

# Image Recognition Android App

Genia BigData 4th, FirstLine

박준식, 이찬녕, 이형석, 임유하



# Intro

- **Why ?**

- Osmo에 대한 흥미
- 코딩 교육에 대한 관심

- **Android Application?**

- 팀원 모두 Android
- 스마트폰 내 카메라 활용

- **Purpose**

- 이미지 인식 기능이 있는 모바일 앱 구현
- 실제 모바일 앱이 만들어지는 전반적인 프로세스 탐구 및 구현



# Contents



**Time Table**



**Modeling  
Mobile Application  
Database  
Pipeline**



**Simulation  
Result**



# Time Table

업무	담당자	일정		
		23.10.30 ~ 23.11.17		
WEEK		1	2	3
착수 보고서 작성	공통	■		
기획 및 역할 분담	공통	■		
레퍼런스 모델 연구	박준식		■■■■■	■■■■■
	임유하		■■■■■	
	이찬녕		■■■■■	
어플리케이션 개발	이형석		■■■■■	■■■■■
DB 연동			■■■■■	■■■■■
Server 및 API 배포	박준식		■■■■■	■■■■■
API 연동	이형석		■■■■■	■■■■■
문서 작업	임유하		■■■■■	■■■■■
시스템 테스트	공통			
최종 정리	공통			

# Used Model

- **Juuns**

- 이미지 속에서 필요한 contours를 찾고 학습을 진행하여 이미지 인식 모델 개발

- **YOLO v8 Tuning**

- 기존 SOTA Image Model을 Tuning하여 본 프로젝트에 맞는 방향으로 개발

# Juuns



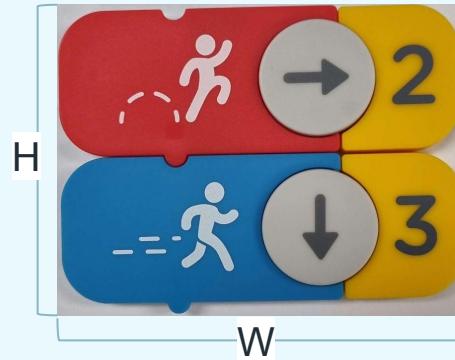
일정 크기 이상의 **Contour**들을  
구해서 이미지에 그려넣음

어떤 **Contours**가 **화살표**고 **숫자**인지  
어떻게 구분할 것인가?





