

동거동락 인생과외
집사부 일체

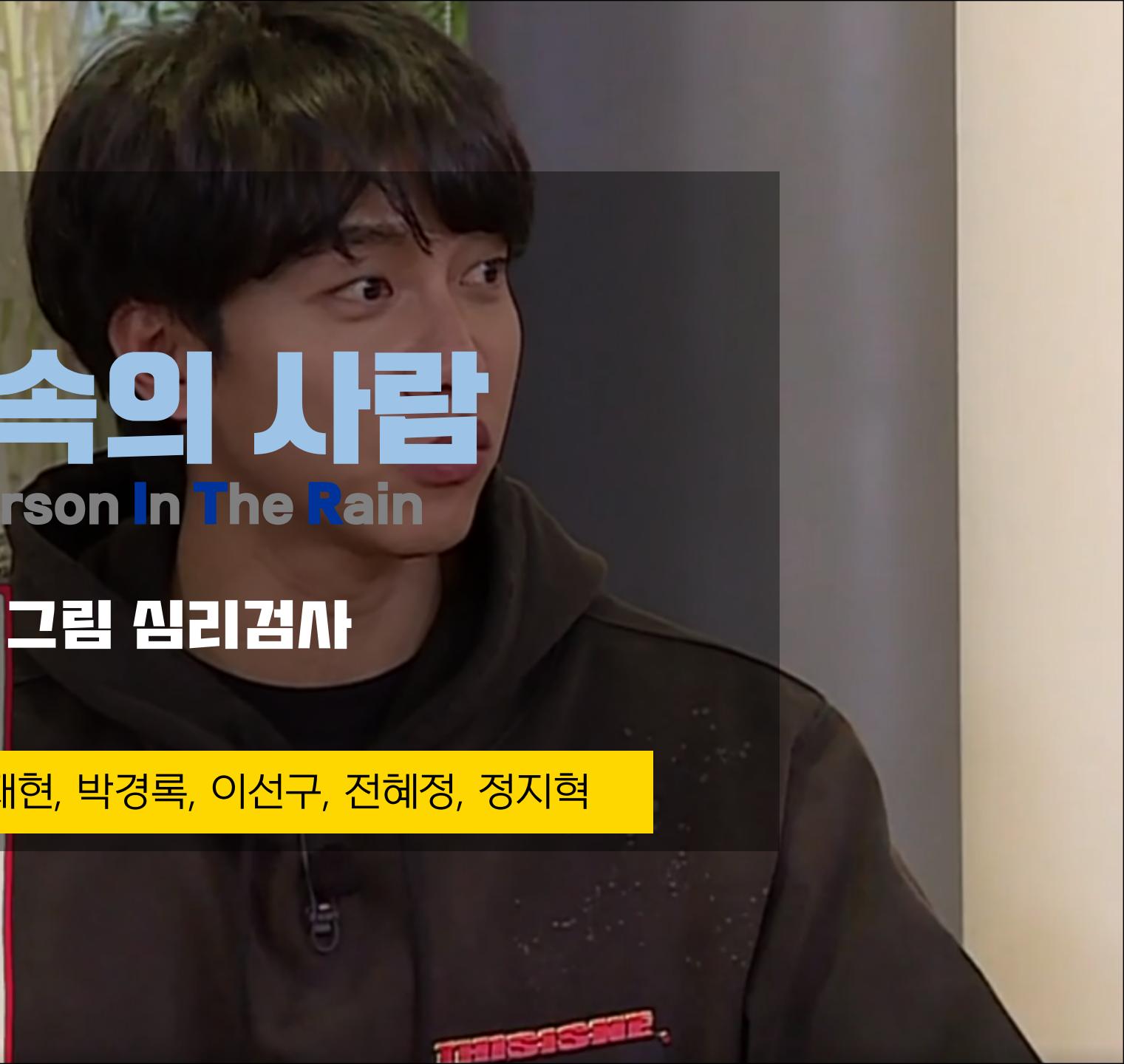
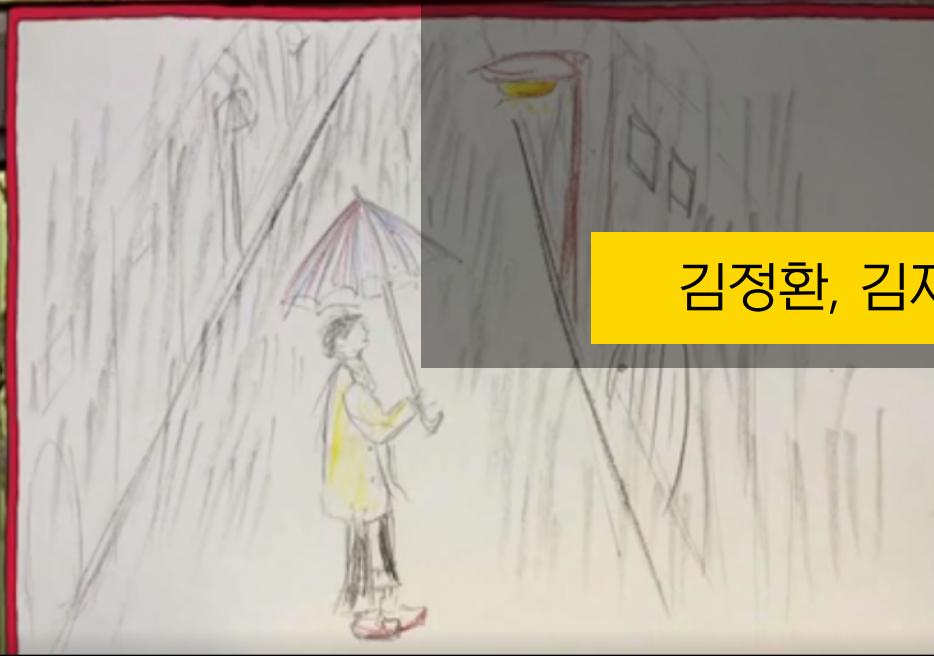
오늘 저녁 6시 25분

