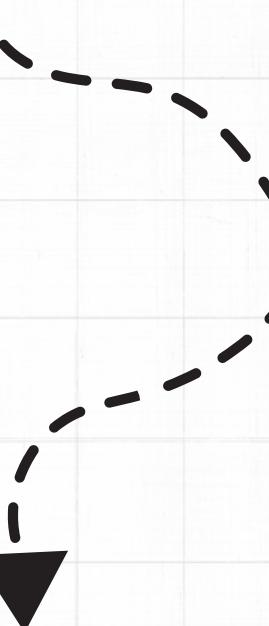


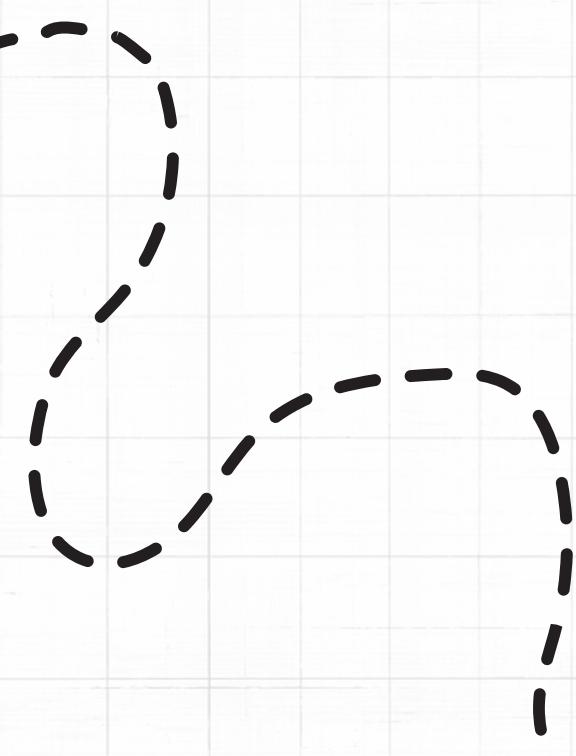
카메라 기반 거북목 방지 앱

모바일 컴퓨팅과 응용 기말 발표

2016-19965 김성은



Problem & Goal

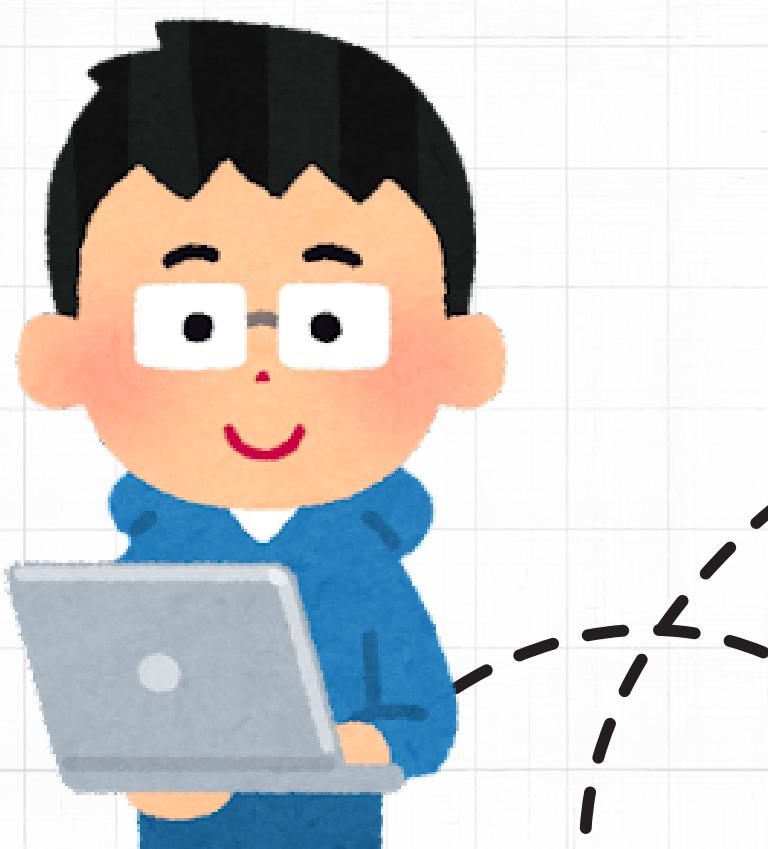


문제 상황

- 한국인 기준으로 현대인이 앓아서 지내는 시간은 하루 평균 7시간
- 프로그래머 등 컴퓨터를 다루는 직장은 그보다 더 오랜 시간을 앓아서 보내게 됨
- 코로나 장기화 이후 거북목 증후군의 발병자는 더더욱 증가
- 한번 발병 시 도수치료 및 교정 등에 큰 재정적 부담이 발생