# Juuns



**if**  $2 < W / H < 3$

→ 블럭 1개

**elif**  $1.2 < W / H < 1.4$

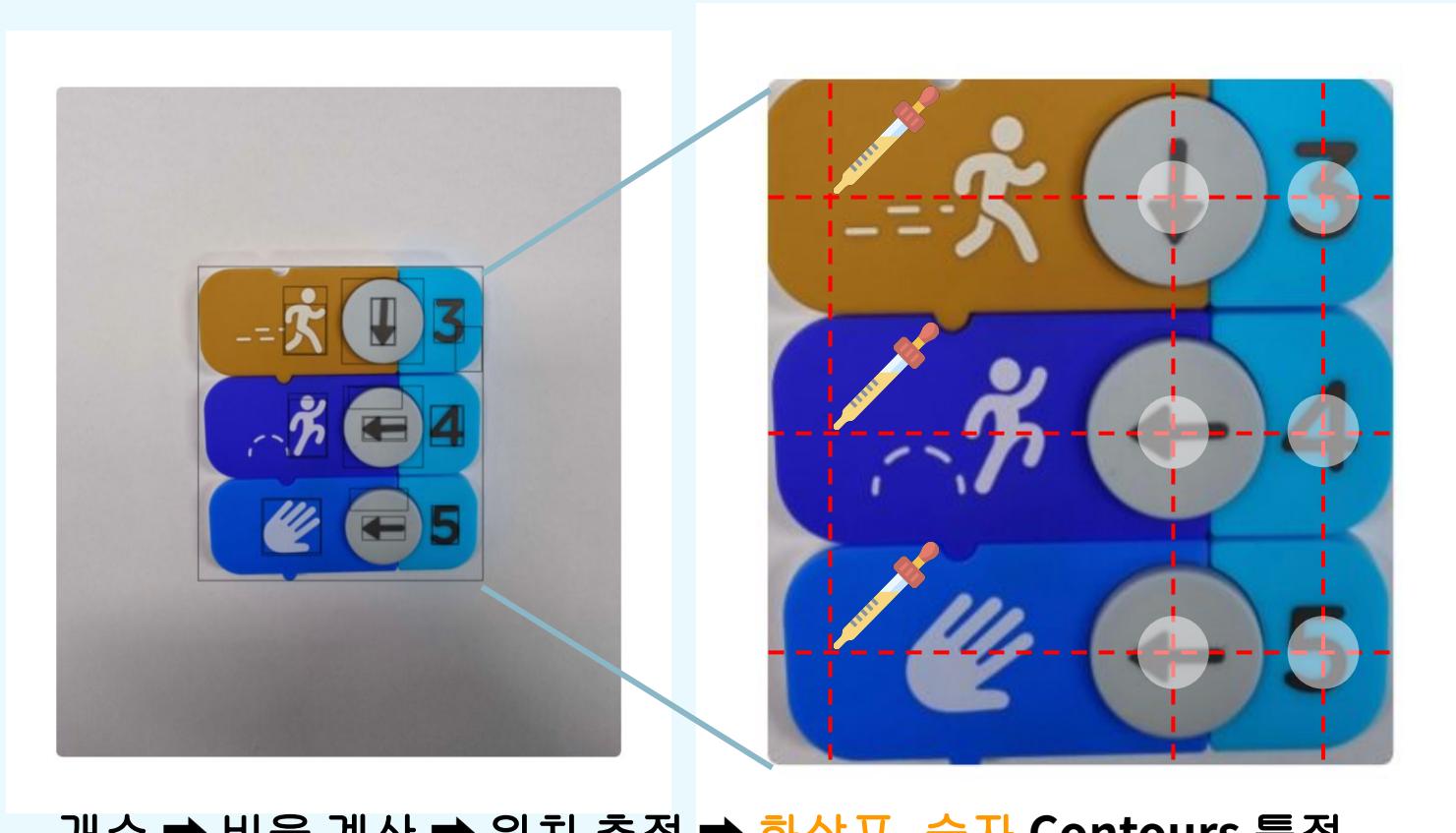
→ 블럭 2개

**elif**  $0.8 < W / H < 1.1$

→ 블럭 3개



# Juuns



개수 → 비율 계산 → 위치 추정 → 화살표, 숫자 Contours 특정



# Juuns

## < Contours >

Used Model 1



osmo\_arrow\_d.jpg  
g



osmo\_arrow\_d2.j  
pg



osmo\_arrow\_l.jpg  
g



osmo\_arrow\_l2.j  
pg



osmo\_arrow\_r.jpg  
g



osmo\_arrow\_r2.j  
pg



osmo\_arrow\_u.jpg  
g



osmo\_arrow\_u2.j  
pg



osmo\_arrow\_u3.j  
pg



osmo\_number\_fi  
ve.jpg



osmo\_number\_fi  
ve2.jpg



osmo\_number\_f  
our.jpg



osmo\_number\_t  
hree.jpg



osmo\_number\_t  
hree2.jpg



osmo\_number\_t  
hree3.jpg



osmo\_number\_t  
wo.jpg



osmo\_number\_t  
wo2.jpg



osmo\_number\_t  
wo3.jpg

학습을 위해 추출한 Contours





# Juuns

Used Model 1

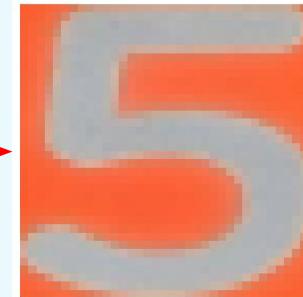
< Create Image >



