영유아 위험 행동 감지 시스템

2조

김윤정, 원하은, 정가희, 정준하

Intro

AI-KU!

아이쿠!

잠시라도 눈을 떼면 금세 사고치는 우리 아이,
눈이 열 개라도 모자란 당신을 위해
딥 러닝 기반 영유아 위험 행동 감지 시스템
'아이쿠'가 힘이 되어드리겠습니다.

Contents

01

I. 연구방향 설정

1. 연구 배경
2. 연구의 필요성
3. 프로젝트 목표
4. 일정 및 프로세스

02

II. 시스템 설계

1. 주요 기능 및 시스템
2. 데이터 수집
3. 데이터 전처리
4. 모델 구축 및 모델링
5. 학습 결과

03

III. 시스템 구현

1. UI 소개
2. 시연 영상

04

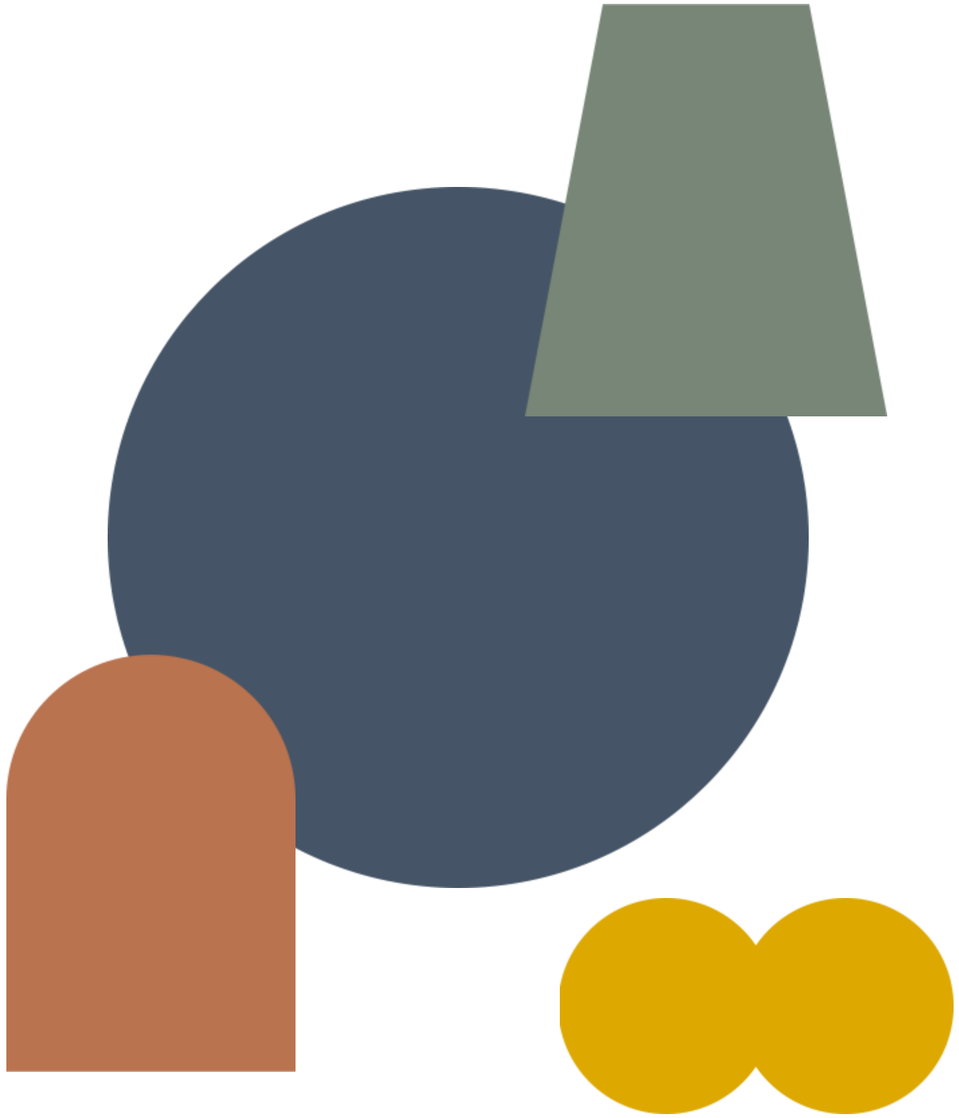
IV. 차별성 및 효과

1. 기존 서비스와의 차별점
2. 서비스 활용 방안
3. 기대 효과
4. 개선 방안

01

I. 연구방향 설정

1. 연구 배경
2. 연구의 필요성
3. 프로젝트 목표
4. 일정 및 프로세스



1-1 연구 배경

- 주변에 **초보 엄마**가 된 지인들의 고민 상담에서부터 시작
- 다양한 걱정거리가 있지만 특히 집에서 일어나는 **크고 작은 안전 사고**가 가장 걱정된다 함

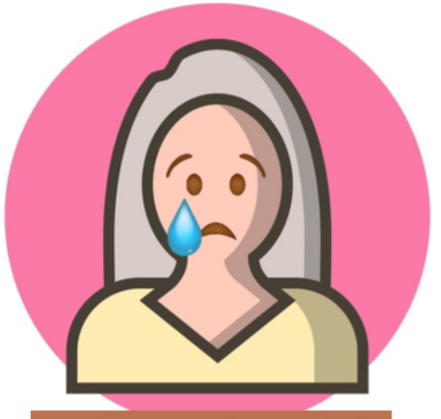
27세 A모씨(육아 1년차)

잠깐이라도 눈을 떼고 있을 수가 없어요😭

아이가 조용할 때 오히려 더 불안해요

재택 근무와 육아를 병행하는 것이 정말 힘들어요😟

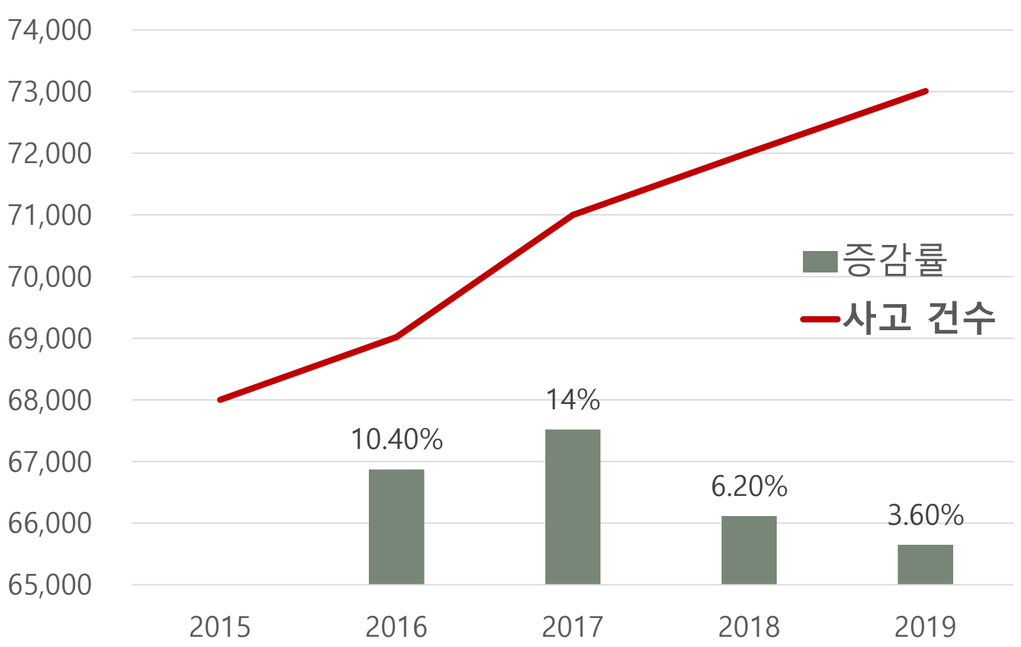
일, 집안일, 육아까지 쉬운 일이 하나 없어요



30세 B모씨(육아 2년차)

