)
```

We had used the epochs as 15 and batch-size as 20 to train the model.

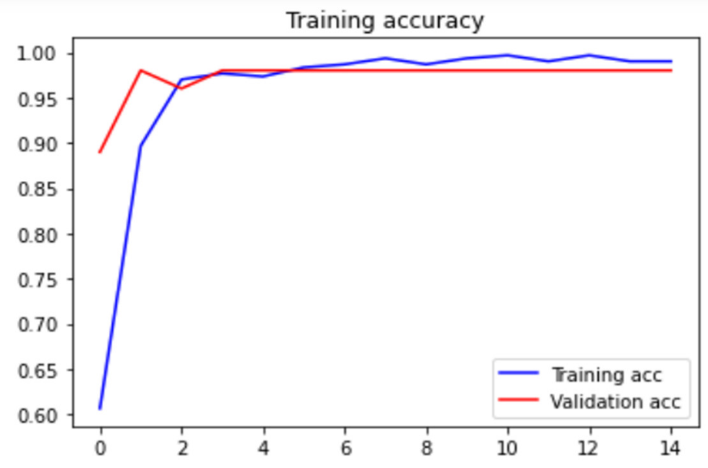
```
# Set batchsize according to your system
epochs = 15
batchsize = 20

# Start training
history = model.fit(x=augment.flow(trainX, trainY, batch_size=batchsize), validation_data=(testX, testY),
steps_per_epoch= len(trainX) // batchsize, epochs=epochs)