빗속의 사람

Person In The Rain

그림 심리검사

김정환, 김재현, 박경록, 이선구, 전혜정, 정지혁



빗속의 사람

Person In The Rain

그림 심리검사

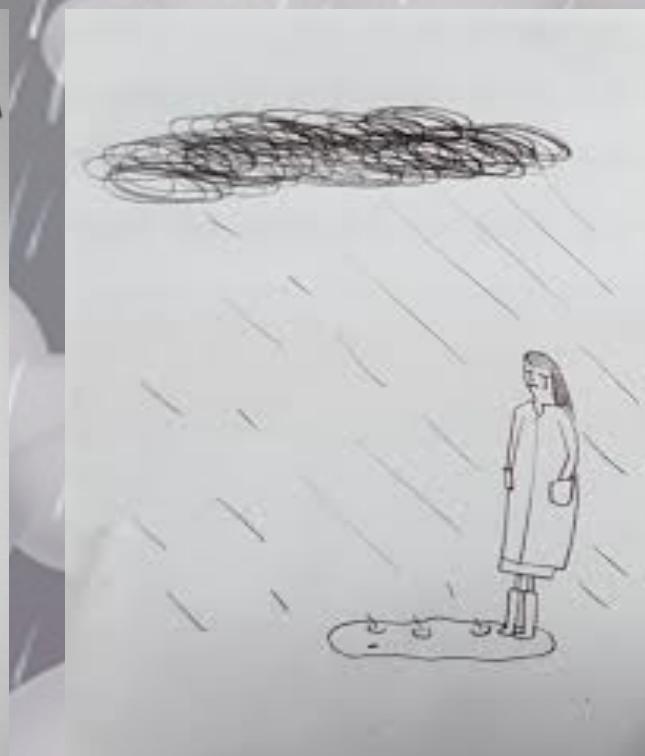
김재현
김정환
박경록
이선구
전혜정
정지혁

목차

1. 빗속의 사람 그림검사 소개
2. PITR 모델링 과정
3. 코드 설명 및 구현 화면
4. 한계점
5. 참고자료

1. 빗속의 사람(PITR)

- 빗속에 서 있는 사람을 그리게 하여 자아강도와 스트레스 대처능력의 수준을 확인하는 투사적 그림 검사
- 빗물, 웅덩이, 천둥, 먹구름 등은 모두 스트레스를 표현
- 우산, 건물 등 비를 막을 수 있는 도구는 스트레스 방어력을 나타냄



1-1. 검사 방법

- 지시사항 전달: 그리기 전 기본적인 지시사항 전달
- 검사실시: 피검자가 지시사항에 따라 그림을 그리는 단계
- 추가질문: 그림 그리기가 완료된 후 검사자가 추가적인 질문을 하는 단계

주의!

사람은 막대사람(졸라맨)이
아닌 사람으로 그려주세요.



비가 내리고 있습니다.

빗속에 서 있는 사람을 그려주세요.

중간에 떠오르는 것이 있다면 자유롭게 그려주세요.

준비물

- A4용지
- 연필
- 지우개



1-2. 검사결과 해석

• 객관적평가

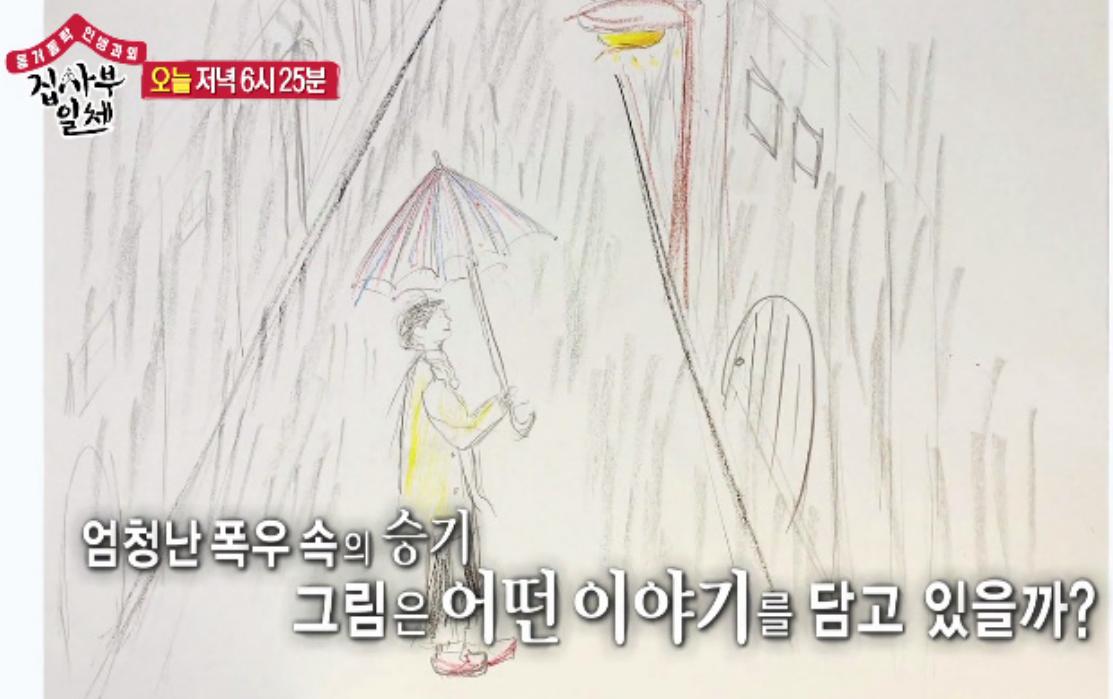
- 스트레스 척도와 자원 척도를 바탕으로 그림에 대한 스트레스 점수와 자원 점수를 산출한다.
- 산출된 점수를 바탕으로 최종 대처 능력점수를 계산하여 해석한다