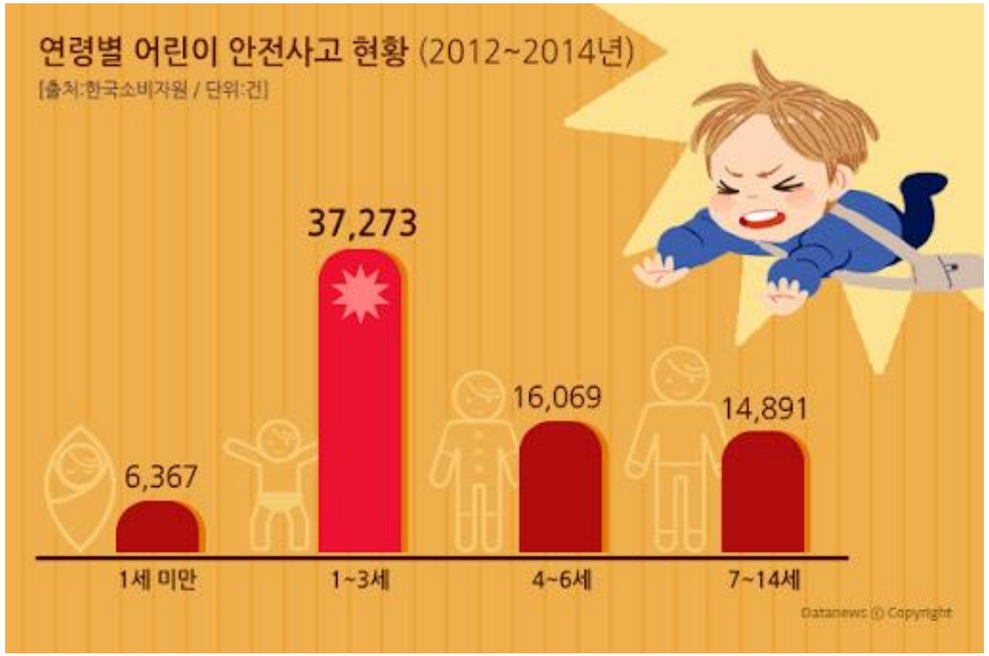
1-1 연구 배경

<어린이 안전사고 현황>



출처: 한국소비자원, 2019

- 사고 건수는 **2015년을 기점으로 매년 증가**
- 전체 사고 중 **영유아 안전사고 비율은 매우 높은 비중을 차지**한다.



1-1 연구 배경

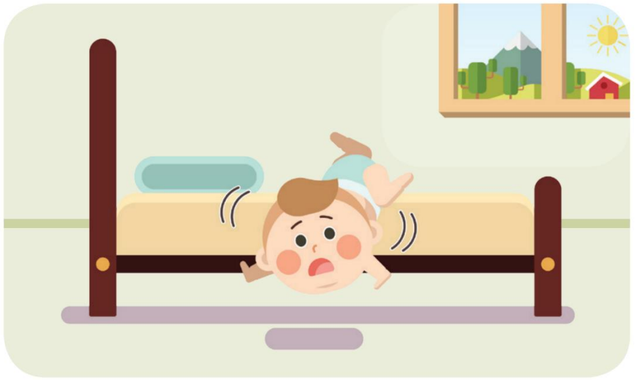
홈 > 경제

부모 부주의에...인덕션에 화상 입고 아령에 발 찌이는 아이들

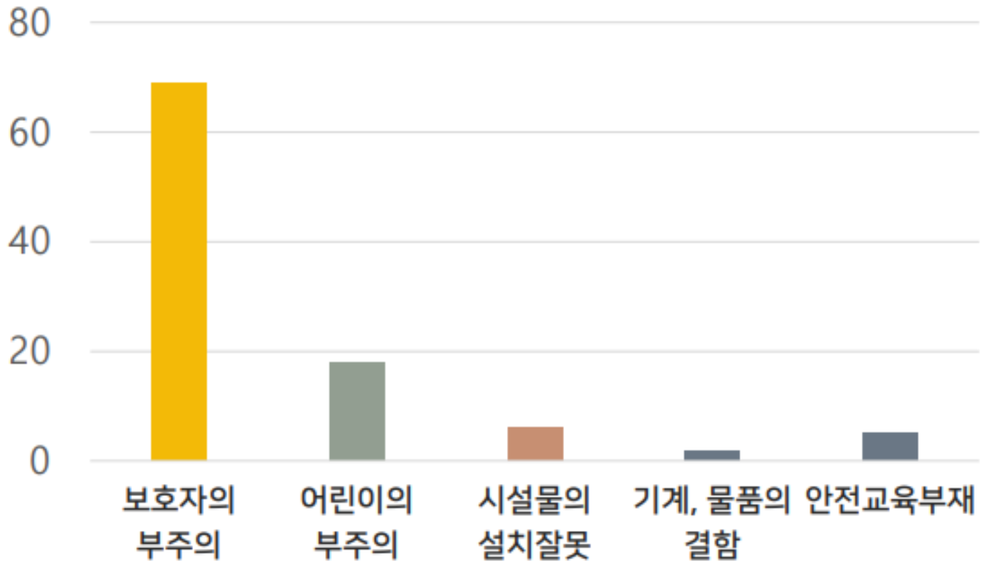
지난 4년간 소아낙상 291건..침상난간 36.1%로 최다

엄태선 기자 | 승인 2020.06.11 06:33 | 댓글 0

낙상 환경적 요인..보호자 부재 25.7%



< 어린이 안전 사고 주요 원인 >



출처: 한국소비자원, 2019

- 어린이 안전사고 약 70%가 **보호자의 실수 및 부주의**로 인해 발생

1-2 연구의 필요성

- **보호자 부주의**

  보호자의 부주의로 인한 사고 사례들이 꾸준히 보도 되고 있다.

- **보호자의 책임?**

24시간 아이들에게만 신경 쓸 수 없는 보호자. 사고의 책임은 모두 보호자에게 돌아갈 수 밖에 없다.

< 영유아용 홈카메라 등장 >

- 현재 시중에 판매되고 있는 홈카메라는 소리 센서, 동작 센서를 통해 실시간으로 **영유아의 행동을 모니터링하는 기능만 제공**

1-2 연구의 필요성

< 영유아용 홈카메라 등장 >

- **보호자 부주의**

보호자의 부주의로 인한 사고 사례들이 꾸준히 보고되고 있다.