# Use model.fit_generator function instead if TF version < 2.2
#history = model.fit_generator(x = augment.flow(trainX, trainY, batch_size=batchsize), validation_data=(testX, testY),
#steps_per_epoch= len(trainX) // batchsize, epochs=epochs)
```

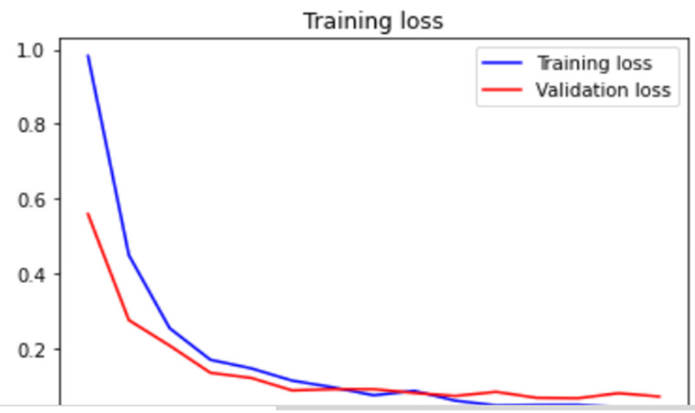
```
Epoch 1/15
15/15 [==============================] - 38s 3s/step - loss: 0.9826 - accuracy: 0.6067 - val_loss: 0.5586 - val_accuracy: 0.890
0
Epoch 2/15
15/15 [==============================] - 32s 2s/step - loss: 0.4492 - accuracy: 0.8967 - val_loss: 0.2753 - val_accuracy: 0.980
0
Epoch 3/15
15/15 [==============================] - 32s 2s/step - loss: 0.2537 - accuracy: 0.9700 - val_loss: 0.2072 - val_accuracy: 0.960
0
Epoch 4/15
15/15 [==============================] - 32s 2s/step - loss: 0.1692 - accuracy: 0.9767 - val_loss: 0.1344 - val_accuracy: 0.980
0
Epoch 5/15
15/15 [==============================] - 32s 2s/step - loss: 0.1463 - accuracy: 0.9733 - val_loss: 0.1207 - val_accuracy: 0.980
0
Epoch 6/15
15/15 [==============================] - 32s 2s/step - loss: 0.1136 - accuracy: 0.9833 - val_loss: 0.0874 - val_accuracy: 0.980
0
```

## Result

Model Accuracy: - We have achieved a Validation Accuracy of 98% and loss 0.087.

We get these plots for Training Loss and Training Accuracy:



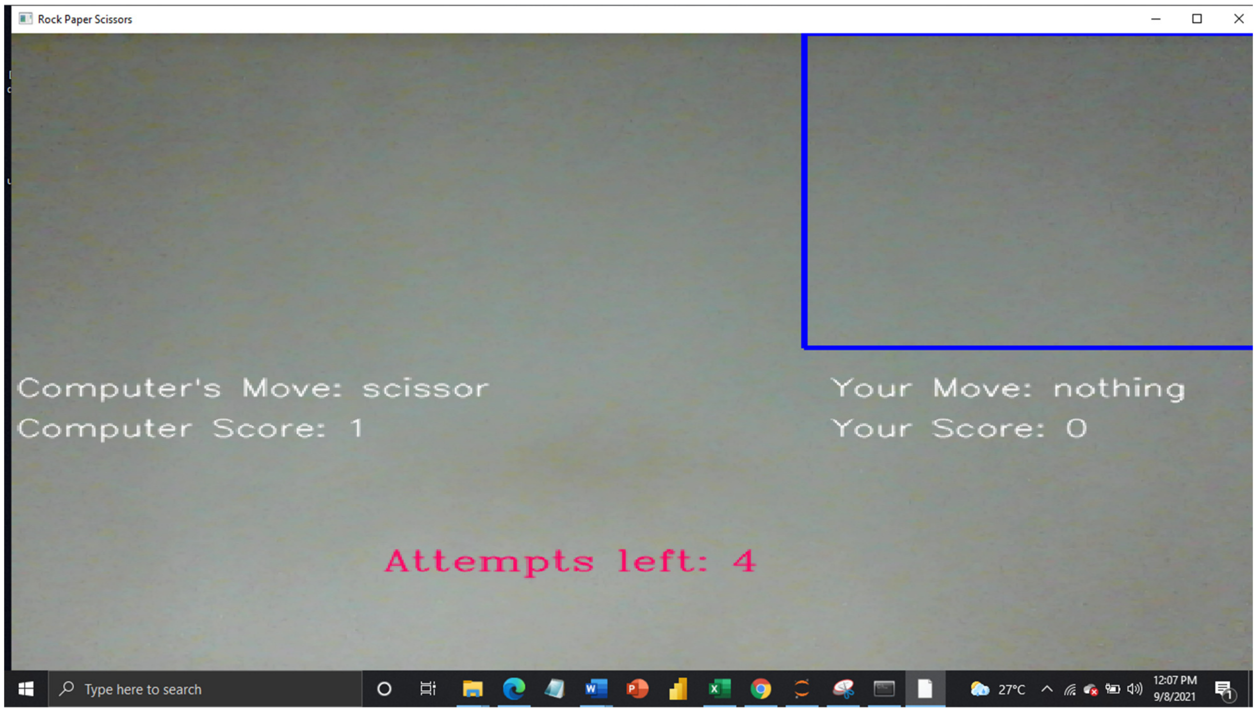


## 4 Model Output

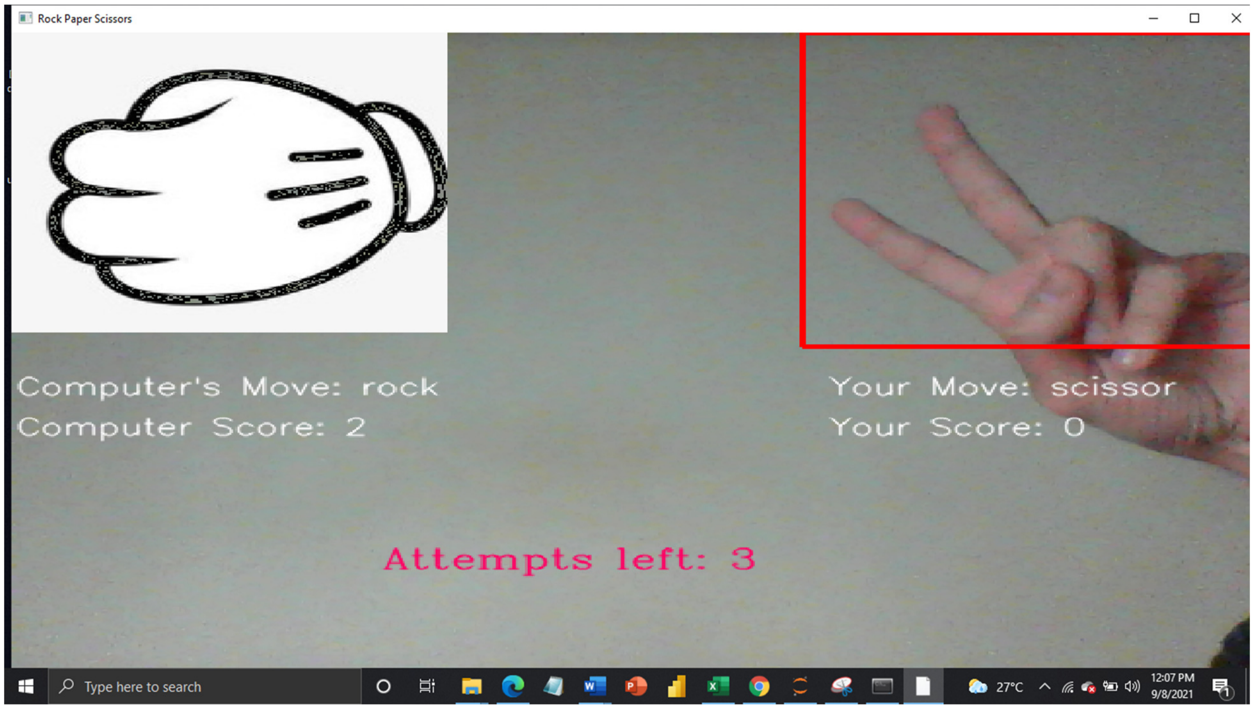
How to access the game
step 1. Run play.py script on cmd using below command