*대처능력점수 = 자원점수 - 스트레스점수

스트레스 스케일			
항목	항목명	채점 지침	점수
S1	비가 없다	0- 비가 있다, 1- 비가 없다	
S2	비가 있다	0- 비가 없다, 1- 비가 있다	
S3	많은 비	Template Grid 사용 비가 차지한 공간 = R 사람이 차지한 공간 × 1.5 = WPerson R - WPerson = 경과가 음수(-) = 0, 경과가 양수(+) = 1	
S4	비의 스타일	0- 비가 점으로만 묘사 1- 비가 원, 선, 눈물방울 등	
S5	비의 방향	0- 비가 무질서하게 / 고르게 퍼져있음 1- 비가 특정인을 향하고 있음. 비스듬하거나 사람 위에 직접 내리고 있음	
S6	비의 접촉	0- 비가 인물의 어떤 부분 / 우산, 모자와 접촉되 지 않음 1- 비가 인물이나 그 인물의 어떤 부분과 접촉	
S7	젖다	0- 인물이 젖을 것 같지 않거나 마른 상태 1- 인물이 젖을 것 같음(비가 생략되더라도)	
S8	바람	0- 바람이 없다, 1- 바람이 있다	

자원 스케일			
항목	항목명	채점 지침	점수
R1	보호물이 있다	0- 보호물이 없다(R10으로), 1- 보호물이 있다.	R1 - R16 = 0점, 1점
R2	우산이 있다	0- 우산이 없다(R4로) 1- 우산이 있다(접혀 있는 것도 가능)	
R3	들고 있는 우산	0- 우산과 손잡이를 이상하게 들고 있을 때 1- 우산을 적절하게 들고 있을 때	
R4	다른 보호 장치	0- 우산과 비를 피하는 장비(우비, 비모자, 장화) 외의 다른 보호 장치가 없음 1- 우산과 비를 피하는 다른 보호 장치가 있음 (나무, 차, 가방, 자켓)	
R5	적절한 보호물의 크기	Template Blueruler 사용 보호물의 폭 In. 인물의 폭 In. 0- 보호물의 폭=< 인물의 폭 1- 보호물의 폭 > 인물의 폭	
R6	완전 무결한 보호물	0- 보호물이 손상됨 1- 보호물이 완전함	
R7	비우	0- 인물이 우비를 입고 있지 않음 1- 인물이 보호하는 우비를 입고 있음	
R8	비모자	0- 인물이 모자나 보호하는 것이 없음 1- 인물이 비모자나 머리를 보호하는 것이 있음	
R9	장화	0- 인물이 장화를 신고 있지 않음 1- 인물이 장화나 다른 보호하는 것을 신고 있음	

1-3. 집사부일체 그림검사 편



2. PITR 모델링



2-1. 객체 탐지 모델(Mask R-CNN)