영유아의 위험 행동을 즉각적으로 알려주는 시스템 부재

- **보호자의 책임?**

24시간 아이들에게만 신경 쓸 수 없는 보호자. 사고의 책임은 모두 보호자에게 돌아갈 수 밖에 없다.

- 현재 시중에 판매되고 있는 홈카메라는 소리 센서, 동작 센서를 통해 실시간으로 **영유아의 행동을 모니터링하는 기능만 제공**

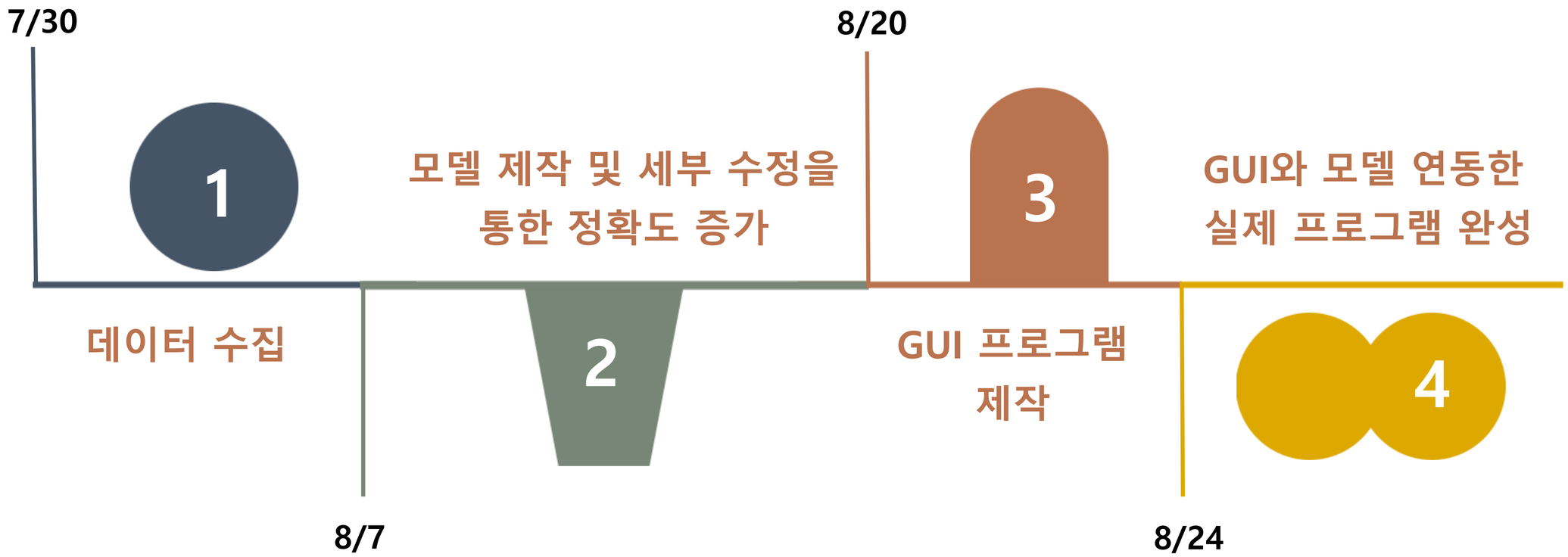
1-3 프로젝트 목표

영유아 위험 행동을 학습한 모델 제작.

실시간으로 관측되는 영상의 프레임과 학습된 모델을 연동시켜 아이의 행동을 실시간 판단.

위험 행동 관측 시 아이나 부모에게 경고메세지를 알려 위험 행동 제지.

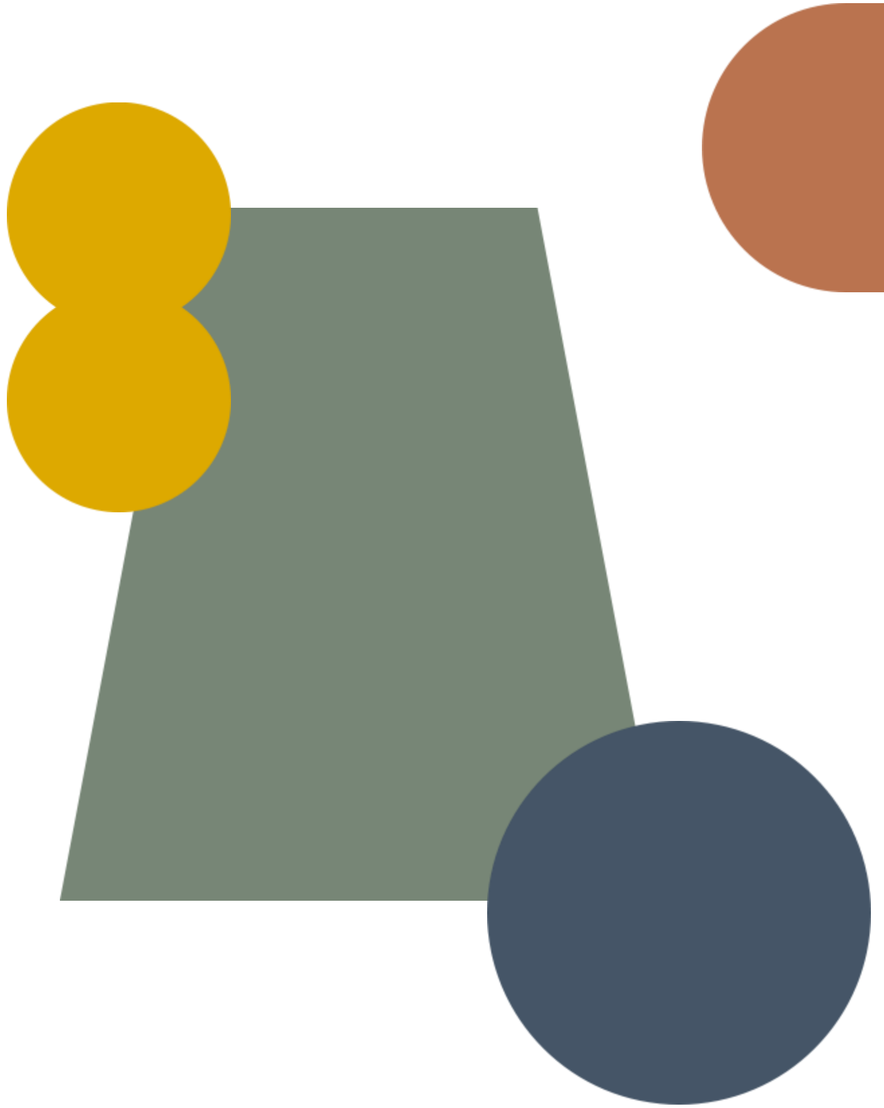
1-4 일정 및 프로세스


7/30
1
데이터 수집
8/7
2
모델 제작 및 세부 수정을
통한 정확도 증가
8/20
3
GUI 프로그램
제작
8/24
4
GUI와 모델 연동한
실제 프로그램 완성

02

II. 시스템 설계

1. 주요 기능 및 시스템
2. 데이터 수집
3. 데이터 전처리
4. 모델 구축 및 모델링
5. 학습 결과



2-1 주요 기능 및 시스템

1 카메라를 통해 아이들의 행동을 실시간으로 관찰 한다.

