Project Name: Human Sitting Pose Recognition

Subject Name: Deep Learning and Neural Networks
Subject Code: 15CSE380

Project by:

Hitesh J: CSE18042

Kaazima Ifrah: CSE18073

Nuthi Sriram: CSE18083

Potti Priya: CSE18091

Project guide: Dr. Suja.P

Abstract

In recent years, due to the advent of personal computing and consequently its rapid spread into consumer lives, more and more people spend time on computers. The increasing hours spent in front of computers leads to many developing bad sitting postures. This bad sitting posture leads to many serious negative consequences on one's health. Therefore, it is more important, now more than ever, to create a solution that prevents bad computer posture.

Humans are creatures of habit. On average, it takes many years to develop lasting habits. Therefore, it is virtually impossible to simply stop one that has been recurring for many years. Due to their habit of bad posture, it is nearly impossible for them to stop. In the case of back posture, the issue is not simply a mental one. People who have bad posture for long periods of time have physical changes in their muscles and joints which prevent their bodies from simply straightening out. Therefore, as the user cannot stop bad posture alone, it's important to create alternative solutions.

The output can tell you whether a person is sitting in a Straight position, Hunchback position (leaning forward), Reclined position (leaning backward) and if the person is Folding Hands and Folding Legs (Kneeling Position).

We implemented using CNN

A Convolutional Neural Network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.

Implementation of CNN in our Project

```
def relu(x): return Activation('relu')(x)


def conv(x, nf, ks, name, weight_decay):
kernel_reg = l2(weight_decay[0]) if weight_decay else None
bias_reg = l2(weight_decay[1]) if weight_decay else None


x = Conv2D(nf, (ks, ks), padding='same', name=name,
kernel_regularizer=kernel_reg,
bias_regularizer=bias_reg,
kernel_initializer=random_normal(stddev=0.01),
bias_initializer=constant(0.0))(x)
return x


def pooling(x, ks, st, name):
x = MaxPooling2D((ks, ks), strides=(st, st), name=name)(x)
return x


def vgg_block(x, weight_decay):
# Block 1
x = conv(x, 64, 3, "conv1_1", (weight_decay, 0))
x = relu(x)
x = conv(x, 64, 3, "conv1_2", (weight_decay, 0))
x = relu(x)
x = pooling(x, 2, 2, "pool1_1")


# Block 2
x = conv(x, 128, 3, "conv2_1", (weight_decay, 0))
```

```
x = relu(x)
x = conv(x, 128, 3, "conv2_2", (weight_decay, 0))
x = relu(x)
x = pooling(x, 2, 2, "pool2_1")

# Block 3
x = conv(x, 256, 3, "conv3_1", (weight_decay, 0))
x = relu(x)
x = conv(x, 256, 3, "conv3_2", (weight_decay, 0))
x = relu(x)
x = conv(x, 256, 3, "conv3_3", (weight_decay, 0))
x = relu(x)
x = conv(x, 256, 3, "conv3_4", (weight_decay, 0))
x = relu(x)
x = pooling(x, 2, 2, "pool3_1")

# Block 4
x = conv(x, 512, 3, "conv4_1", (weight_decay, 0))
x = relu(x)
x = conv(x, 512, 3, "conv4_2", (weight_decay, 0))
x = relu(x)

# Additional non vgg layers
x = conv(x, 256, 3, "conv4_3_CPM", (weight_decay, 0))
x = relu(x)
x = conv(x, 128, 3, "conv4_4_CPM", (weight_decay, 0))
x = relu(x)
return x

def stage1_block(x, num_p, branch, weight_decay):
```

```
# Block 1
x = conv(x, 128, 3, "Mconv1_stage1_L%d" % branch, (weight_decay, 0))
x = relu(x)
x = conv(x, 128, 3, "Mconv2_stage1_L%d" % branch, (weight_decay, 0))
x = relu(x)
x = conv(x, 128, 3, "Mconv3_stage1_L%d" % branch, (weight_decay, 0))
x = relu(x)
x = conv(x, 512, 1, "Mconv4_stage1_L%d" % branch, (weight_decay, 0))
x = relu(x)
x = conv(x, num_p, 1, "Mconv5_stage1_L%d" % branch, (weight_decay, 0))
return x


def stageT_block(x, num_p, stage, branch, weight_decay):
# Block 1
x = conv(x, 128, 7, "Mconv1_stage%d_L%d" % (stage, branch), (weight_decay, 0))
x = relu(x)
x = conv(x, 128, 7, "Mconv2_stage%d_L%d" % (stage, branch), (weight_decay, 0))
x = relu(x)
x = conv(x, 128, 7, "Mconv3_stage%d_L%d" % (stage, branch), (weight_decay, 0))
x = relu(x)
x = conv(x, 128, 7, "Mconv4_stage%d_L%d" % (stage, branch), (weight_decay, 0))
x = relu(x)
x = conv(x, 128, 7, "Mconv5_stage%d_L%d" % (stage, branch), (weight_decay, 0))
x = relu(x)
x = conv(x, 128, 1, "Mconv6_stage%d_L%d" % (stage, branch), (weight_decay, 0))
x = relu(x)
x = conv(x, num_p, 1, "Mconv7_stage%d_L%d" % (stage, branch), (weight_decay, 0))
return x


def apply_mask(x, mask1, mask2, num_p, stage, branch):
```

```
w_name = "weight_stage%d_L%d" % (stage, branch)

if num_p == 38:
w = Multiply(name=w_name)([x, mask1]) # vec_weight

else:
w = Multiply(name=w_name)([x, mask2]) # vec_heat
return w
```