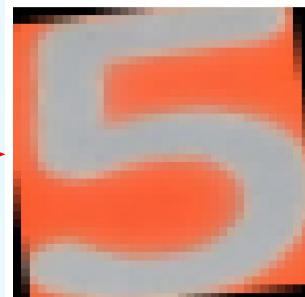
원본



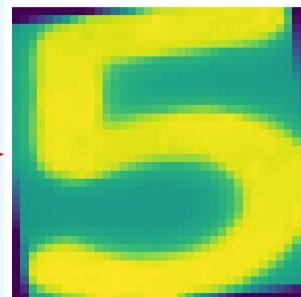
색 반전



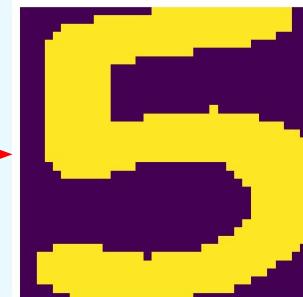
36 x 36



무작위 회전



BGR to Gray



이진화

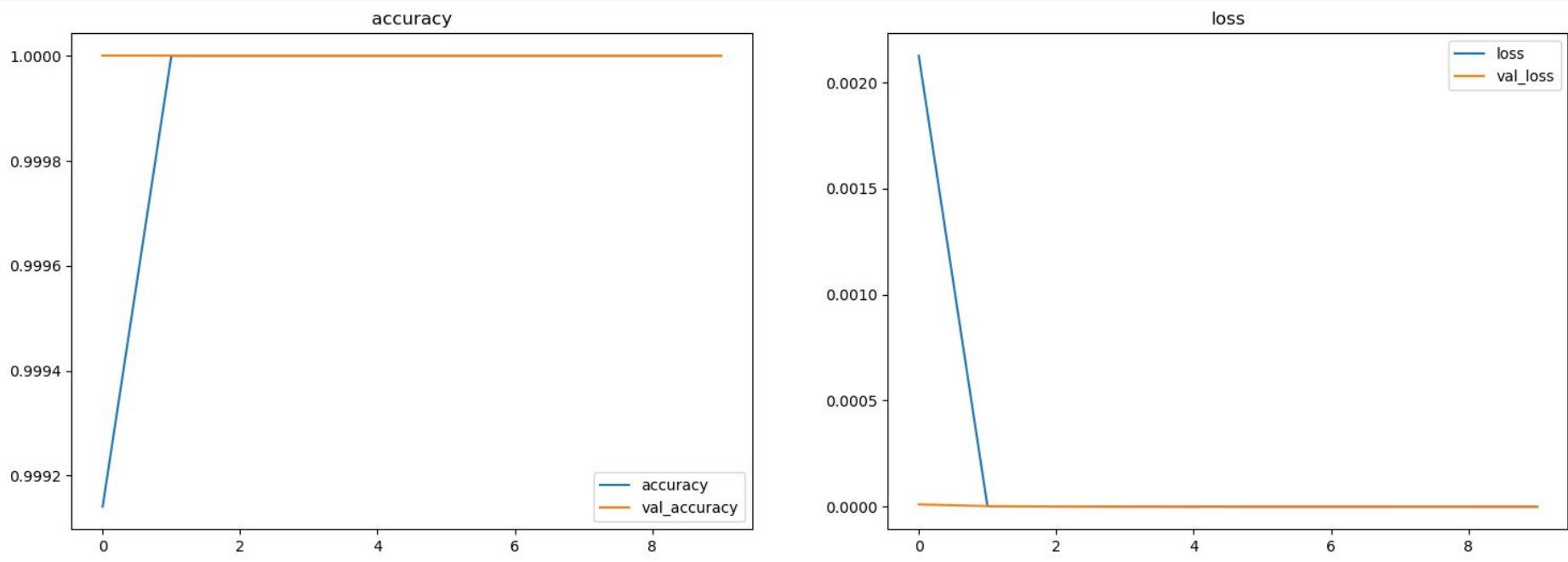




# Juuns

Used Model 1

## < Train History >

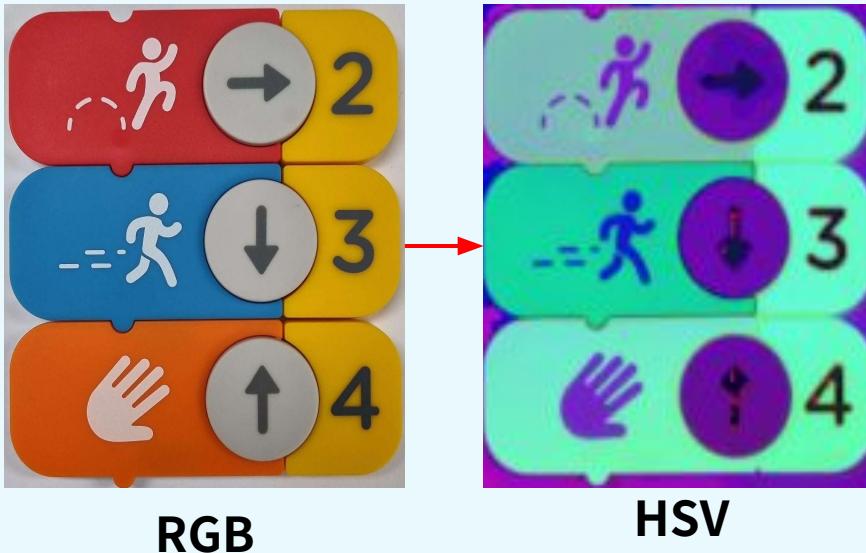




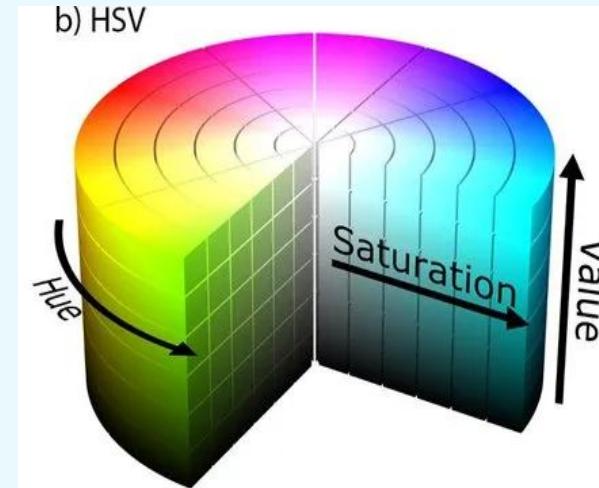
# Juuns

## < Action >

Used Model 1



{actions : 'Run', 'Hand', 'Jump'}



```
if H < 50 → Run  
elif H < 115 → Hand  
else → Jump
```





# Juuns

## < Result >

Used Model 1



```
1/1 [=====]  
1/1 [=====]  
['U', 'R', 'L']  
[3, 4, 5]  
['Jump', 'Run', 'Hand']
```

```
ValueError: min() arg is an empty sequence
```





# YOLO v8

< roboflow를 이용한 모델 학습 >

## 1. Create Repository

Create New Project

Tongsil / [New Public Project](#)

Project Type

 Object Detection  
Find multiple things and their specific location.

