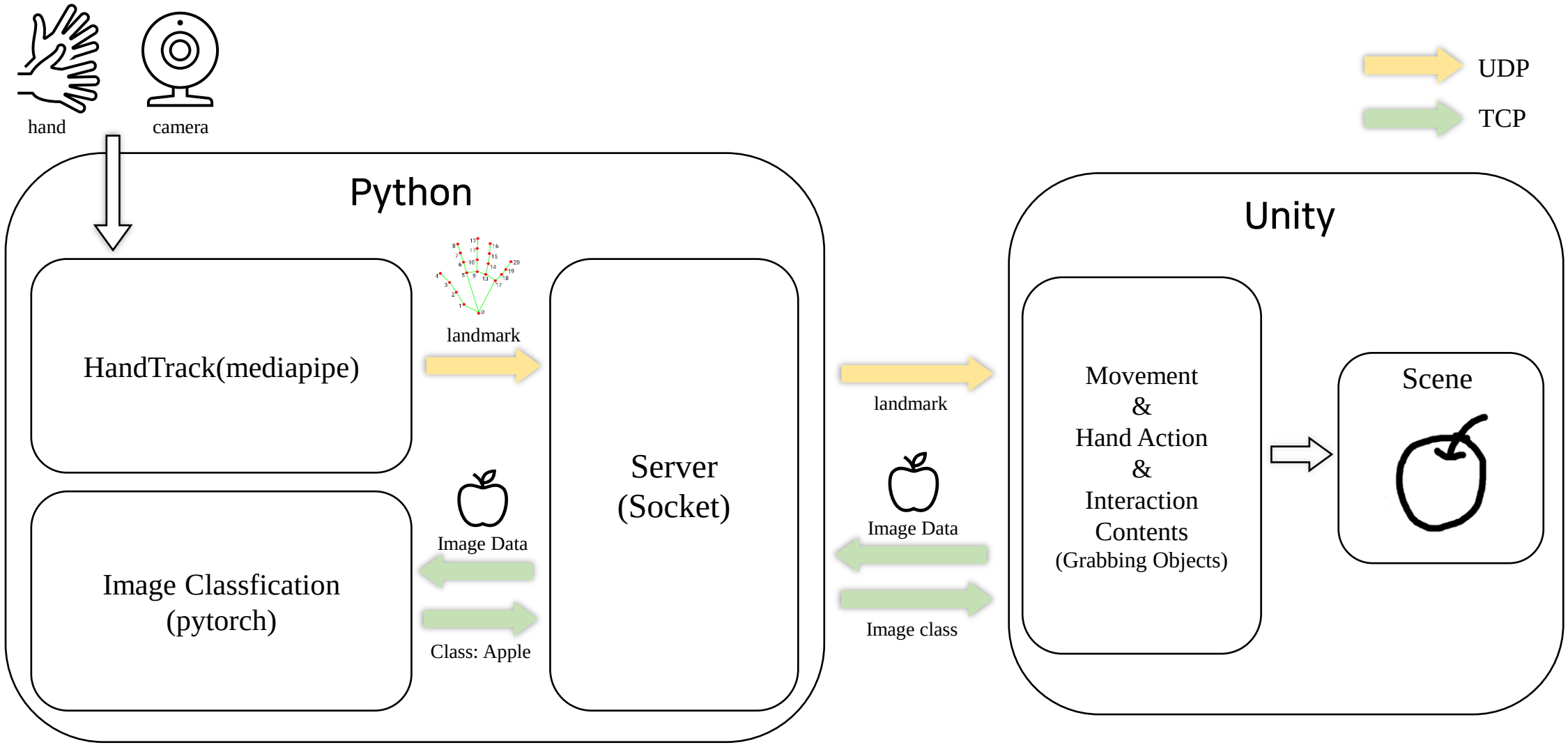
[22-2] 응용프로그래밍 Team project

DRAWING ROOM

9조. 김형빈 박지호 전혜령

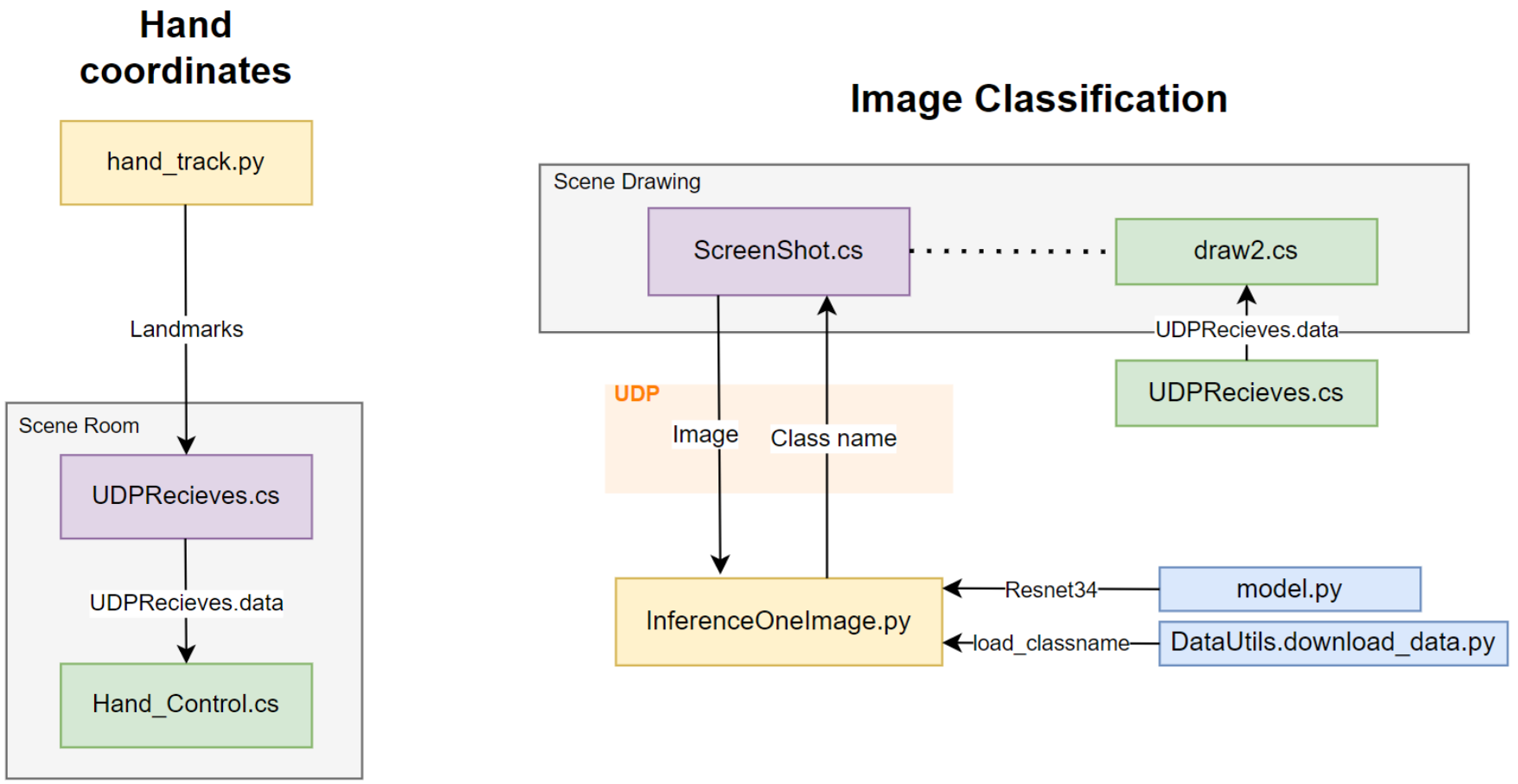
Contents

Preview


hand
camera
UDP
TCP
Python
HandTrack(mediapipe)
Image Classfication
(pytorch)
landmark
Image Data
Class: Apple
Server
(Socket)
landmark
Image Data
Image class
Unity
Movement
&
Hand Action
&
Interaction
Contents
(Grabbing Objects)
Scene

Preview

- Code structure



1. UNITY

1. Room

목표 : 2가지 object(펜, 지우개)를 습득해 drawing scene으로 진입하는 것

- 손 관절 마디 단위의 실시간 트래킹 구현
- 손의 움직임과 카메라의 움직임을 동기화해 1인칭 시점 구현
- 캐릭터의 이동은 키보드를 이용
- 물건을 습득하는 애니메이션 구현