YOLO v5

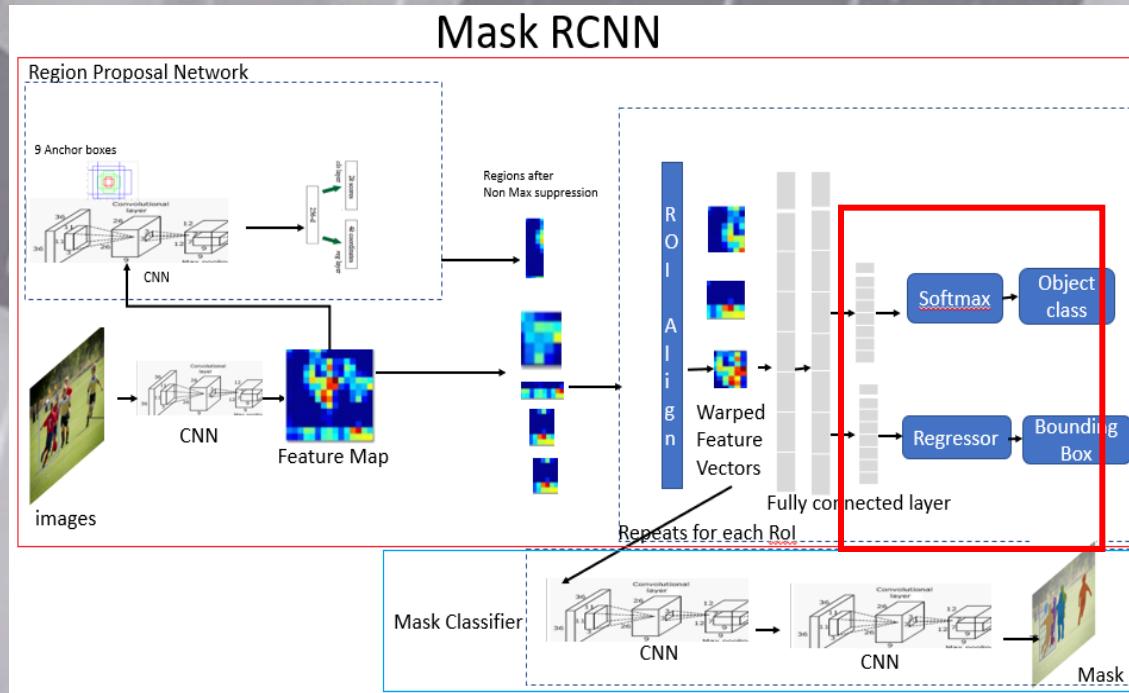
빠른 처리 속도

C++기반의 프레임 워크

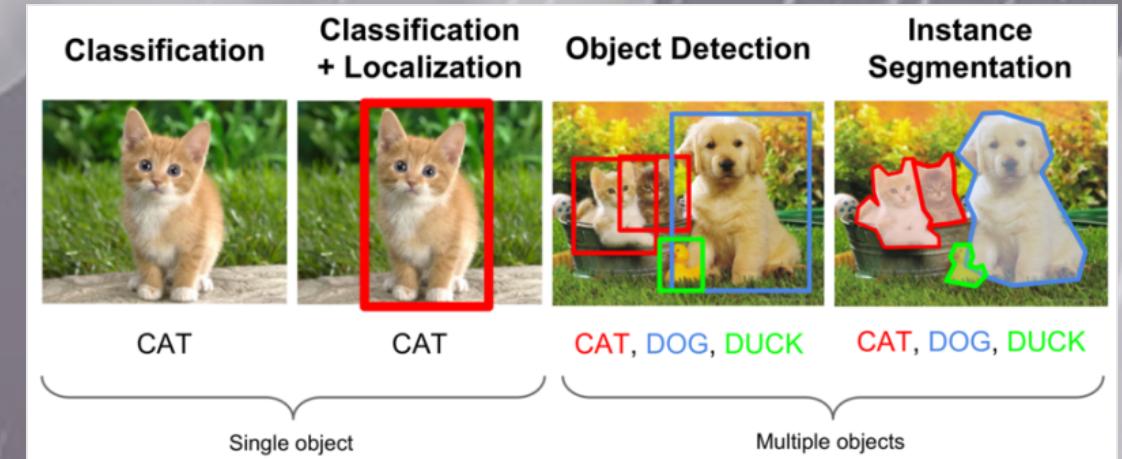
R-CNN 모델

YOLO 보다 속도는 느리지만 좋은 성능

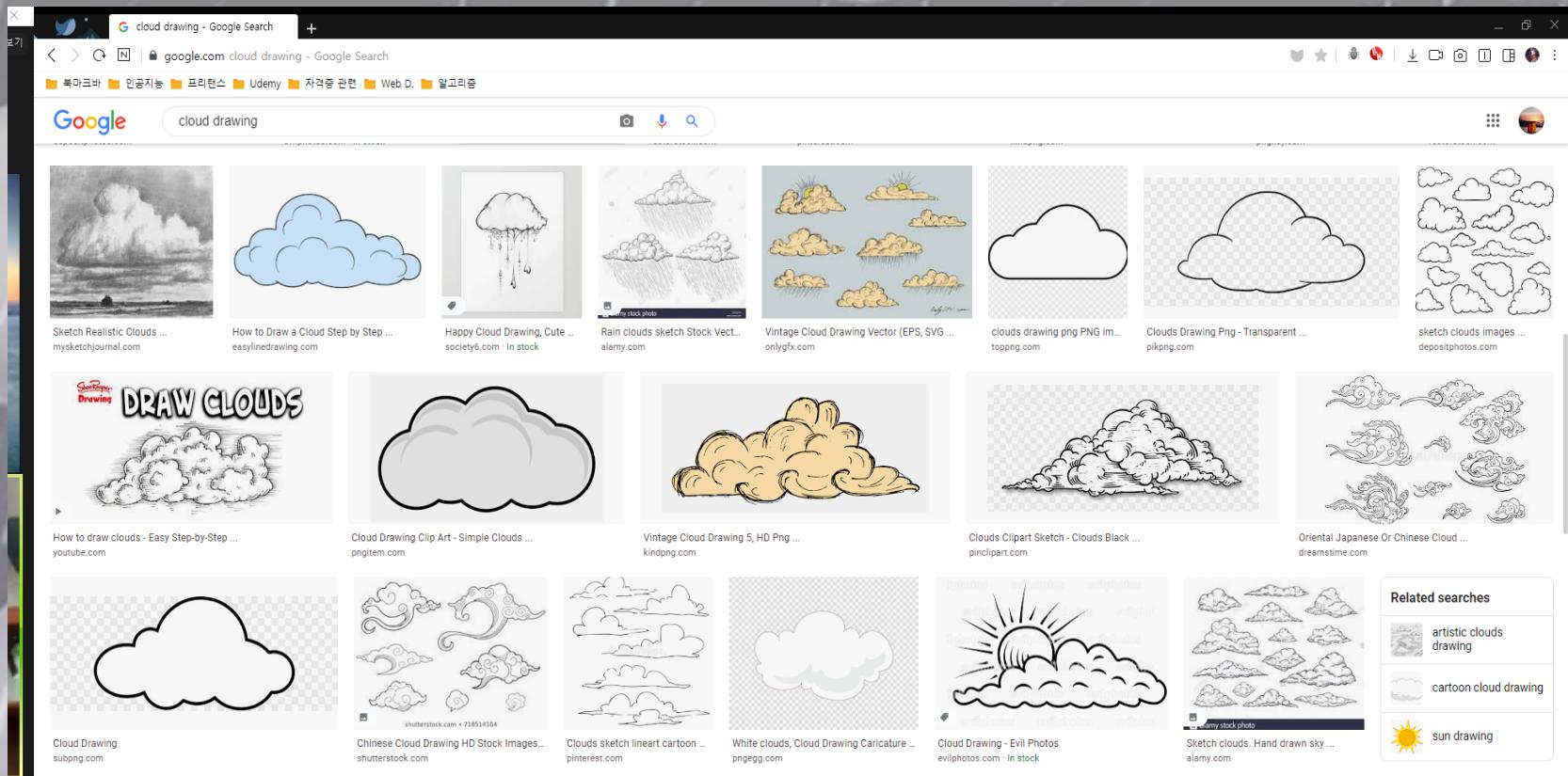
Python을 이용할 수 있는 환경을 제공



객체를 Masking 하는 Mask R-CNN을 사용



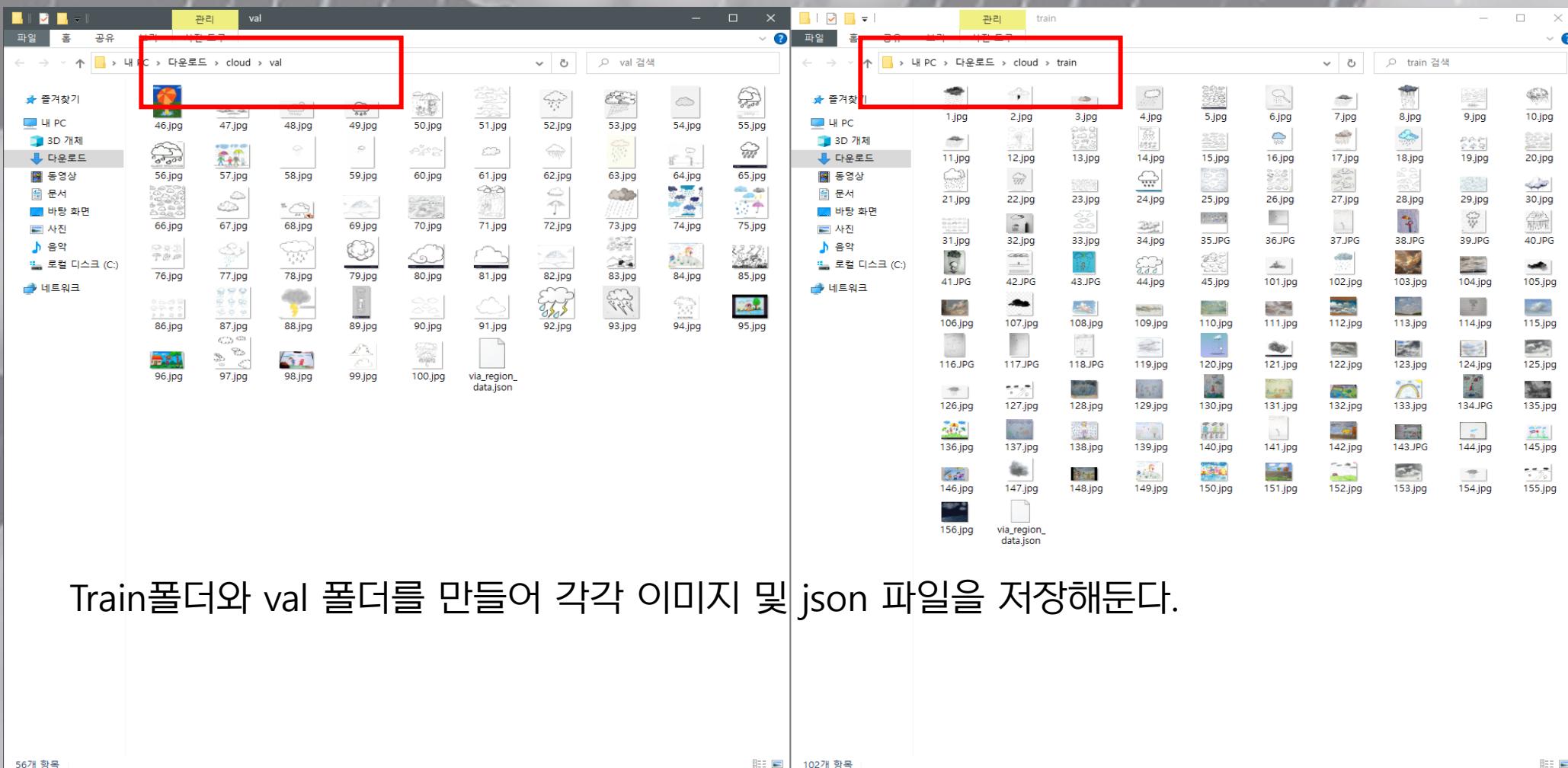
2-2. 레이터 수집 과정



물체 별 그림을 크롤링한 후 저장



물체의 선을 따준 후에 json 파일로 저장해준다.



Train폴더와 val 폴더를 만들어 각각 이미지 및 json 파일을 저장해둔다.

2-3. 학습 과정

MaskRCNNtrain_Umbrella.ipynb

파일 수정 보기 삽입 렌타일 도구 도움말 모든 변경사항이 저장됨

파일

[1] ! git clone https://github.com/matterport/Mask_RCNN.git

Cloning into 'Mask_RCNN'...
remote: Enumerating objects: 956, done.
remote: Total 956 (delta 0), reused 0 (delta 0), pack-reused 956
Receiving objects: 100% (956/956), 125.23 MiB / 39.96 MiB/s, done.
Resolving deltas: 100% (562/562), done.

[2] ! git clone https://github.com/AISangam/Mask-RCNN-bottle-training.git

Cloning into 'Mask-RCNN-bottle-training'...
remote: Total 97 (delta 0), reused 0 (delta 0), pack-reused 97
Unpacking objects: 100% (97/97), done.

[3] ! cd Mask_RCNN

/content/Mask_RCNN

[4] ! python setup.py install

WARNING:root:Fail to load requirements file, so using default ones
running install
running bdist_egg
running egg_info
creating mask_rcnn.egg-info
writing mask_rcnn.egg-info/PKG-INFO
writing dependency_links to mask_rcnn.egg-info/dependency_links.txt
writing top-level names to mask_rcnn.egg-info/top_level.txt
writing manifest file 'mask_rcnn.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
writing manifest file 'mask_rcnn.egg-info/SOURCES.txt'