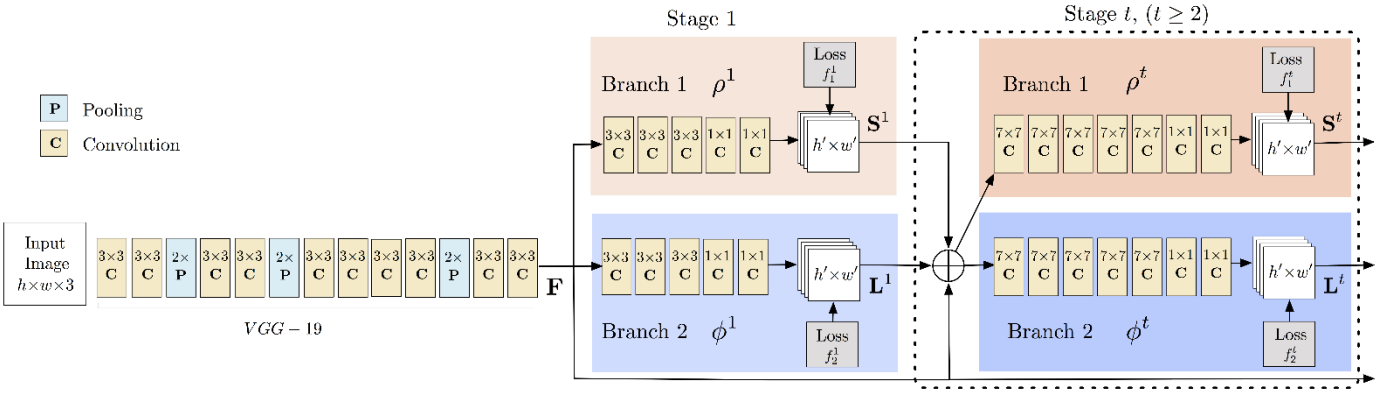
Tools & Python Libraries

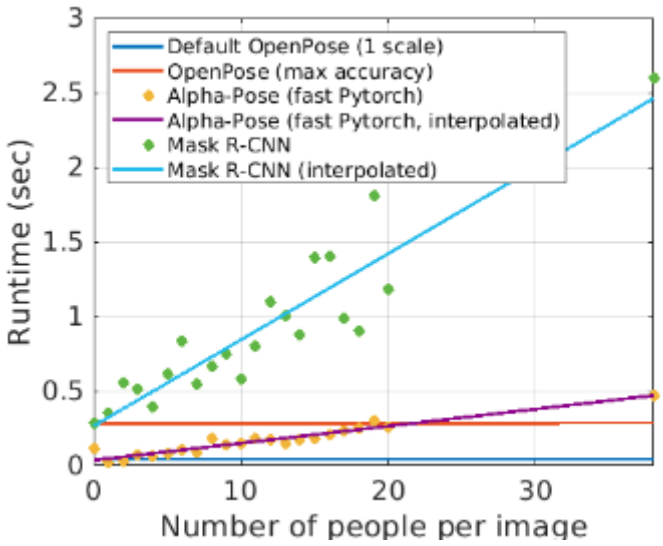
- Keras
- Open-pose
- Numpy
- Keras-Applications
- Keras-Preprocessing
- Scipy
- opencv-contrib-python
- opencv-python
- Tensorflow

Data set

```
COCO API - Dataset
    ├── datasets
        └── coco_2017_dataset
            ├── annotations
            │   ├── person_keypoints_train2017.json
            │   └── person_keypoints_val2017.json
        │   ├── train2017/*
        │   └── val2017/*
```

Model Architecture





We are making use of OpenPose since that is the best among all the available posture recognition libraries

An inference time comparison between the 3 available pose estimation libraries (same hardware and conditions): OpenPose, Alpha-Pose (fast Pytorch version), and Mask R-CNN. The OpenPose runtime is constant, while the runtime of Alpha-Pose and Mask R-CNN grow linearly with the number of people.

Description of modules used

Posture_image (the main module): This module is a static module. It consists of the process function which is used to find the key points of full body using Open Pose.

Posture_realtime (the main module): This module is a dynamic module which gives the results in realtime by analying the lateral view of the person through webcam.

Process(): This module finds the key points of full body using OpenPose.

calcAngle(): This module calculates the angle between two points with respect to x-axis (horizontal axis).

calcDistance(): This module calculates the distance between two points.

checkPosition(): This module checks the position of spine (Straight position, Hunchback position (leaning forward), Reclined position (leaning backward)).

checkHandFold(): This module checks if hands are folding or not.

calcKneeling(): This module checks the position of legs, whether kneeling or not.

Problems faced

Problem 1: Not detecting hunch back unless it's a very huge hunch

Solution: Rewrote the function to calculate the slope from left ear to left hip or else right ear to left hip

Initially: Slope from neck to the hip was being calculated