목표

- 평소 올바른 자세를 상시 유지해야 함
- 그러나 의식적으로 습관을 고치기는 쉽지 않다
- AI를 활용한 앱으로 올바른 자세를 유지하게 하자!
- 켜두면 공부/작업 등 오랫동안 앉아 있는 도중 자세가 흐트러질 때마다 알림을 주는 앱을 구상



Approach

사용 모델

BlazePose

- 3D Keypoint 출력
- MoveNet에 비해 다소 모델 사이즈가 큼

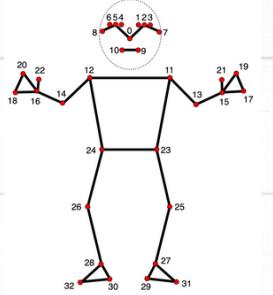


BlazePose

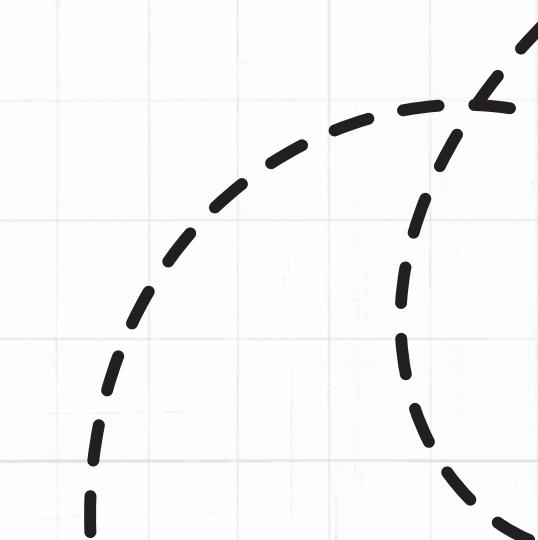
numerical calculation

Negative

Dense Layers



Binary Output



Rule-based Gate

- 수식으로 구성된, 후술할 모델에 inference를 할 것인지 판별하는 단계
- 간단한 계산으로 자세가 크게 벗어나지 않았다면 추가 inference 단계를 건너뛴다.