 Classification  
Assign labels to the entire image.

 Instance Segmentation  
Detect multiple objects and their actual shape.

Show More ↓

Project Name

Find\_Arrow\_Number

What are you detecting? [?](#)

arrow-number

License

CC BY 4.0

Cancel

Create Public Project

Create New Project 클릭 후,

Project Type – Object Detection 선택





# YOLO v8

< roboflow를 이용한 모델 학습 >

Used Model 2

## 2. Image Data Labeling

Upload Want to change the classes on your annotated images?

Batch Name: Uploaded on 11/15/23 at 11:03 am Tags: Search or add tags for images...

All Images 22 Annotated 0 Not Annotated 22

Drag and drop images and annotations.

Select Files Select Folder

The screenshot shows a user interface for image annotation. At the top, there are buttons for 'Upload' (with a note about changing classes), 'Save and Continue', 'Batch Name' (set to 'Uploaded on 11/15/23 at 11:03 am'), and 'Tags' (with a search bar). Below this, a status bar shows 'All Images 22', 'Annotated 0', and 'Not Annotated 22'. A large central area is titled 'Drag and drop images and annotations.' and contains a grid of small image thumbnails. Each thumbnail has a file name below it. The images appear to be various Osmo game pieces and arrows. There are also two rows of smaller, darker images at the bottom. On the right side of the main grid, there are two buttons: 'Select Files' and 'Select Folder'.



훈련 모델에 사용할 이미지 데이터를 가져온 후, 라벨링 할 부분에 드래그한 후 클래스 추가.

모든 사진의 라벨링 작업 후 Add # image to Dataset 클릭한 후에, 이미지 증강(선택사항) 후,  
Train, Valid, Test 비율 설정(70%, 20%, 10%)





# YOLO v8

< roboflow를 이용한 모델 학습 >

## 3. Export Dataset

This version doesn't have a model.

Train an optimized, state of the art model with Roboflow or upload a custom trained model to use features like Label Assist and Model Evaluation and deployment options like our auto-scaling API and edge device support.

[Get More Credits](#)   [Custom Train and Upload](#)

Available Credits: 0

3386 Total Images

[View All Images →](#)

Dataset Split

TRAIN SET	VALID SET	TEST SET
2964 Images	282 Images	140 Images

Custom Train and Upload

X

Export and Train

Export your dataset and then custom train with it using this provided code snippet. Select your model type below.

YOLOv8 (New!) ▾ [Get Snippet](#)

Upload Model Weights

Upload model weights to use with our model powered features and deployment options.

[How to Upload Weights ↗](#)

Custom Train and Upload - Get Snippet 후에 zip 파일로 데이터셋 내보내기





# YOLO v8

〈 roboflow를 이용한 모델 학습 〉

Used Model 2

## 4. Google Colab 실행 후 모델 학습 실행하기

Image sizes 800 train, 800 val						
Using 2 dataloader workers						
Logging results to runs/detect/train						
Starting training for 25 epochs...						
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
1/25	7.78G	1.883	3.295	1.988	131	800: 100% 128/128 [02:03<00:00, 1.04it/s]
	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95: 100% 1/1 [00:01<00:00, 1.80s/it]
	all	19	128	0.474	0.572	0.47 0.192
	2	19	15	0.291	0.8	0.302 0.11
	3	19	18	0.244	0.667	0.332 0.139
	4	19	16	1	0	0.511 0.225
	5	19	18	0.612	0.444	0.551 0.181
	arrow	19	61	0.225	0.951	0.652 0.307
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
2/25	8.56G	1.558	2.384	1.716	113	800: 100% 128/128 [01:54<00:00, 1.11it/s]
	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 1.44it/s]
	all	19	128	0.718	0.759	0.835 0.392
	2	19	15	0.443	0.867	0.788 0.302
	3	19	18	1	0.211	0.732 0.281
	4	19	16	0.964	0.875	0.942 0.534
	5	19	18	0.428	0.889	0.83 0.444
	arrow	19	61	0.755	0.951	0.881 0.4
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
3/25	8.51G	1.507	2.362	1.626	151	800: 1% 1/128 [00:04<08:50, 4.18s/it]

Roboflow에서 labeling한 데이터셋 가져온 후 모델 학습하기  
(epoch: 25, Image Pixel: 800)