1. UNITY

2. Draw

목표 : 주어진 키워드의 특징을 살린 그림을 그려 AI classification을 성공시키는 것이 목표

- 손이 주먹 쥔 모양으로 추정될 때 pen과 trail renderer 활성화
- 손의 움직임을 정교하게 추적해 3D 모델과 동기화. 실제 손의 자취에 따라 화면에 그림이 그려짐
- 손이 우측 상단의 버튼과 맞닿을 시 그림 초기화(지우개)

→ TrailRenderer.Clear() 함수 사용

- 손이 우측 하단의 버튼과 맞닿을 시 그림 제출

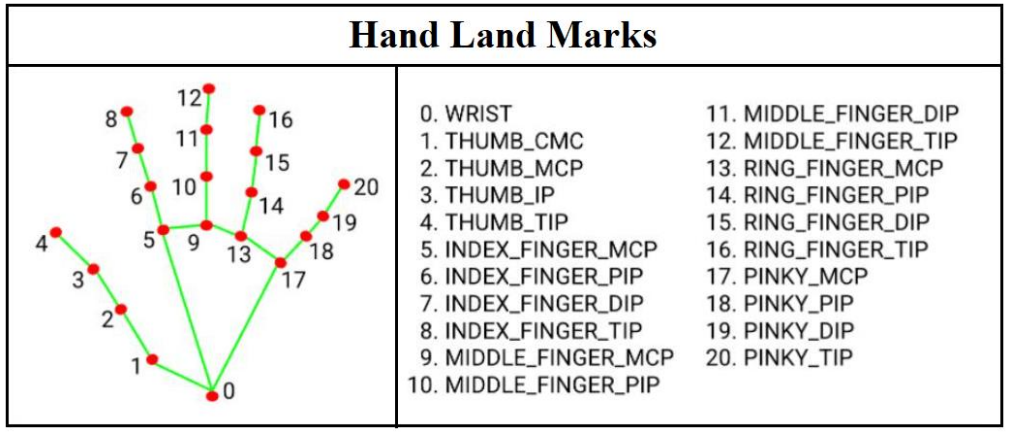
→ PNG형태로 파이썬에 송신

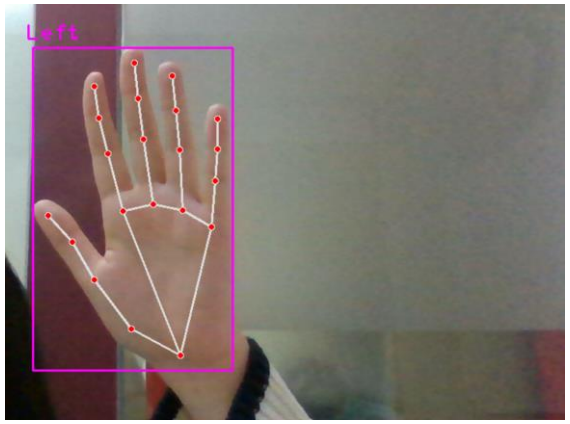
- Classification 결과를 수신해 화면에 출력

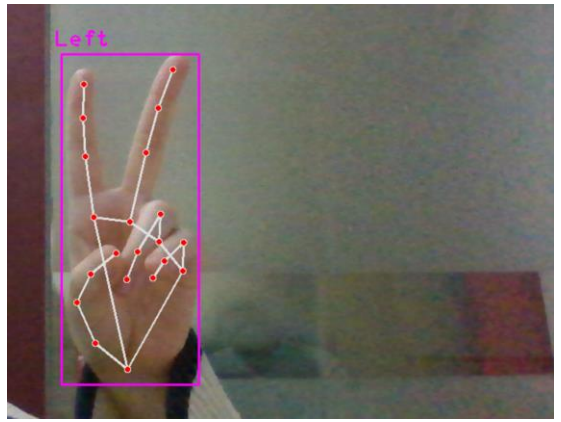
2. PYTHON

1. Hand tracking

- cvzone/HandTrackingModule.py 사용





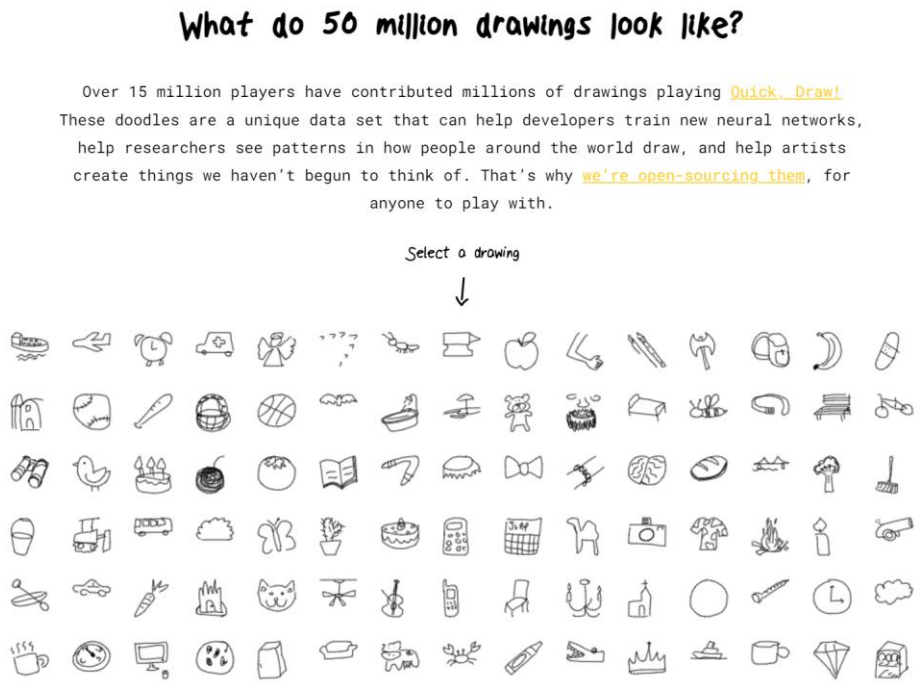


- 기본적으로 mediapipe hand module 사용
- [handtype, x_1, y_1, z_1, x_2, y_2, z_2……] 형태의 리스트로 전송

2. PYTHON

2. Image classification

- Data Preprocessing



Images

30 Classes

sun	clock
laptop	hot_dog
baseball_bat	smiley_face
eyeglasses	apple
book	bed
bread	shorts
table	broom
cloud	flower
chair	spider
headphones	cell_phone
face	car
eye	camera
Airplane	tree
snake	bicycle
pants	
star	

2. PYTHON

2. Image classification

Try1: Custom Model

	Accuracy
10 epochs	55%
30 epochs	77%

```
class ConvNet(nn.Module):
    def __init__(self, numclasses):
        super(ConvNet, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 64, 3, 1, 1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2)
        )
        self.layer2 = nn.Sequential(
            nn.Conv2d(64, 128, 3, 1, 1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2)
        )
        self.layer3 = nn.Sequential(
            nn.Conv2d(128, 512, 3, 1, 1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2)
        )
        self.fc = nn.Sequential(
            nn.Linear(512*3*3, 512),
            nn.Linear(512, numclasses)
        )
```

```
    def forward(self, net):
        net = self.layer1(net)
        net = self.layer2(net)
        net = self.layer3(net)
        net = net.view(net.size(0), -1)
        net = self.fc(net)
        return net


def convnet(numclasses):
    return ConvNet(numclasses)
```

2. PYTHON

2. Image classification

Try2: Resnet 18

```
def resnet18(numclasses, pretrained=False):
    model = models.resnet18(pretrained)
    conv1_out_channels = model.conv1.out_channels
    model.conv1 = nn.Conv2d(1, conv1_out_channels, kernel_size=3,
                            stride=1, padding=1, bias=False)
    model.maxpool = nn.MaxPool2d(kernel_size=2)
    fc_features = model.fc.in_features
    model.fc = nn.Linear(fc_features, numclasses)
    return model
```