Installing library code to build/bdist.linux-x86_64/egg
running install_egg_info
running build_egg
creating build_egg
creating build/lib
creating build/lib/arcnn
copying arcnn/utils.py > build/lib/arcnn
copying arcnn/visualize.py > build/lib/arcnn
copying build/lib/arcnn/parallel_model.py > build/bdist.linux-x86_64/egg/arcnn
copying arcnn/_init_.py > build/bdist.linux-x86_64/egg/arcnn
copying arcnn/model.py > build/lib/arcnn
copying arcnn/config.py > build/lib/arcnn
creating build/bdist.linux-x86_64
creating build/bdist.linux-x86_64/egg
copying build/bdist.linux-x86_64/egg/arcnn
copying build/lib/arcnn/utils.py > build/bdist.linux-x86_64/egg/arcnn
copying build/lib/arcnn/visualize.py > build/bdist.linux-x86_64/egg/arcnn
copying build/lib/arcnn/parallel_model.py > build/bdist.linux-x86_64/egg/arcnn
copying build/lib/arcnn/_init_.py > build/bdist.linux-x86_64/egg/arcnn
copying build/lib/arcnn/model.py > build/bdist.linux-x86_64/egg/arcnn
byte-compiling build/bdist.linux-x86_64/egg/arcnn/utils.py to utils.cpython-37.pyc
byte-compiling build/bdist.linux-x86_64/egg/arcnn/visualize.py to visualize.cpython-37.pyc
byte-compiling build/bdist.linux-x86_64/egg/arcnn/parallel_model.py to parallel_model.cpython-37.pyc
byte-compiling build/bdist.linux-x86_64/egg/arcnn/_init__.py to __init__.cpython-37.pyc
byte-compiling build/bdist.linux-x86_64/egg/arcnn/model.py to model.cpython-37.pyc
byte-compiling build/bdist.linux-x86_64/egg/arcnn/config.py to config.cpython-37.pyc
creation build/bdist.linux-x86_64/egg/arcnn/arcnn