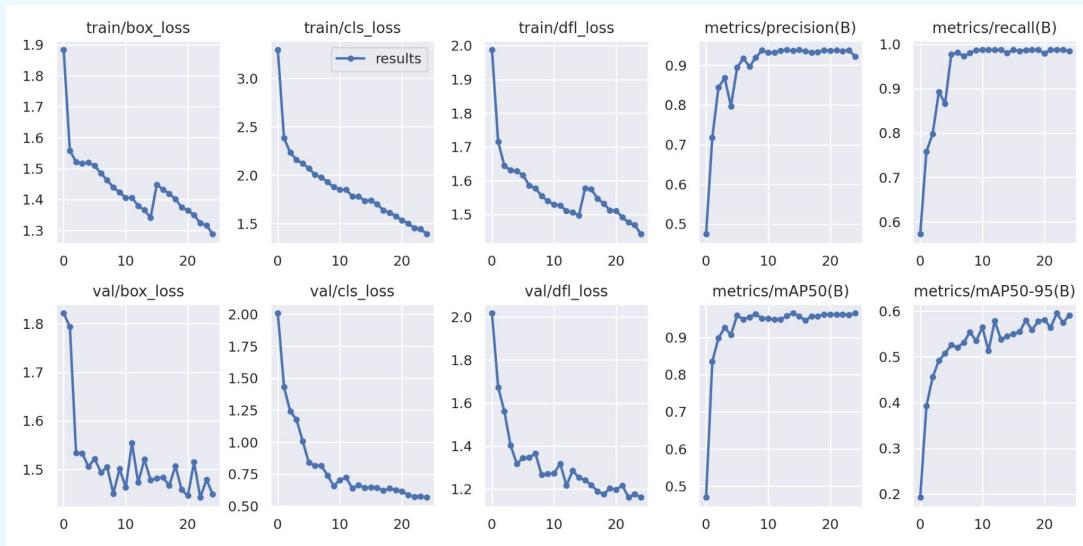
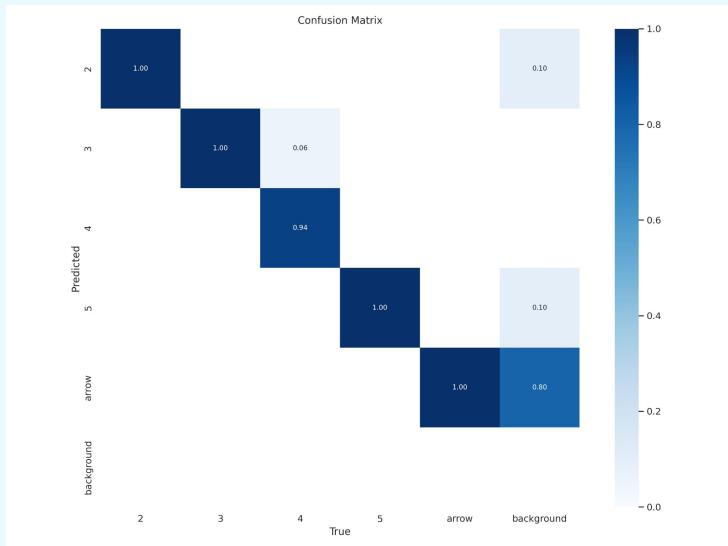