	Accuracy
10 epochs	67%
30 epochs	85%

Try3: Resnet 34

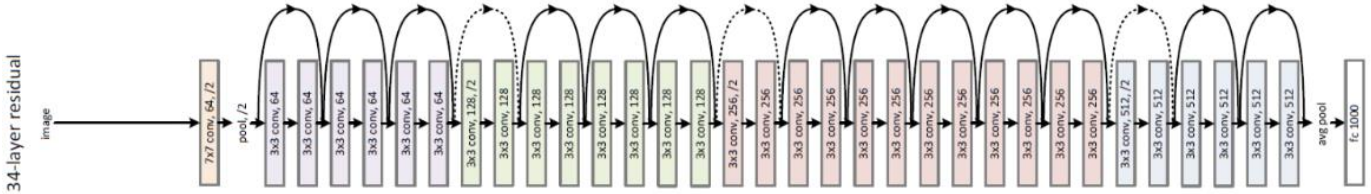
```
def resnet34(numclasses, pretrained=False):
    model = models.resnet34(pretrained)
    conv1_out_channels = model.conv1.out_channels
    model.conv1 = nn.Conv2d(1, conv1_out_channels, kernel_size=3,
                            stride=1, padding=1, bias=False)
    model.maxpool = nn.MaxPool2d(kernel_size=2)
    fc_features = model.fc.in_features
    model.fc = nn.Linear(fc_features, numclasses)
    return model
```

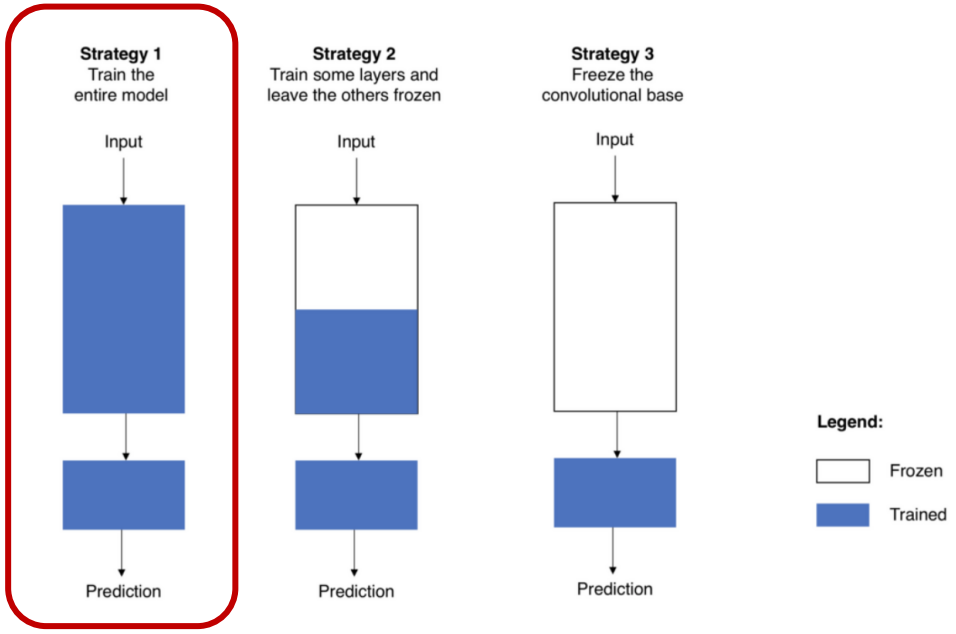
	Accuracy
10 epochs	67%
30 epochs	90%

2. PYTHON

2. Image classification

최종 모델: Resnet 34





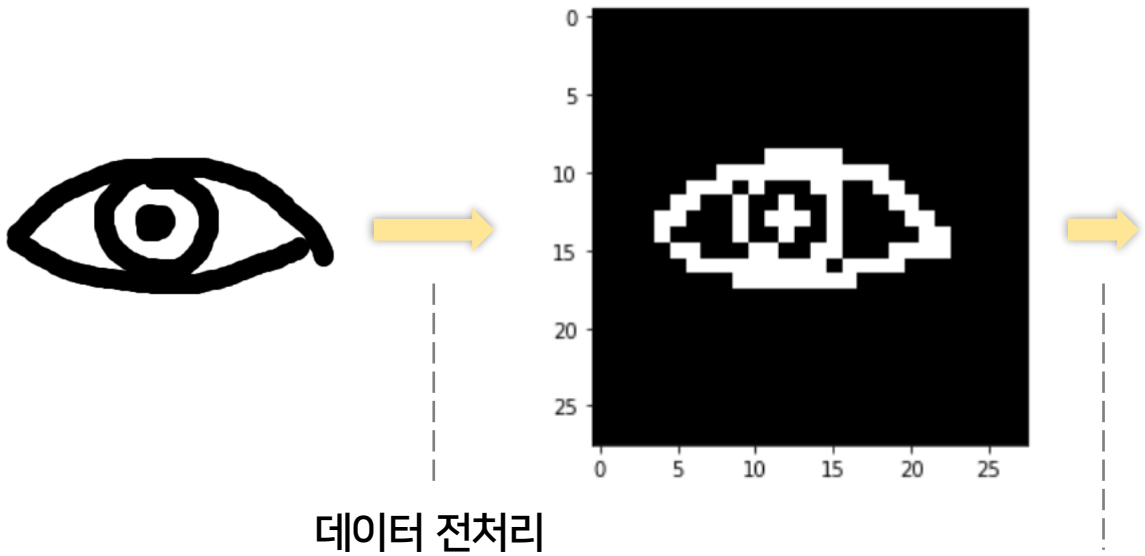
- epoch : 30
- model : resnet34
- batch size : 256
- lr : 0.1