2 경고 행동 시 카메라를 통해 경고 메시지를 보낸다.

3 위험 행동 시 보호자의 핸드폰으로 알림이 간다.

2-1 주요 기능 및 시스템

안전 행동

- 걸음
- 앉아 있음
- 음식 섭취
- 그 외

경고 행동

- 위험 물건 잡기

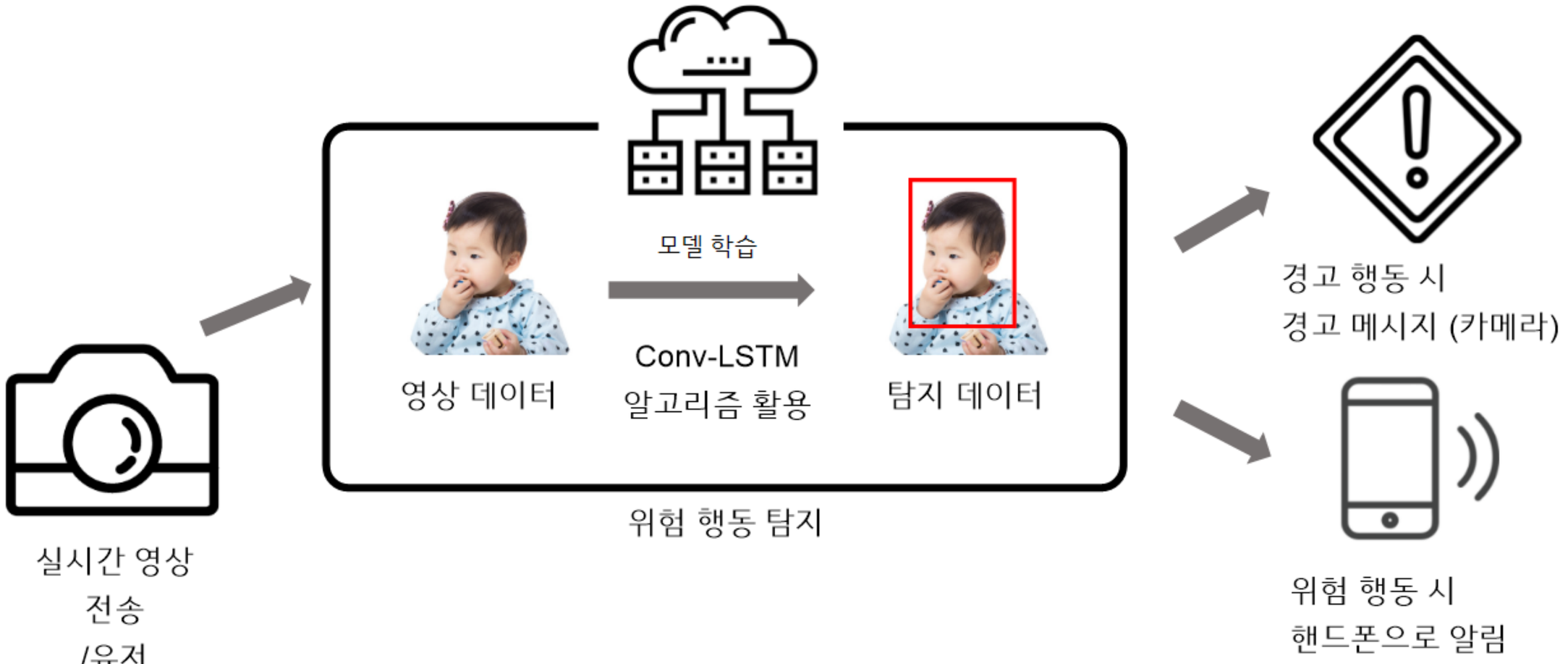
녹음된 경고 메시지가 아이에게 재생된다.

위험 행동

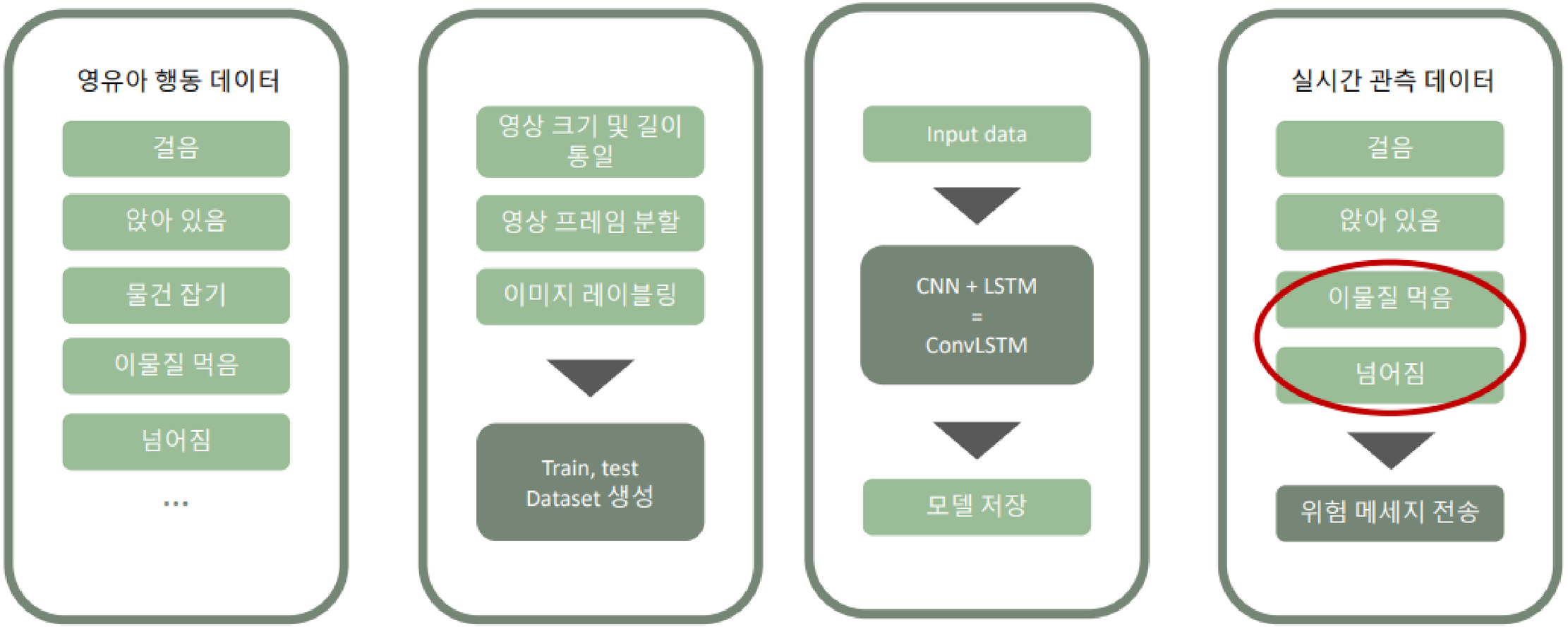
- 이물질 입에 댐
- 넘어짐, 낙상
- 높은 곳에 기어 올라감

보호자에게 문자 메시지가 전송된다.