# YOLO v8

< roboflow를 이용한 모델 학습 >

Used Model 2

## 4-1. Google Colab 실행 후 모델 학습 실행하기



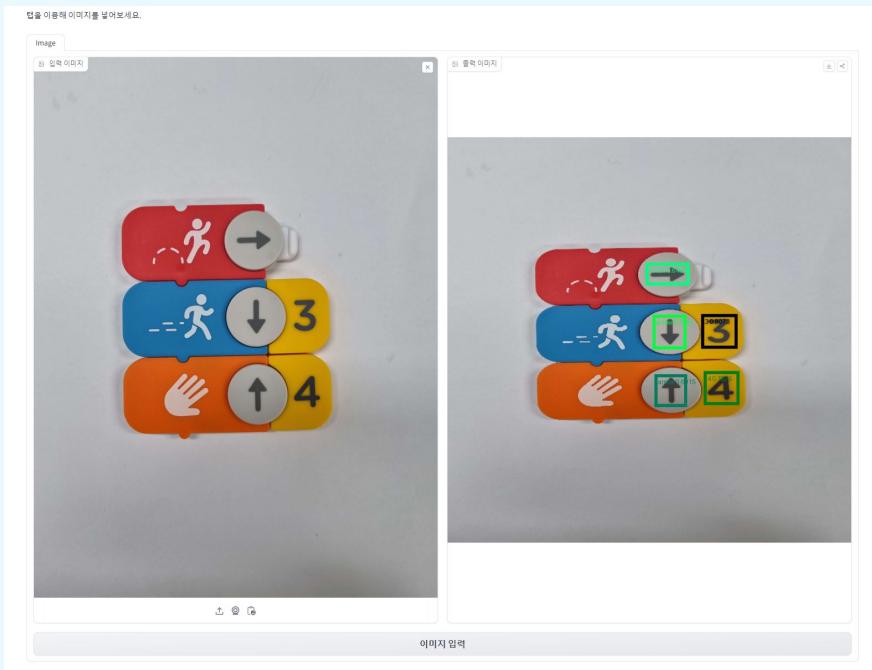


# YOLO v8

< Hugging Face 사이트로 이미지 인식 >

Used Model 2

## 5. Hugging Face 사이트를 이용하여 이미지 인식 테스트 생성



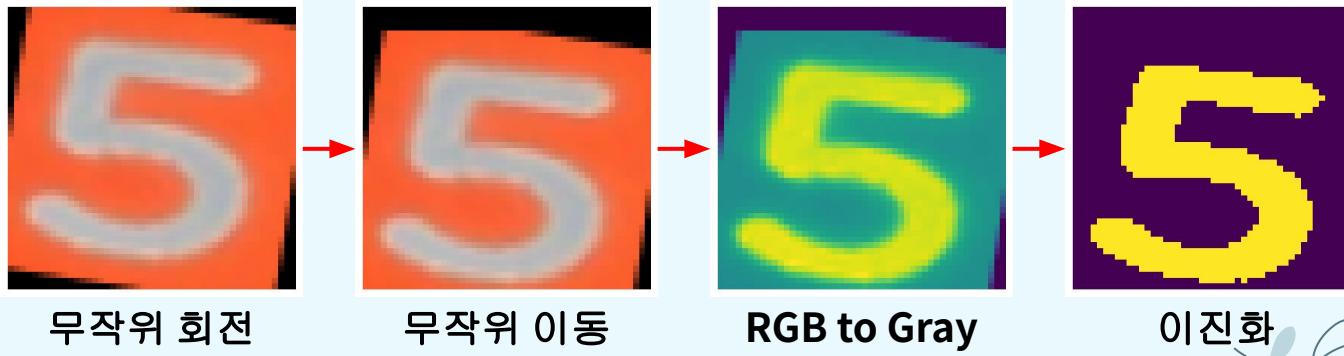
1. 학습된 yolo 모델을 gradio 모듈을 이용해서 테스트 생성
2. 학습된 yolo 모델 가져오기 및 이미지 input 박스 생성 코드 작성후 app.py에 저장
3. 필요한 모듈을 requirements.txt에, yolo 모델 라벨값을 values.py에 작성 후 저장
4. Hugging Face에 Space 생성 후, Files에 app.py, best.pt, requirements.txt, values.py 업로드 후 app 클릭 -> 테스트 박스 생성



# YOLO v8 Tuning



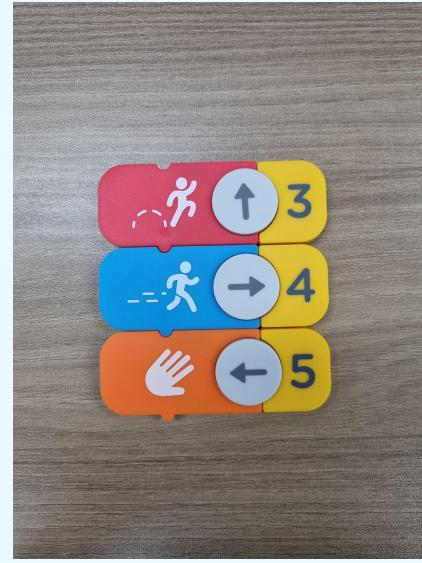
# YOLO v8 Tuning



# YOLO v8 Tuning



```
1/1 [=====]  
1/1 [=====]  
1/1 [=====]  
1/1 [=====]  
1/1 [=====]  
1/1 [=====]  
1/1 [=====]  
1/1 [=====]  
[ 'U' , 'R' , 'L' ]  
[ 3 , 4 , 5 ]  
[ 'Jump' , 'Run' , 'Hand' ]
```



# Mobile Application

< Mobile >



< Web >



다른 언어에 대한 경험 ✕ → 파이썬 내에서 앱을 완성하는 방법 탐구

Mobile 구현 실패에 대한 대안 → Web에 대한 기술 탐구도 진행

# Mobile Application



Flutter



Firebase Realtime Database

빠른 Prototype 도출 → Flutter 선택

DB 구축 → 클라우드 호스팅 서비스를 제공하는 Firebase

# Database

- Firebase - **Cloud Firestore**

The screenshot shows the Firebase Cloud Firestore interface. At the top, there's a navigation bar with icons for home, user, and a specific document ID: 1699950803825. To the right is a dropdown for "Google Cloud의 추가 기능". Below the navigation is a table structure. The left column shows a collection named "user" with a sub-collection "user". The middle column lists documents with IDs: 1699923183321883, 1699927513988630, 1699927584522270, 1699927694453076, and 1699927719607057. The right column shows a detailed view of the first document, which has a field "image" with a URL and a field "time" with a timestamp.

(default)	user	⋮
+ 컬렉션 시작	+ 문서 추가	+ 컬렉션 시작
user	1699923183321883 1699927513988630 1699927584522270 1699927694453076 1699927719607057	1699950803825193 ⋮ + 필드 추가 image: "https://firbasestorage.googleapis.com/v0/b/firstline-17bda.appspot.com/o/images%2F1699950803825193?alt=media&token=d09b2a1c-73b3-449d-86f6-fd580fc57538" time: 2023년 11월 14일 오후 5시 33분 28초 UTC+9

- Firebase - **Realtime Database**

The screenshot shows the Firebase Realtime Database interface. It displays a list of child nodes under a root node. Each child node contains a URL for an image and a timestamp. The nodes are numbered sequentially: 1700133668134512, 1700133745282615, and 1700133786662561. To the right of the list is a decorative graphic featuring three red circles connected by lines and a large blue leaf.