- optimizer : SGD
- loss function : cross-entropy
- accuracy : 90%

2. PYTHON

2. Image classification

- Inference




```
# predict
result = PredictSingleImage(net, data)
```

```
tensor([[ 0.2638, -0.5398, -1.6652,  0.0292, -0.5666,  0.9885, -1.2899,  0.7040,
         -2.1998, -1.5896,  2.2839,  9.7234,  1.0468, -0.0482, -0.4314,  0.0739,
         -0.5499,  0.0844,  1.8666, -1.1743, -1.7301, -0.6513, -1.1335,  0.3619,
         -1.2093, -0.1811, -0.2529,  0.3707, -1.9676, -0.6170]],
       grad_fn=<AddmmBackward0>)
tensor([11])
```

```
def load_classname(classtxt_root = "class_text/30categories.txt"):
    f = open(classtxt_root,"r")
    # And for reading use
    classes = f.readlines()
    classes = [c.replace('\n','').replace(' ','_') for c in classes]
    return classes
```

```
classname = load_classname()
```

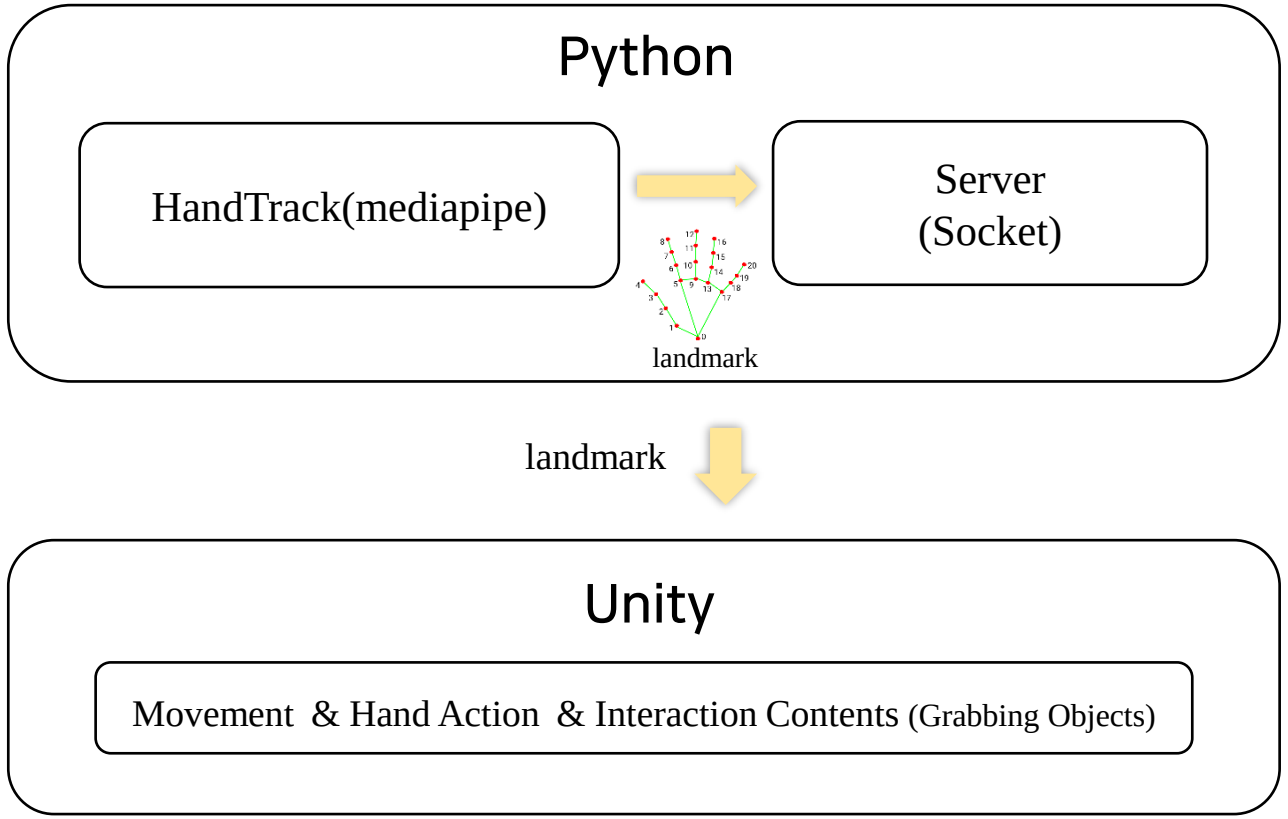
```
classname[result]
```

```
'eye'
```