2-1 주요 기능 및 시스템


실시간 영상
전송
/유저
영상 데이터
모델 학습
Conv-LSTM
알고리즘 활용
탐지 데이터
위험 행동 탐지
경고 행동 시
경고 메시지 (카메라)
위험 행동 시
핸드폰으로 알림

2-1 주요 기능 및 시스템


영유아 행동 데이터
걸음
앉아 있음
물건 잡기
이물질 먹음
넘어짐
...
영상 크기 및 길이 통일
영상 프레임 분할
이미지 레이블링
Train, test Dataset 생성
Input data
CNN + LSTM = ConvLSTM
모델 저장
실시간 관측 데이터
걸음
앉아 있음
이물질 먹음
넘어짐
위험 메세지 전송

2-2 데이터 수집

아이 행동 영상 수집

- **개방 데이터** : 없음
- **직접 수집** : 유튜브 영상
  - 유튜브 등의 웹에서 행동 영상 자료 직접 수집
  - 단계별 행동 영상 데이터
    (3초 내외 분량의 동영상 2000개)

2-3 데이터 전처리

①
```
data_dir = "/content/drive/MyDrive/video_data"
img_height , img_width = 64, 64
seq_len = 16
```

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

```
Mounted at /content/drive
```

②
```
classes = ["fall_down", "grab", "swallow", "walk","sit","climb",'eat','safe']
```

③
```
def frames_extraction(video_path):
    frames_list = []

    vidcap = cv2.VideoCapture(video_path)
    total_frames = vidcap.get(cv2.CAP_PROP_FRAME_COUNT)
    #total frame 갯수를 16으로 나눠줌으로써 총 영상의 프레임을 16등분하는 프레임들만 추출하게 함
    frames_step = total_frames//seq_len

    for j in range(seq_len):
        #parameter 1 즉 frame number을 frame에 설정해준다. (j*frames_step)
        vidcap.set(1,j*frames_step)
        success,image = vidcap.read()
        #이미지를 frames_list에 담는다.
        if success:
            image = cv2.resize(image, (img_height, img_width))
            frames_list.append(image)
        else:
            print("Defected frame {}".format(video_path))
            break

    return frames_list
```

① 이미지 크기와 프레임 개수 맞춤

② 8개의 클래스 설정

③ 프레임 추출 코드 : 한 영상의 전체 프레임을 16등분하는 프레임을 추출

2-3 데이터 전처리

④

```
import vidaug.augmentors as va

def create_data(input_dir):
    X = []
    Y = []

    sometimes = lambda aug: va.Sometimes(1, aug) # video augmentation

    seq_1 = va.Sequential([
        sometimes(va.HorizontalFlip()), #좌우 반전
    ])
    seq_2 = va.Sequential([
        sometimes(va.RandomShear(0.2,0.2)), #영상 기울이기
    ])

    classes_list = os.listdir(input_dir)

    for c in classes_list:
        print(c)
        files_list = os.listdir(os.path.join(input_dir, c))
        for f in files_list:
            frames = frames_extraction(os.path.join(os.path.join(input_dir, c), f))
            if len(frames) == seq_len:
                #frame 추출된 기본 영상
                X.append(frames)
                #추출된 frame을 이용하여 3가지 augmentation - 좌우반전, 기울이기, 좌우반전 + 기울이기
                X.append(seq_1(frames))
                X.append(seq_2(frames))
                X.append(seq_2(seq_1(frames)))
                y = [0]*len(classes)
                y[classes.index(c)] = 1
                #이에 따른 라벨값도 4번 추가.
                Y.append(y)
                Y.append(y)
                Y.append(y)
                Y.append(y)


    X = np.asarray(X)
    Y = np.asarray(Y)
    return X, Y
```

⑤

⑥

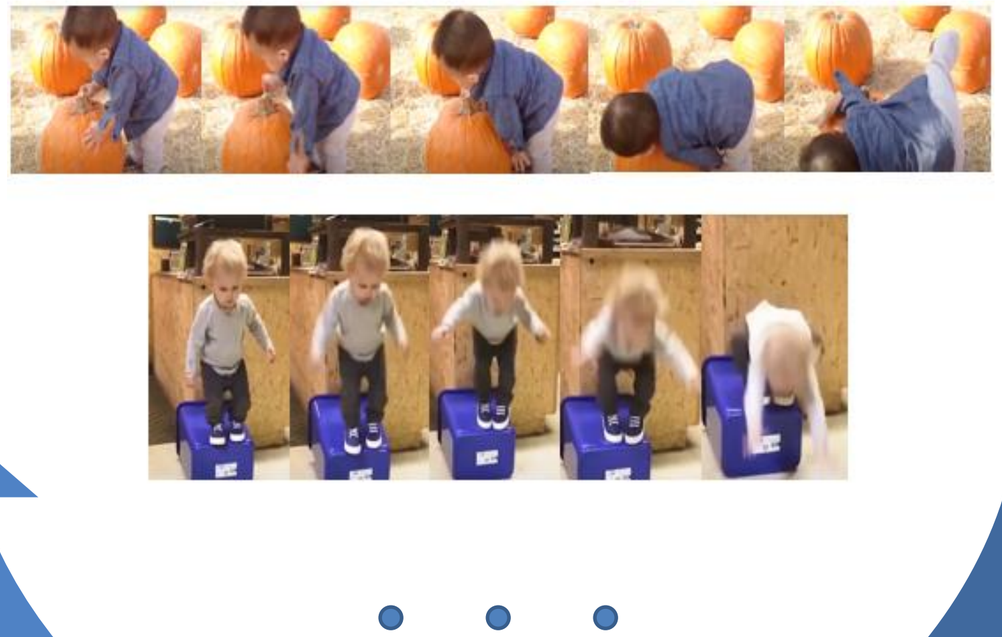
```
X, Y = create_data(data_dir)
X = X/255
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, shuffle=True, random_state=0)
```

④ 프레임 추출과 이미지 레이블링 진행 코드

⑤ 데이터를 보충하기 위해 video augmentation 진행

⑥ 코드를 바탕으로 train, test set 분리

2-4 모델 구축 및 모델링



- 일정 행동 구간 or 일정 시간 동안의 **연속된 행동 영상**을 **이미지 레이블링**
- 전처리한 데이터를 **Conv-LSTM 알고리즘**을 사용하여 **행동 분류 모델 생성**

2-4 모델 구축 및 모델링

```
model = Sequential()

model.add(ConvLSTM2D(filters = 32, kernel_size = (3, 3),strides = (2,2), return_sequences = True, input_shape = (seq_len, img_height, img_width, 3)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Activation('tanh'))
model.add(ConvLSTM2D(filters = 32, kernel_size = (3, 3),strides = (2,2), return_sequences = True))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Activation('tanh'))
model.add(ConvLSTM2D(filters = 64, kernel_size = (3, 3),strides = (2,2), return_sequences = True))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Activation('tanh'))
model.add(ConvLSTM2D(filters = 128, kernel_size = (3, 3),strides = (2,2), return_sequences = False))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Activation('tanh'))


model.add(Flatten())
model.add(Dense(512, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(8, activation = "softmax"))

model.summary()
```