Problem 2: Not detecting kneeling unless the knee is completely bent.

Solution: Angle 60 trial and error value We can tweak this accordingly and results will vary.

Result

This is a software that can output the sitting position of a person when the lateral view (side view) of the person is given as input. The output can tell you whether a person is sitting in a Straight position, Hunchback position (leaning forward), Reclined position (leaning backward) and if the person is Folding Hands and Folding Legs (Kneeling Position). The OpenPose library can detect the key points of the human body. These key points co-ordinates can be used to estimate the sitting posture of the person.

Contribution of each team member

Hitesh J : Integrating the model with the application and reading the input image values.

Kaazima Ifrah: Process module - finding the keypoints of the human body using various math functions, python notebook for analysis, calcangle function.

Potti Priya : Process module - finding the keypoints of the human body using various math functions, checkHandFold funciton.

Sriram Nuthi : Process module - calcDistance function, checkKneeling function, posture_realtime module.

Scope of future work

Any future work should try to fix the following issues:

- Single Person: This software can detect multiple persons keypoints but can only detect the posture of a single person. This can be extended by simply iterating the detection part of the code over all the set of keypoints that are detected for each person.
- Sitting-Postures: Extending the project to work with different postures not just sitting postures, even sleeping postures.
- Limbs detection: All the limbs of the body have to be seen for the algorithm to work perfectly
- Real time image recognition: recognizing the person
- Notifying the person: The detected person needs to be notified if he/she has a bad posture

Whether you will extend the project

Yes, deploying the application on realtime cctv cameras in workplaces and detecting multiple people at a time and sending each individual user a personalised notification message through from the ip cameras directly to their working systems.