3. Socket Programming

1. Hand coord

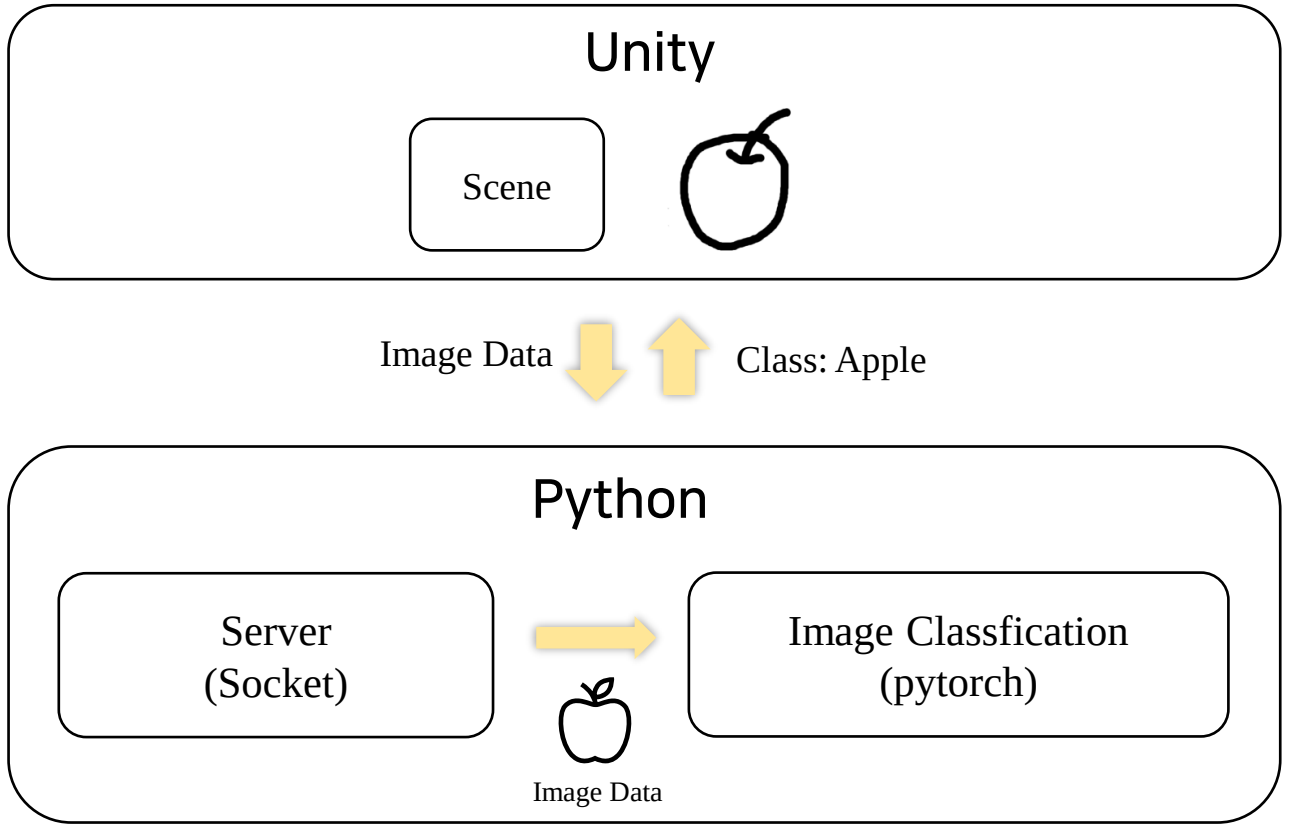
- UDP 통신
  - UDP의 경우 안정성 확보가 되지 않고, 전송 순서가 보장되지 않으며 자신의 포트 확인 이전에는 데이터가 왔는지를 알 수 없음
  - 그러나 TCP에 비해 상대적으로 빠른 속도를 갖는다는 장점이 있음