- 1700133668134512
  - image\_url: "https://firbasestorage.googleapis.com/v0/b/firstline-17bda.appspot.com/o/1700133668134512?alt=media&token=3ad76863-3816-4c02-a41b-701fc"
  - time: "2023-11-16 20:21:14.055316"
- 1700133745282615
  - image\_url: "https://firbasestorage.googleapis.com/v0/b/firstline-17bda.appspot.com/o/1700133745282615?alt=media&token=4541bc65-3cc4-41db-84d2-7eb91"
  - time: "2023-11-16 20:22:31.496241"
- 1700133786662561
  - image\_url: "https://firbasestorage.googleapis.com/v0/b/firstline-17bda.appspot.com/o/1700133786662561?alt=media&token=cc76604e-cbf7-4c8e-bb7a-1e802"
  - time: "2023-11-16 20:23:12.776664"

# Database

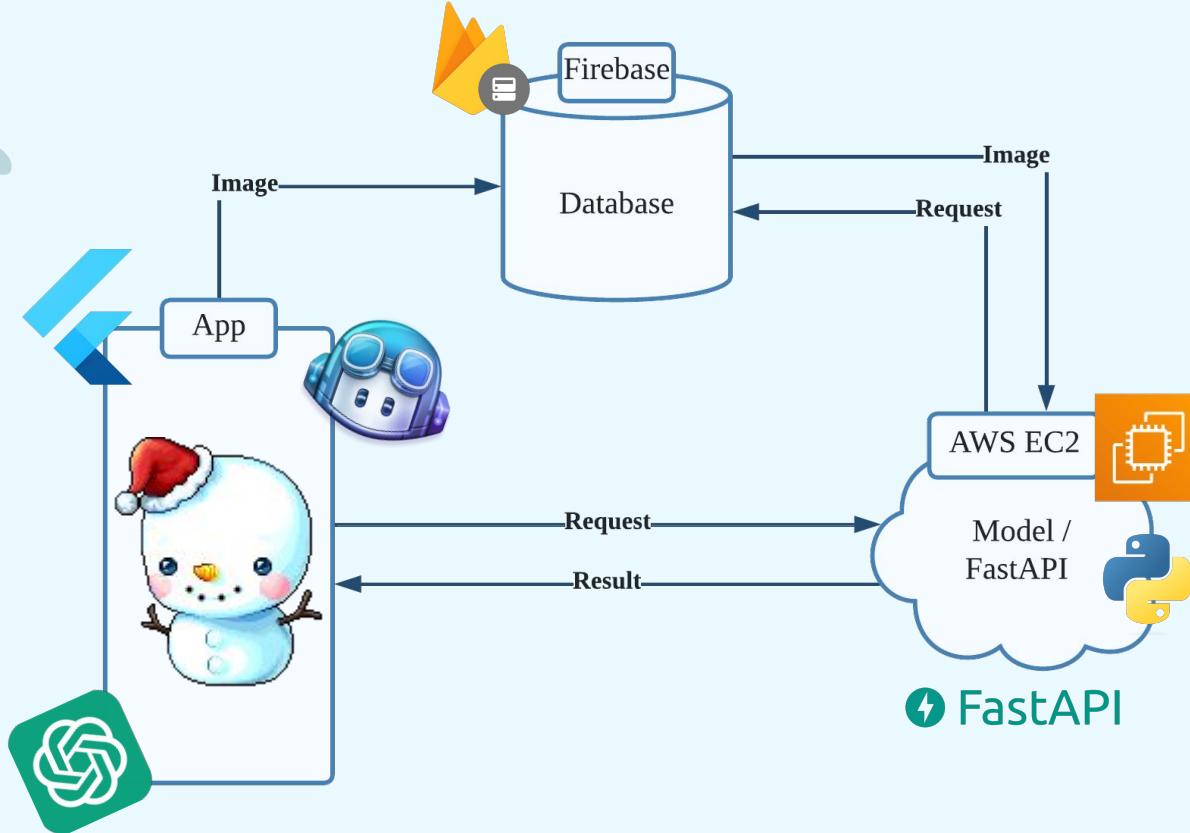
- Firebase - **Realtime Database**



**Realtime Data가 Model을 거쳐 결과가 나와야 함**

→ 두 가지 Database를 비교해 보고 조금 더 빠른 **Realtime Database** 선택

# PipeLine



Big4 FirstLine

초기 기능 구현

- 카메라
- 이동 버튼



# AWS 결제 대시보드

정보

페이지 새로 고침 시간: 2023년 11월 17일 금요일 오후 2시 16분 34초 GMT+9

숨김 해제



## AWS 요약 정보

AWS 비용 개요 보기

이번 달의 총 예상 정보

USD 4.62

**KRW 6,026.14**

총 활성 서비스 수

**6**

현재 당월 누계 잔액

USD 1.17

**KRW 1,527.75**

총 활성 AWS 계정 수

**1**

통화

KRW - South Korean Won