- **4개의 레이어**를 쌓아서 모델 설계. Filter 구성은 VGG16 논문 참조

- Activation function은 **tanh** 이용. 또한 overfitting을 막기 위해 **dropout**과 **batch normalization** 이용.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_lst_m2d (ConvLSTM2D)    (None, 16, 31, 31, 32)    40448
_________________________________________________________________
batch_normalization (BatchNo (None, 16, 31, 31, 32)    128
_________________________________________________________________
activation (Activation)      (None, 16, 31, 31, 32)    0
_________________________________________________________________
conv_lst_m2d_1 (ConvLSTM2D)  (None, 16, 15, 15, 32)    73856
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 15, 15, 32)    128
_________________________________________________________________
activation_1 (Activation)    (None, 16, 15, 15, 32)    0
_________________________________________________________________
conv_lst_m2d_2 (ConvLSTM2D)  (None, 16, 7, 7, 64)      221440
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 7, 7, 64)      256
_________________________________________________________________
activation_2 (Activation)    (None, 16, 7, 7, 64)      0
_________________________________________________________________
conv_lst_m2d_3 (ConvLSTM2D)  (None, 3, 3, 128)         885248
_________________________________________________________________
batch_normalization_3 (Batch (None, 3, 3, 128)         512
_________________________________________________________________
activation_3 (Activation)    (None, 3, 3, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 512)               590336
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 4104
=================================================================
Total params: 1,816,456
Trainable params: 1,815,944
Non-trainable params: 512
_________________________________________________________________
```

2-5 학습 결과

```
def schedular(epoch):
  if epoch < 10:
    return 0.001
  else:
    return 0.001 * math.exp(0.1*(10-epoch))
```

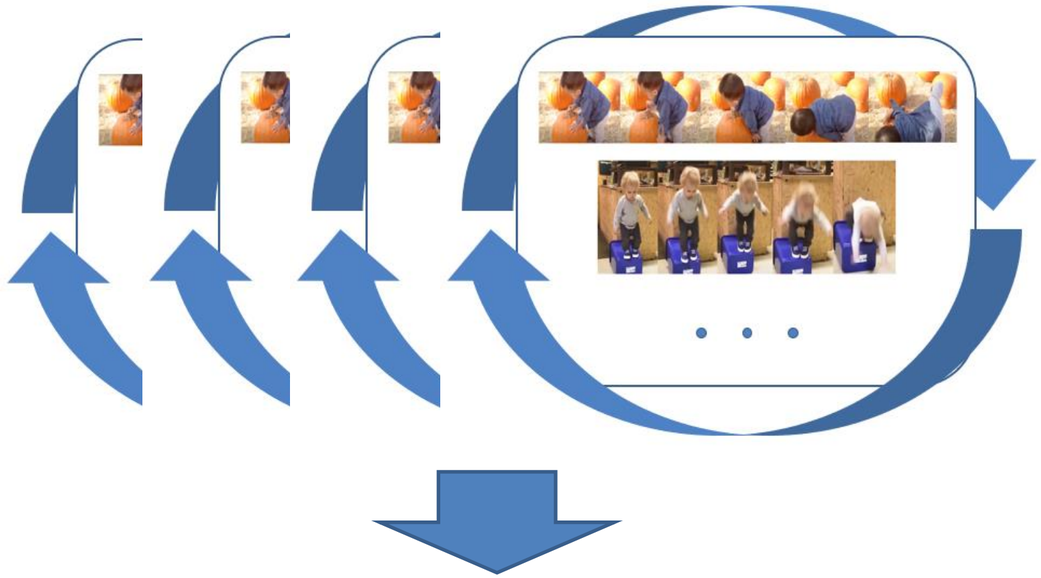
```
opt = tf.keras.optimizers.Adam(lr=0.0005)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=["accuracy"])

# earlystop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)
callbacks = tf.keras.callbacks.LearningRateScheduler(schedular)

history = model.fit(x = X_train, y = y_train, epochs=25, batch_size = 128, shuffle=True, validation_split=0.2, callbacks=callbacks)