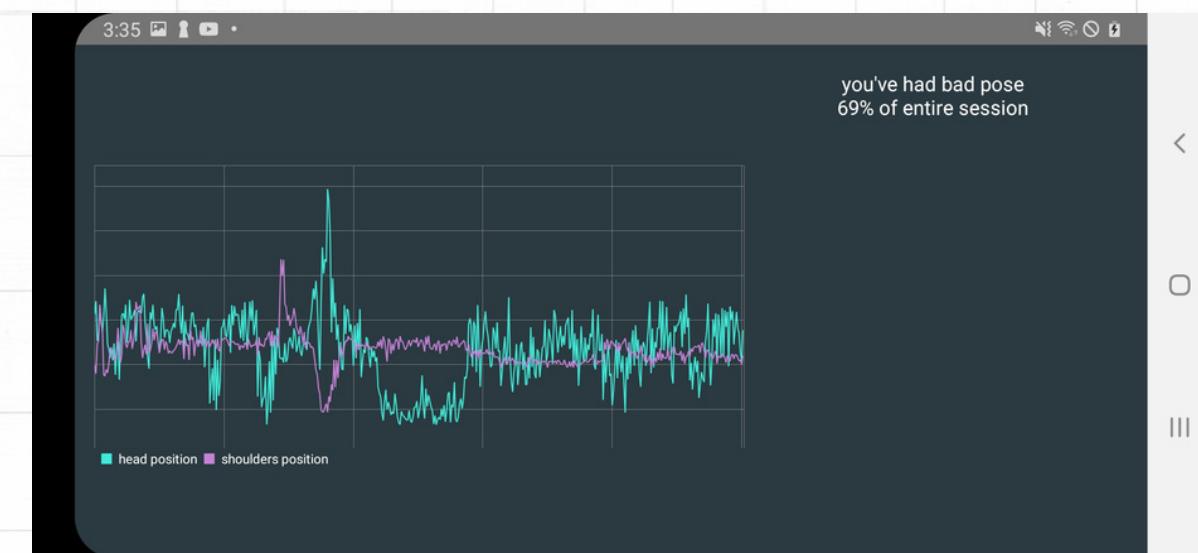
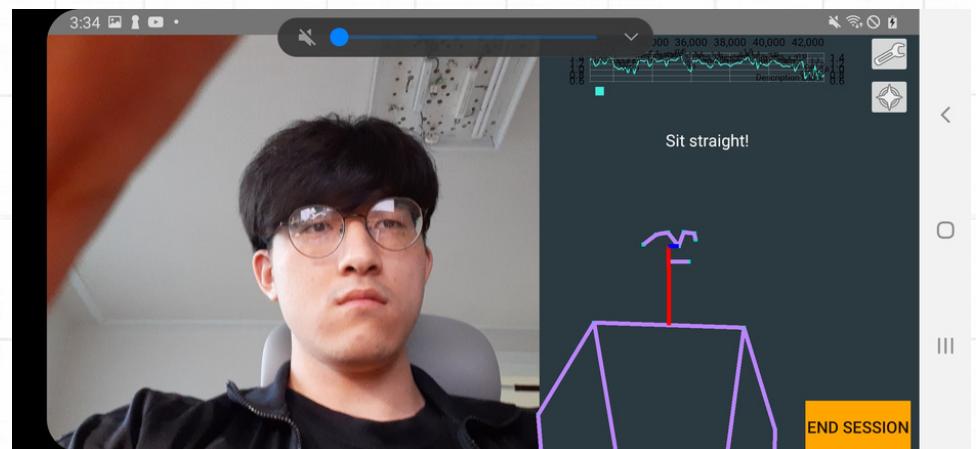
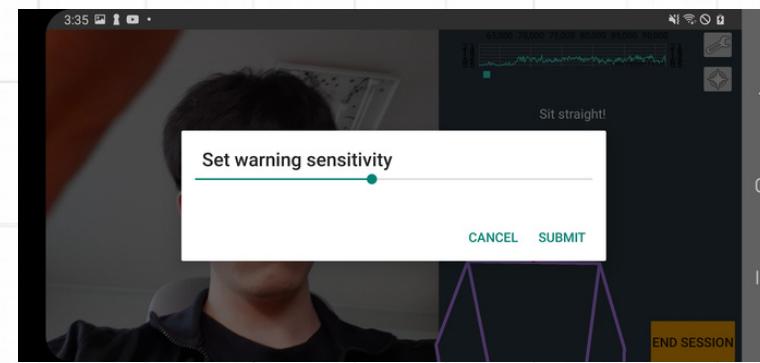
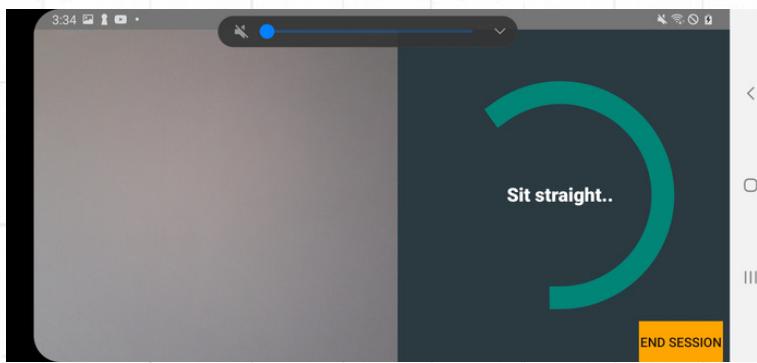
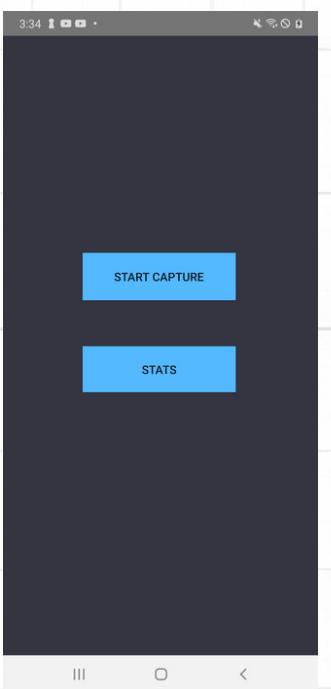
Custom DNN

- BlazePose를 거친 Keypoint를 입력값으로 받는 DNN
- 각 좌표 값을 입력으로 받는, Fully-Connected layer들로 구성된 모델
- 거북목 상태인지의 confidence를 출력