```
$ python play.py
```

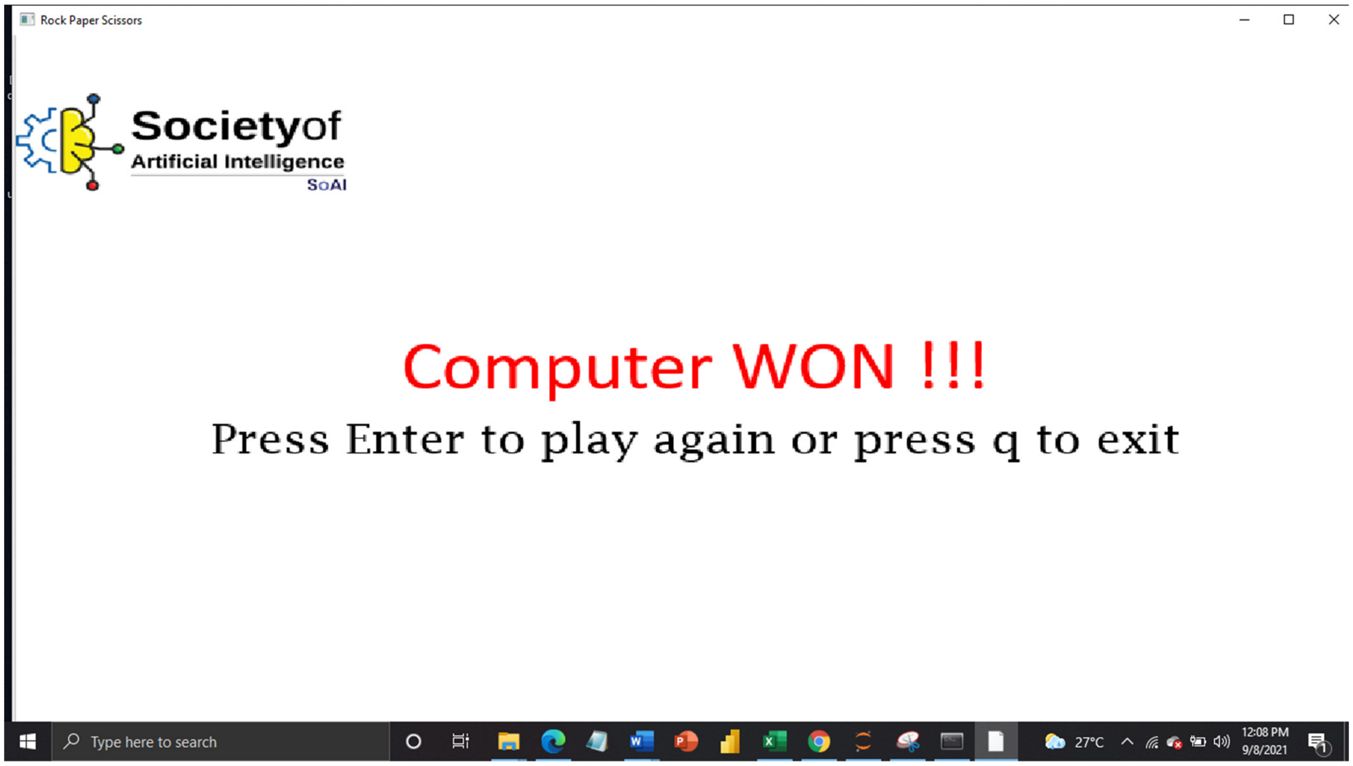
step 2. kindly wait for load the model (make sure you should have TensorFlow 2.2, OpenCV 4x, and scikit-learn 0.23x installed in your system.)
step 6. Enjoy the game!!!!



```
Computer's Move: scissor          Your Move: nothing
Computer Score: 1                 Your Score: 0

              Attempts left: 4
```



```
Computer's Move: rock             Your Move: scissor
Computer Score: 2                 Your Score: 0

              Attempts left: 3
```


Rock Paper Scissors

Computer WON !!!

Press Enter to play again or press q to exit

## 5 Created By [CV Team]

- *Deepika Goel*
- *Viknesh C.*
- *Kiran Rajput*
- *Ashish Sunny*

## 6 References

- https://www.tensorflow.org/
- https://docs.opencv.org/4.5.2/
- https://towardsdatascience.com/how-to-win-over-70-matches-in-rock-paper-scissors-3e17e67e0dab