import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val','accuracy','val_accuracy'])
plt.show()
```

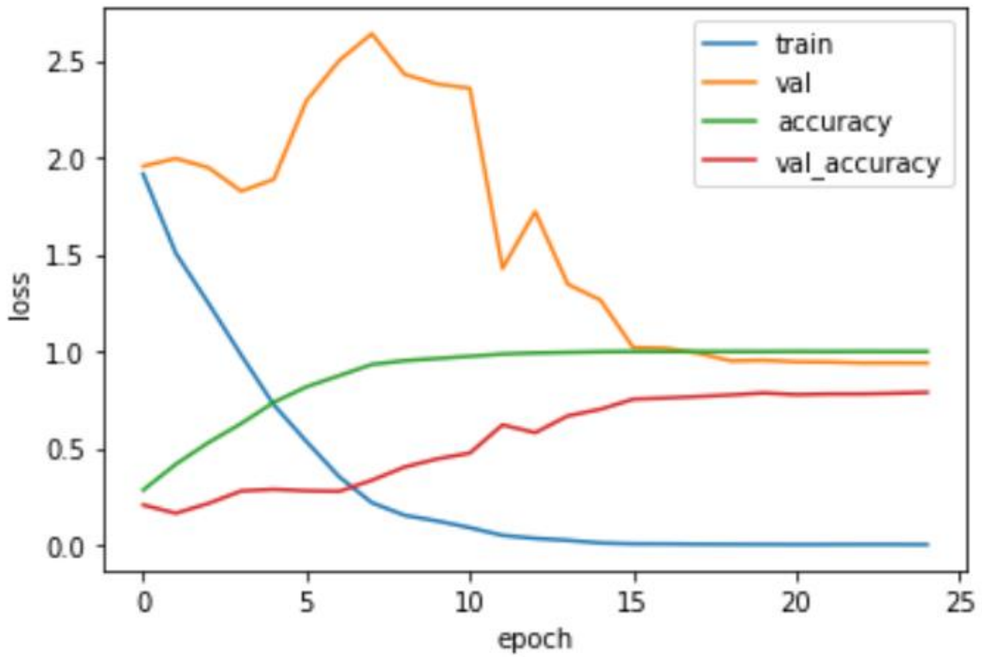
- 모델 학습을 진행 – 여러 번의 학습 중 가장 학습 성능이 좋았던 **learning rate schedular** 이용



4aug_epoch25_withsafe.h5

- **25번**의 epoch를 진행 후 모델 저장

2-5 학습 결과



- 학습 결과, validation loss는 0.9392, validation accuracy는 0.7879로 **79%**의 정확도를 보임

	precision	recall	f1-score	support
0	0.79	0.76	0.78	277
1	0.81	0.76	0.78	219
2	0.76	0.84	0.80	283
3	0.77	0.75	0.76	232
4	0.85	0.86	0.86	188
5	0.70	0.77	0.74	146
6	0.80	0.64	0.71	81
accuracy			0.78	1426
macro avg	0.78	0.77	0.78	1426
weighted avg	0.78	0.78	0.78	1426

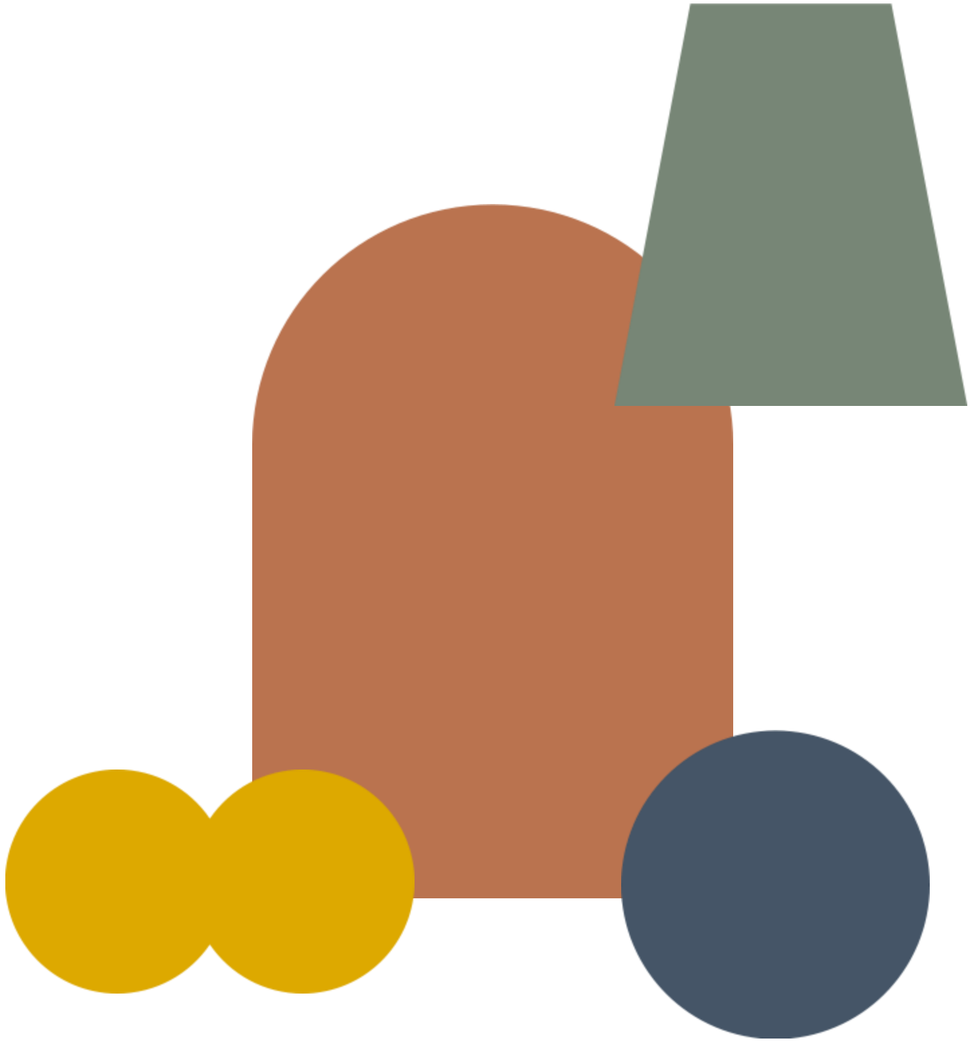
accuracy score : 0.782608695652174

- Test data에 대해 최종 정확도 **78.2%** 보임.

03

III. 시스템 구현

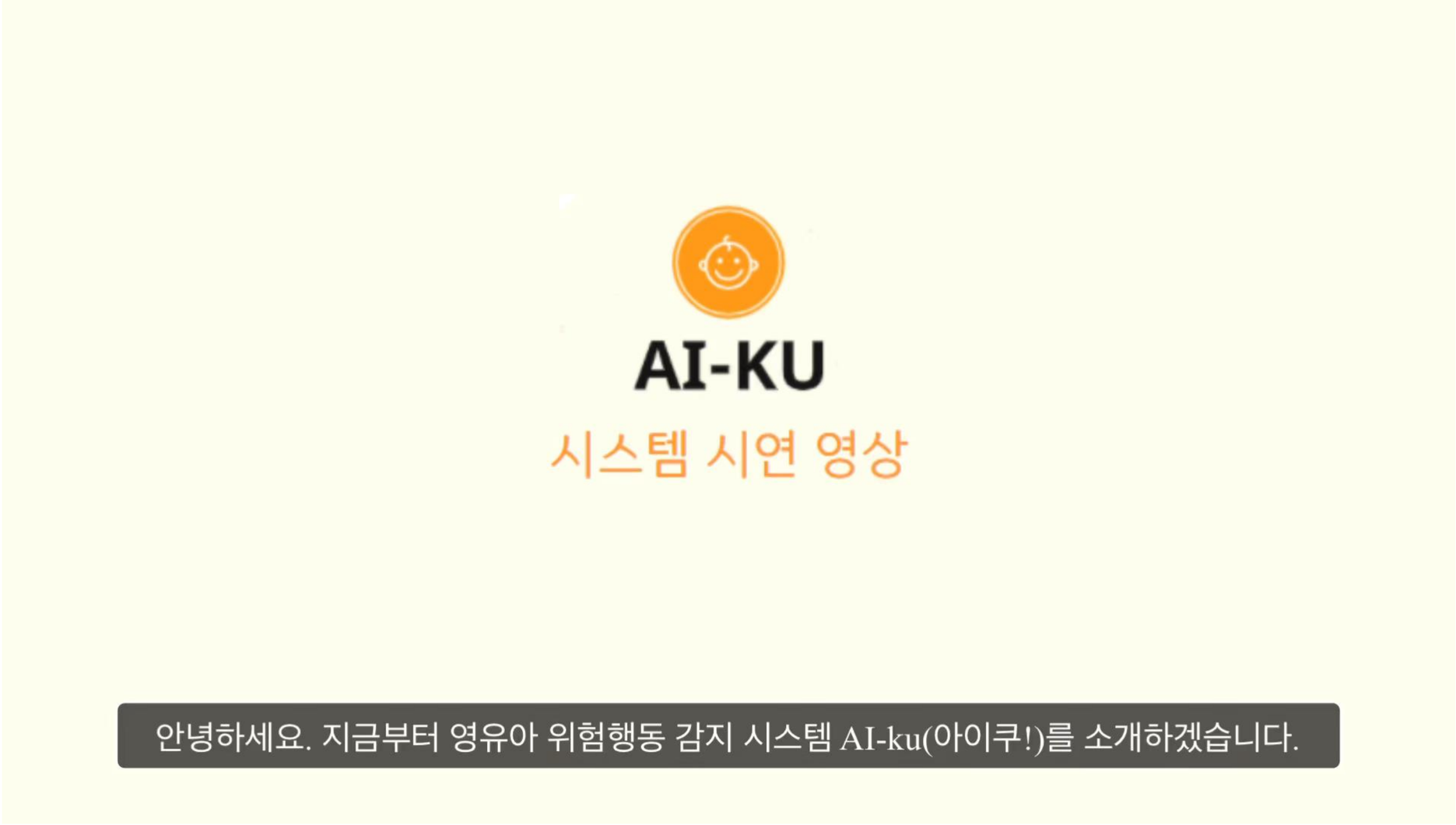
1. UI 소개
2. 시연 영상



3-1 UI 소개


AI-KU System
AI-ku!
image
<전화번호와 오디오를 저장한 후 카메라를 실행하세요>
번호를 입력하세요:
-없이 숫자만 가능
녹음을 시작하세요: (5초 동안)
화면 끄기
아이가 안전해요!
① 카메라 버튼
② 전화번호
③ 경고 메시지 녹음
④ 웹캠 화면
⑤ 아이 상태
⑥ 화면 끄기

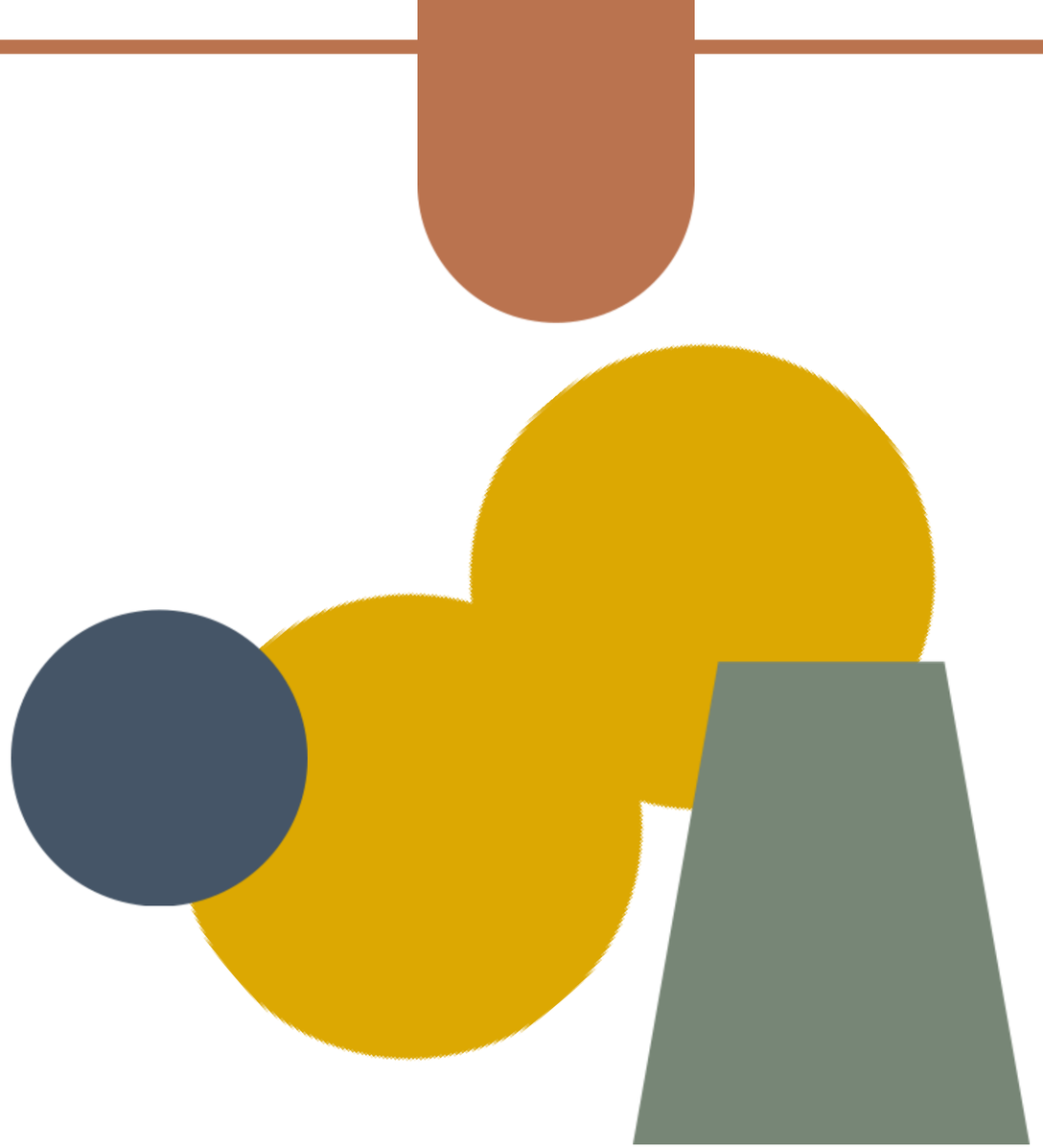
3-2 시연 영상