디스크 35.93 GB 사용 가능

git clone으로 mask
rcnn을 드라이브에
가져오기

bottle.py X

```
56 # Configurations
57 #####
58 #
59
60 class CustomConfig(Config):
61     """Configuration for training on the toy dataset.
62     Derives from the base Config class and overrides some values.
63     """
64     # Give the configuration a recognizable name
65     NAME = "bottle umbrella"
66
67     # We use a GPU with 12GB memory, which can fit two images.
68     # Adjust down if you use a smaller GPU.
69     IMAGES_PER_GPU = 2
70
71     # Number of classes (including background)
72     NUM_CLASSES = 1 + 1 # Background + toy
73
74     # Number of training steps per epoch
75     STEPS_PER_EPOCH = 100
76
77     # Skip detections with < 90% confidence
78     DETECTION_MIN_CONFIDENCE = 0.9
79
80 #####
81 #
82 # Dataset
83 #####
84
85 class CustomDataset(utils.Dataset):
86
87     def load_custom(self, dataset_dir, subset):
88         """Load a subset of the bottle dataset.
89         dataset_dir: Root directory of the dataset.
90         subset: Subset to load: train or val
91         """
92
93         # Add classes. We have only one class to add.
94         self.add_class("bottle", 1, "bottle")
95
96         # Train or validation dataset?
97         assert subset in ["train", "val"]
98         dataset_dir = os.path.join(dataset_dir, subset)
99
100
101         # Load annotations
102         # VGG Image Annotator saves each image in the form:
103         # { 'filename': '28503151_5b5b7ec140_b.jpg',
104         #   'regions': [
105         #     '0': {
106         #       'region_attributes': {},
107         #       'shape_attributes': {
108         #         'all_points_x': [...],
109         #         'all_points_y': [...],
110         #         'name': 'polygon'},
111         #       ...
112         #       more regions ...
113     }
```

새로운 데이터를 학습시키기 위
해 .py 파일의 내용을 수정해줌

```

mrcnn_mask_conv3      (TimeDistributed)
mrcnn_mask_bn3        (TimeDistributed)
mrcnn_class_conv2     (TimeDistributed)
mrcnn_class_bn2       (TimeDistributed)
mrcnn_mask_conv4     (TimeDistributed)
mrcnn_mask_bn4       (TimeDistributed)
mrcnn_bbox_fc         (TimeDistributed)
mrcnn_mask_deconv    (TimeDistributed)
mrcnn_class_logits   (TimeDistributed)
mrcnn_mask            (TimeDistributed)

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/optimizers.py:793: The name tf.train.Optimiz
/usr/local/lib/python3.7/dist-packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning: Conve
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape."
/usr/local/lib/python3.7/dist-packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning: Conve
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape."
/usr/local/lib/python3.7/dist-packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning: Conve
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape."
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/tensorflow_backend.py:1033: The name
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/tensorflow_backend.py:1020: The name
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/tensorflow_backend.py:1020: The name

/usr/local/lib/python3.7/dist-packages/keras/engine/training_generator.py:49: UserWarning: Using a generator with
  UserWarning('Using a generator with `use_multiprocessing=True`')
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/callbacks.py:1122: The name tf.summary.merge
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/callbacks.py:1125: The name tf.summary.FileWriter

Epoch 1/10
2021-02-24 00:56:19,124112: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened d
2021-02-24 00:56:20,879155: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened d
100/100 [=====] - 168s/2s/step - loss: 1.5410 - rpn_class_loss: 0.0185 - rpn_bbox_loss:
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/callbacks.py:1265: The name tf.Summary is de

Epoch 2/10
100/100 [=====] - 116s 1s/step - loss: 0.8539 - rpn_class_loss: 0.0111 - rpn_bbox_loss:
Epoch 3/10
100/100 [=====] - 116s 1s/step - loss: 0.5744 - rpn_class_loss: 0.0085 - rpn_bbox_loss:
Epoch 4/10
100/100 [=====] - 116s 1s/step - loss: 0.4520 - rpn_class_loss: 0.0059 - rpn_bbox_loss:
Epoch 5/10
100/100 [=====] - 116s 1s/step - loss: 0.3608 - rpn_class_loss: 0.0064 - rpn_bbox_loss:
Epoch 6/10
100/100 [=====] - 116s 1s/step - loss: 0.3003 - rpn_class_loss: 0.0050 - rpn_bbox_loss:
Epoch 7/10
100/100 [=====] - 115s 1s/step - loss: 0.2647 - rpn_class_loss: 0.0055 - rpn_bbox_loss:
Epoch 8/10
100/100 [=====] - 116s 1s/step - loss: 0.2300 - rpn_class_loss: 0.0050 - rpn_bbox_loss:
Epoch 9/10
100/100 [=====] - 116s 1s/step - loss: 0.2113 - rpn_class_loss: 0.0038 - rpn_bbox_loss:
Epoch 10/10
100/100 [=====] - 116s 1s/step - loss: 0.1956 - rpn_class_loss: 0.0040 - rpn_bbox_loss:

```

10번의 에포크로 훈련을 진행

```

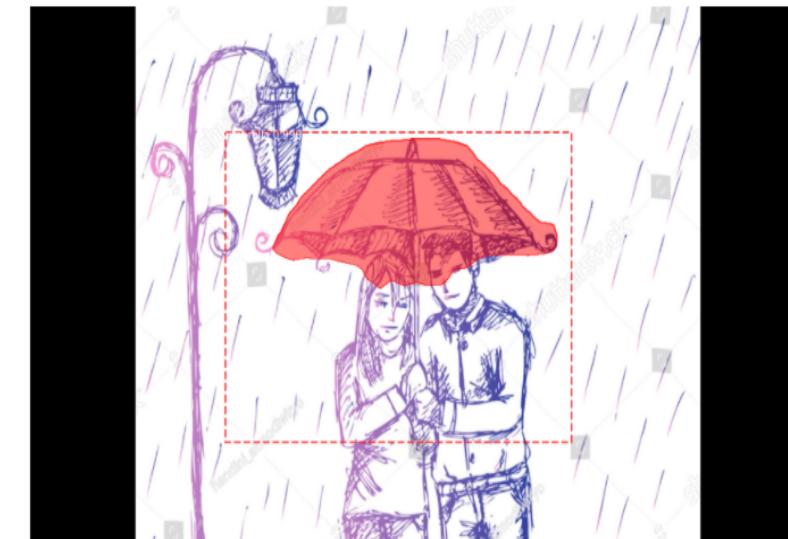
[22] image_id = random.choice(dataset.image_ids)
image, image_meta, gt_class_id, gt_bbox, gt_mask =#
  modellib.load_image_gt(dataset, config, image_id, use_mini_mask=False)
info = dataset.image_info[image_id]
print("image ID: {}, {}, {}".format(info["source"], info["id"], image_id))
dataset.image_reference(image_id))

# Run object detection
results = model.detect([image], verbose=1)

# Display results
ax = get_ax()
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                           dataset.class_names, r['scores'], ax=ax,
                           title="Predictions")
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)

image ID: umbrella,99.jpg (13) /content/drive/MyDrive/Colab Notebooks/mrcnn_trianing_model_umbrella/content/train_umbrella/dataset/val/99.jpg
Processing 1 images
image          shape: (1024, 1024, 3)    min: 0,00000 max: 255,00000 uint8
molded_images  shape: (1, 1024, 1024, 3)  min: -123,70000 max: 151,10000 float64
image_metas    shape: (1, 14)               min: 0,00000 max: 1024,00000 int64
anchors        shape: (1, 261888, 4)    min: -0,35390 max: 1,29134 float32
gt_class_id    shape: (1,)                min: 1,00000 max: 1,00000 int32
gt_bbox        shape: (1, 4)               min: 152,00000 max: 700,00000 int32
gt_mask        shape: (1024, 1024, 1)  min: 0,00000 max: 1,00000 bool
Predictions

```



3. 코드 설명 및 구현 화면

```
In [3]: import time  
# 그리기 전 지시사항 전달  
instr = 지시사항  
  
1) 용지에 빗속의 사람을 그리세요.  
2) 만화나 막대기 같은 사람이 아닌 완전한 사람을 그리세요.  
3) 그림을 그릴 때는 옆 사람의 그림을 보고 그리거나, 서로 의논하여 그리지 않고 혼자서 생각대로 자유롭게 그리세요.  
""  
  
print(instr)  
for x in range(3):  
    time.sleep(1)  
    print(".", end="")  
  
지시사항  
  
1) 용지에 빗속의 사람을 그리세요.  
2) 만화나 막대기 같은 사람이 아닌 완전한 사람을 그리세요.  
3) 그림을 그릴 때는 옆 사람의 그림을 보고 그리거나, 서로 의논하여 그리지 않고 혼자서 생각대로 자유롭게 그리세요.
```

피검자가 그림을 그리기 전
숙지해야 할 지시사항 전달

```
In [*]: # 그림 완료 후 추가질문  
  
print("\n")  
print("그림 그리기가 완료되었다면, 다음의 질문에 차례대로 답해주시기 바랍니다.")  
  
time.sleep(2)
```

```
q1 = input("1) 비는 얼마만큼 내리나요? : \n(많이, 보통, 적게)")  
q2 = input("2) 비에 얼마나 맞고 있나요?: \n(많이, 조금, 없음)")  
q3 = input("3) 그 사람의 기분은 어떻습니까? \n(좋다, 그저그렇다, 나쁘다) : ")  
  
# 답변내용 저장  
answer_result = [q1, q2, q3]
```

추가 질문에 답하도록 하여 피검자의 답변 내용을 리스트에 저장함. 이후 점수 산출에 사용될 예정.

```
그림 그리기가 완료되었다면, 다음의 질문에 차례대로 답해주시기 바랍니다.  
1) 비는 얼마만큼 내리나요? :  
(많이, 보통, 적게)보통  
2) 비에 얼마나 맞고 있나요?:  
(많이, 조금, 없음)
```

```
In [*]: import os  
import sys  
import random  
import math  
import re  
import time  
import numpy as np
```

필요 도구 & 모델 불러오기

```
1 import os  
2 import sys  
3 import random  
4 import math  
5 import re  
6 import time  
7 import numpy as np  
8 import tensorflow as tf  
9 import matplotlib  
10 import matplotlib.pyplot as plt  
11 import matplotlib.patches as patches  
12 from PIL import Image  
13 import warnings  
14 warnings.filterwarnings("ignore")  
15
```

```
16 # Import Mask RCNN  
17 from mrcnn import utils  
18 from mrcnn import visualize  
19 from mrcnn.visualize import display_images  
20 import mrcnn.model as modellib  
21 from mrcnn.model import log  
22
```

```
23 # 예측 모델 임포트  
24 import umbrella  
25 import person  
26 import cloud  
27 import puddle  
28 %matplotlib inline
```

필요한 도구들 가져오기

Mask_RCNN 가져오기

물체별로 학습시킨 예측 모델을 불러온다.
총 네 개의 모델 존재.