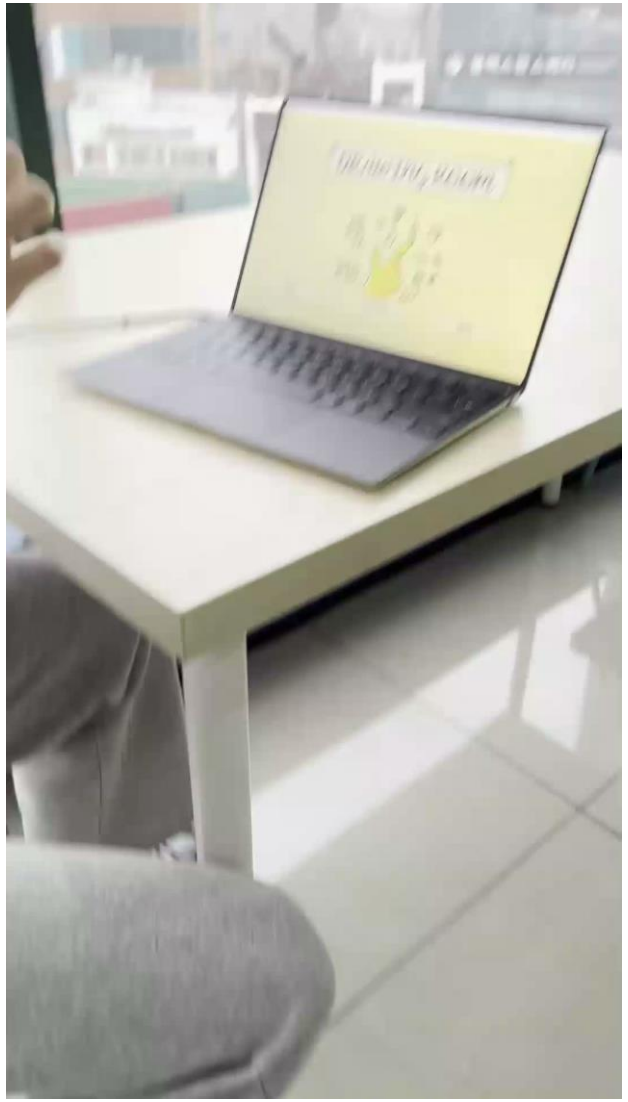
3. Socket Programming

2. Send Image

- TCP 통신
  - TCP 통신의 경우 계속해서 소켓을 통해 데이터를 주고 받을 수 있는 양방향 통신 가능
  - 데이터를 받지 못하였을 때 다시 요청하는 메커니즘이 존재하므로 연결을 신뢰할 수 있고 순차적임



Result



[Github] https://github.com/jiho-00/Unity_HandTracking_DeepLearning

한계점

1) 손 구현

- Hand landmark를 따온 뒤 실시간으로 대응되는 3D hand를 구현하는데 어려움을 겪음

2) 딜레이, 버벅임

- 반복문 줄이기, 코드 최적화

3) TCP 통신 오류 (각 소켓 주소는 하나만 사용할 수 있다)

- 통신 script가 반복적으로 실행될 때 발생함
- 소켓 옵션에 SO_REUSEADDR 적용해 같은 포트에 다른 소켓이 bind 하는 것을 허용
- 반복문에서 소켓 생성하지 않도록 알고리즘 수정