04

IV. 차별성 및 효과

1. 기존 서비스와의 차별점
2. 서비스 활용 방안
3. 기대 효과
4. 개선 방안



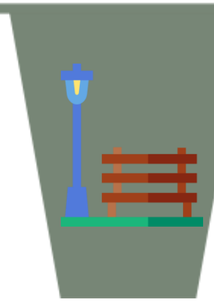
4-1 기존 서비스와의 차별점

	아이쿠!	KT IOT 홈캠	AI 보모
대상 연령층	영유아(1-3세)	노인, 영유아, 반려동물	신생아
기기 종류	스마트폰/ 웹캠	Cctv용 홈캠	Cctv, 행동 감지 센서
주요 기능	**움직임 감지/ 안전- 경고- 위험 3단계로 행동 분류/ 경고-위험에 맞는 알림 서비스**	소리 및 움직임 감지/ 오랫동안 움직임 없을 경우 알림	소리 및 움직임 감지/ 아이 울음소리로 감정 파악

4-2 활용방안

가정

- 가정 내 보호자가 요리를 하거나 청소를 할 때, 혹은 짧은 외출을 다녀오는 경우 등 아이 옆에 있지 못하는 경우



야외

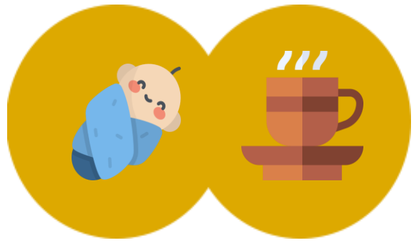
- 야외의 놀이 시설 등에서 놀고 있는 아이를 지속적으로 보지 못하는 경우

어린이집

- 어린이집에서 선생님이 동시에 모두를 관리하지 못하는 경우

키즈 카페

- 보호자가 대화를 나누고 있어 아이가 위험한 것을 인지하지 못한 경우



4-3 기대효과

1	대형 사고로 이어지는 것을 막고, 사고에 대한 신속한 대처 가능.	가정 내 영유아 안전사고 감소
2	키즈 카페, 놀이터, 어린이집 cctv 등 어린이 안전사고가 많이 일어나는 곳에서도 사용 가능	가정 외 영유아 안전사고 감소
3	일상적 기기에 연결 가능해 보급이 쉬움	홈캠/돌보미 비용 절감
4	보호자의 심리적 안정감 증대 및 육아 스트레스 감소	부모와 아이에게 있어 안정적인 육아 가능

4-4 개선 방향

데이터 수집의 한계로 인해 행동 인식이 부정확한 점

위험 물질에 제한되지 않은 '잡기' 경고 행동

실질적 효과 확인 부족

이용자 데이터 수집과 주기적으로 업데이트를 통해 정확성 증가

위험 물질 인식 모델을 제작해 현재 모델과 연동할 계획

사용자들의 의견을 적극 반영해 지속적인 모델의 수정을 이어 나갈 것

THE END

Thank you

2조

김윤정, 원하은, 정가희, 정준하