

1) Connecting to drive

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

2) Get the model:

```
from keras.models import load_model
```

```
model = load_model('/content/drive/Shared drives/Projet-ML-M2-PLS/models/position_model')
```

```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode
import numpy as np
import uuid
```

```
def take_photo(quality=0.8):
```

```
    js = Javascript('''
        async function takePhoto(quality) {
            var finish = false;
            const div = document.createElement('div');
            const capture = document.createElement('button');
            capture.textContent = 'Reconnaître la position';
            const stop = document.createElement('button');
            stop.textContent = 'Stop';
            stop.onclick = function() {finish = true; capture.click()};
            div.appendChild(capture);
            div.appendChild(stop);

            const video = document.createElement('video');
            video.style.display = 'block';
            const stream = await navigator.mediaDevices.getUserMedia({video: true});

            document.body.appendChild(div);
            div.appendChild(video);
            video.srcObject = stream;
            await video.play();
```

```
            // Resize the output to fit the video element.
            google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);
```

```
            // Wait for Capture to be clicked.
            await new Promise((resolve) => capture.onclick = resolve);
            if(finish){
```

```

        stream.getTracks().forEach(function(track) {
            track.stop();
        });
        div.innerHTML = ""
        return null
    }

    const canvas = document.createElement('canvas');
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    canvas.getContext('2d').drawImage(video, 0, 0);
    stream.getVideoTracks()[0].stop();
    div.remove();
    return {data: canvas.toDataURL('image/jpeg', quality)};
}
'''
display(js)

photo = eval_js('takePhoto({})'.format(quality))
if photo is None:
    return None
data = photo['data']
binary = b64decode(data.split(',')[1])
img_array = np.frombuffer(binary, dtype=np.uint8)
img_array = np.expand_dims(img_array, axis=0)

return img_array

```

3) Start the webcam and recognize position:

```

def recognize():
    img = take_photo()
    if img is None:
        return None
    pred = model.predict(img)
    position = ''
    if pred[0][0]>0.5:
        position='absent'
        return 'Position détecté: ' + position + ' avec une certitude de: ' + pred[0][0]
    elif pred[0][1]>0.5:
        position='bas'
        return 'Position détecté: ' + position + ' avec une certitude de: ' + pred[0][1]
    elif pred[0][2]>0.5:
        position='droit'
        return 'Position détecté: ' + position + ' avec une certitude de: ' + pred[0][2]
    elif pred[0][3]>0.5:
        position='face'
        return 'Position détecté: ' + position + ' avec une certitude de: ' + pred[0][3]
    elif pred[0][4]>0.5:
        position='gauche'

```

```

    return 'Position détecté: ' + position + ' avec une certitude de: ' + pred[0][4]
elif pred[0][5]>0.5:
    position='haut'
    return 'Position détecté: ' + position + ' avec une certitude de: ' + pred[0][5]
else return 'Aucune position detecté'

```

```

from IPython.display import Image

```

```

try:
    while True:
        result = recognize()
        if result is None:
            break
        print(result)

    # Show the image which was just taken.
    #display(Image(filename))
    print('Camera stopped')
except Exception as err:
    # Errors will be thrown if the user does not have a webcam or if they do not
    # grant the page permission to access it.
    print(str(err))

```

```

☞ Position détecté: face avec une certitude de: 0.55
Position détecté: droit avec une certitude de: 0.61
Position détecté: gauche avec une certitude de: 0.64
Position détecté: haut avec une certitude de: 0.7
Position détecté: bas avec une certitude de: 0.51
Position détecté: absent avec une certitude de: 0.8
Camera stopped

```