추세가 있는 동일한 기간의 이전 달

표시할 데이터 없음

총 활성 AWS 리전 수

**2**

# Final

## 📌 추가 기능

- UI 변경
- Database 연동
- AWS 서버 구축 및 API 연동



< Osmo Coding 교구 예시 >

## 🛠️ 사용법

- 좌측 하단 카메라 모양의 버튼을 눌러 사진을 찍는다  
로딩이 완료되면 ★JUUNS★ 또는 ♥YOLO♥ 버튼을 클릭하여 루돌프가 움직이는지 확인한다

- \* 본 어플은 Osmo Coding의 교구 데이터로 제작됨
- \* '루돌프'는 상, 하, 좌, 우 그리고 2~5의 숫자를 인식함 (Motion 인식 불가)
- \* 로딩 이유 : DB에 적재되는 시간, 실험 결과 최대 약 5초
- \* Error Message가 뜨면 화살표를 인식하지 못한 것
- \* 모델을 눌렀는데 아무런 반응이 없다면 사진을 다시 찍고 시도하기!



# System Test

연결 ②

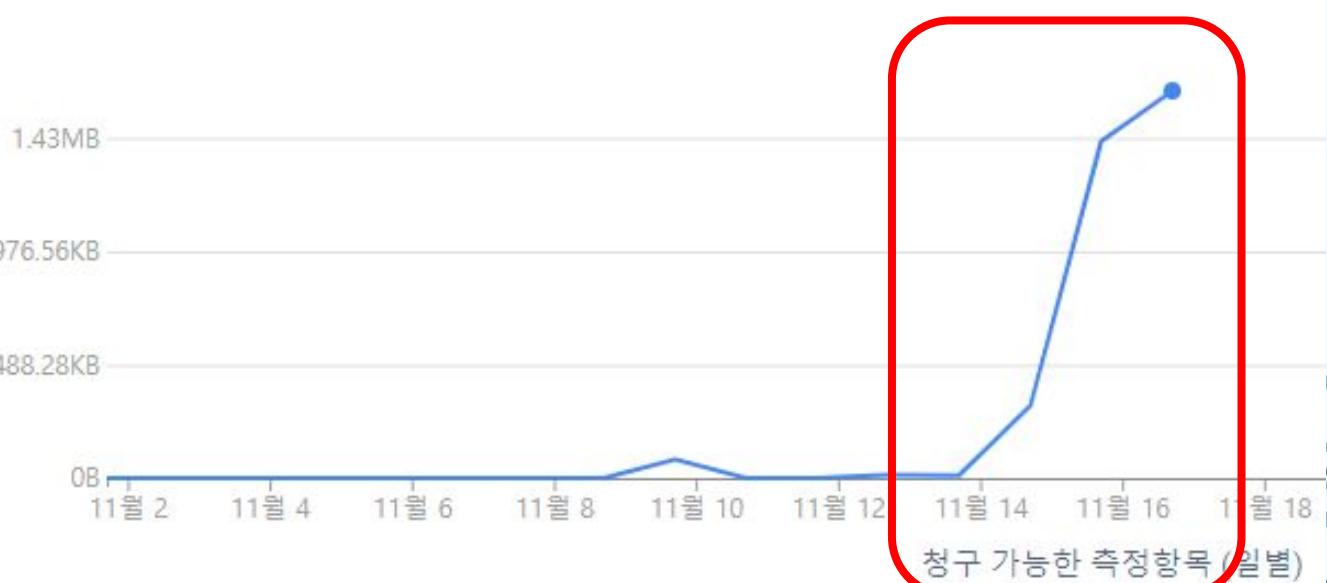
○ 2 / 100

스토리지 ②

○ 22.88KB 현재

다운로드 ②

○ 3.45MB 합계



프로젝트 종료 3일 전부터 본격적인 테스트

# Simulation



<https://youtu.be/u0yVr2vYNkU>

# Result

- 고도화 방안

- 교육 콘텐츠
- 게임성

- 개선점

- 모델 유연성
- 자연 시간 최소화 연구



감사합니다 🧑

# References

5세 유아의 특징 – [5세 발달 특성>연령별 발달정보>육아정보 | 아동교육연구지원센터 \(silla.ac.kr\)](#)

OSMO – [https://www.playosmo.com/ko-KR/](#)

Yolov8 이미지 인식 Computer Vision & Machine Intelligence Lab. [YOLOv8 – Computer Vision & Machine Intelligence \(catholic.ac.kr\)](#)

Opencv – [https://github.com/opencv/opencv](#)

[OpenCV – Open Computer Vision Library](#)

dynamo db, s3 – [https://docs.aws.amazon.com/](#)