User Flow

구현한 기능

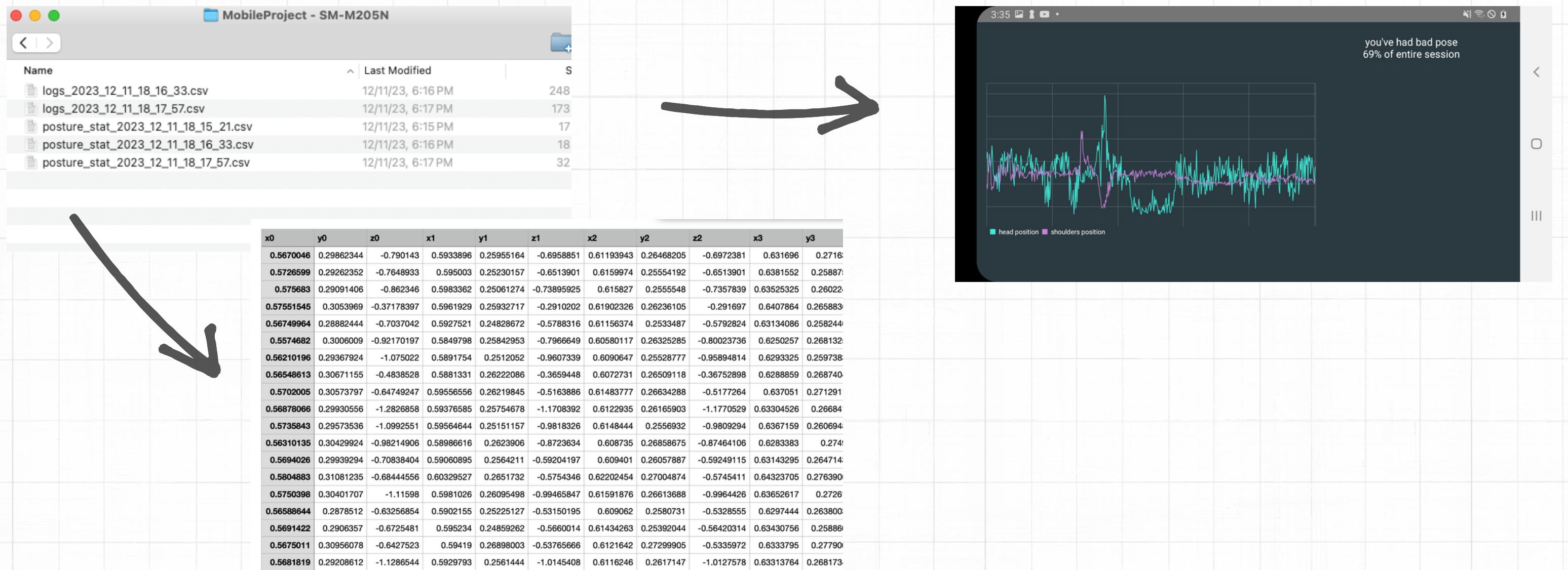
- 정확한 분석을 위한 Calibration 기능
- 앱의 동작 중, DNN을 사용해 유저의 자세를 분석
- 자세가 올바르지 못할 때, 알림을 준다
- 장시간 사용 후 통계 제공



User Flow

추가 기능

- 장시간 사용 후 통계 제공
- 전체 키포인트 로깅 & 저장



Model Output

- 33개의 3d coordinate를 입력으로 받는 모델
- Dense layer들로 구성됨
- 테스트 데이터셋에서 96.8% accuracy 확인
- Quantization 이후 또한 96.8%

```
[12] 1 test_loss, test_accuracy = model.evaluate(X_test, y_test, batch_size=5)
2 print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
25/25 [=====] - 0s 2ms/step - loss: 0.1441 - accuracy: 0.9680
Test Accuracy: 96.80%
```

```
1 model.summary()
2
Model: "posture_detection_model"
+-----+-----+-----+
| Layer (type) | Output Shape | Param # |
+-----+-----+-----+
| dense (Dense) | multiple | 6400 |
| dropout (Dropout) | multiple | 0 |
| dense_1 (Dense) | multiple | 2080 |
| dropout_1 (Dropout) | multiple | 0 |
| dense_2 (Dense) | multiple | 1056 |
| dropout_2 (Dropout) | multiple | 0 |
| dense_3 (Dense) | multiple | 33 |
+-----+-----+-----+
Total params: 9569 (37.38 KB)
Trainable params: 9569 (37.38 KB)
Non-trainable params: 0 (0.00 Byte)
```

```
1 interpreter = tf.lite.Interpreter(model_path=file_path)
2 interpreter.allocate_tensors()
3
4 input_details = interpreter.get_input_details()
5 output_details = interpreter.get_output_details()
6
7 correct_predictions = 0
8 total_predictions = 0
9
10 test_dataset = tf.data.Dataset.from_tensor_slices((X_test, y_test))
11
12 for x, y in test_dataset:
13     x = tf.cast(x, dtype=tf.float32)
14     x = tf.expand_dims(x, axis=0)
15     interpreter.set_tensor(input_details[0]['index'], x)
16
17     interpreter.invoke()
18
19     y_pred = interpreter.get_tensor(output_details[0]['index'])
20
21     if round(y_pred[0][0]) == y.numpy():
22         correct_predictions += 1
23     total_predictions += 1
24
25 accuracy = correct_predictions / total_predictions
26 print(f'Quantized Model Accuracy: {accuracy * 100:.2f}%')
```

Quantized Model Accuracy: 96.80%

Demo

- <https://youtube.com/shorts/JUqvkOH0a80?feature=share>



Thank You