예를 들어 cloud 모델은 그림 내에서 구름을 판단하기 위해 학습됨

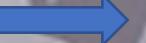
예측 함수 정의 & 예측 결과 저장

```
1 def Custom_predict(custom,custom_name, IMAGE): # predict 모델을 만드는 것
2
3     config = custom.CustomConfig()
4
5     class InferenceConfig(config.__class__):
6         # Run detection on one image at a time
7         GPU_COUNT = 1
8         IMAGES_PER_GPU = 1
9         config = InferenceConfig()
10        TEST_MODE = "inference"
11        DEVICE = "/cpu:0"
12        with tf.device(DEVICE):
13            model = modellib.MaskRCNN(mode="inference", model_dir="", config=config)
14            class_names = [custom_name]
15            weights_path = custom_name+'.h5' # 우리가 한 학습폴더를 불러와서
16            model.load_weights(weights_path, by_name=True)
17
18            results = model.detect([image], verbose=0)
19            r = results[0]
20            r["class_names"] = ["",custom_name] # 딕셔너리에 추가적으로 커스텀 네임을 입력할 수 있도록 만들어 줬다
21
22            return r
```

특정 물체를 인식하기 위해 학습시킨 모델의 가중치 불러오기

모델의 예측결과를 딕셔너리로 가져와 딕셔너리에 물체이름을 추가하여 리턴해줌

```
1 person_result = Custom_predict(person,'person',image)
2 cloud_result = Custom_predict(cloud,'cloud',image)
3 umbrella_result = Custom_predict(umbrella,'umbrella',image)
4 puddle_result = Custom_predict(puddle,'puddle',image)
```



하나의 그림에 대한 각 모델별 예측 결과를 저장 예를 들어 person_result에는 사람을 예측한 결과 값이 저장됨.

물체의 위치와 크기정보를 제공하는 함수 정의

```
1 def status(result, image):
2     # 객체가 없으면 없음으로 반환
3     if not len(result["class_ids"]):
4         return "없음"
5     elif len(result["class_ids"])>1:
6         return "2개 이상"
7
8     mask = result["masks"]
9
10    status = {}
11
12    y_min , y_max , x_min, x_max = y_x(mask)
13
14    y_cen = (y_min+y_max)/2
15    x_cen = (x_min+x_max)/2
16    status["위치 "] = position(y_cen, x_cen)
17
18    status["크기 "] = size(y_min, y_max , x_min,x_max)
19
20
21    return status
```

모델의 예측결과와 그림을 넣으면
물체의 위치와 크기를 딕셔너리 형태로
돌려주는 함수

```
40 def size(y_min, y_max , x_min,x_max):
41     y,x,_ = image.shape
42     y_size = y_max-y_min
43     x_size = x_max-x_min
44
45     if y_size*x_size>(y*x)/8:
46         return "크다"
47     elif y_size*x_size>(y*x)/16:
48         return "보통"
49     else :
50         return "작다"
51
52 def y_x(mask):
53     y_min = np.where(mask)[0].min()
54     y_max = np.where(mask)[0].max()
55     x_min = np.where(mask)[1].min()
56     x_max= np.where(mask)[1].max()
57     return y_min , y_max , x_min, x_max
```

y_x 함수를 통해 물체의 좌표값을
산출하고 이를 바탕으로 물체의 크
기를 판단하는 함수를 정의

```
22 def position(y_cen,x_cen):
23
24     y,x,_ = image.shape
25     position = []
26
27     if x_cen>x*2/3:
28         position.append("오른쪽")
29     elif x_cen>x*1/3:
30         position.append("가운데")
31     else:
32         position.append("왼쪽")
33
34     if y_cen>y*2/3:
35         position.append("아래")
36     elif y_cen>y*1/3:
37         position.append("가운데")
38     else:
39         position.append("위 ")
40
41     return position
```

물체의 y좌표의 중심점과 x좌표의
중심점을 입력값으로 받아
물체의 위치를 리스트 형태로 반
환해주는 함수를 정의

이 세 가지 함수는 이후 점수를 산출하는 함수를 만들 때 활용!

스트레스 점수를 산출하는 함수 정의

→ 스트레스 척도의 항목 중 일부를 가져와, 항목별로 점수를 리턴하는 함수를 만들어 줌.

```
1 # 비의 양
2 def S1(answer):
3     if answer == "많이":
4         return 3
5     elif answer == "보통":
6         return 2
7     return 1
8 # 비와 사람의 접촉
9 def S5(answer) :
10    if answer == "많이 접촉":
11        return 3
12    elif answer == "조금 접촉":
13        return 2
14    return 1
15 # 구름의 주
16 def S7(cloud_result):
17     num = len(cloud_result["class_ids"])
18     if num==0:
19         return 1
20     elif num<3:
21         return 2
22     return 3
23 # 구름의 면적
24 def S8(cloud_result):
25     if not len(cloud_result):
```

앞서 저장했던 질문에 대한 답변을 활용하여 점수를 매김.

```
46 # 웅덩이 면적
47 def S13(puddle_result):
48     if not len(puddle_result):
49         return 1
50     large = 0
51     for i in range(len(puddle_result["class_ids"])):
52         _,_,a,b = y_x(puddle_result["masks"][:, :, i])
53         large+= (b-a)
54     if large<x/10:
55         return 1
56     elif large<x/5:
57         return 2
58     return 3
```

특정 물체의 개수와 면적을 판단하여 점수를 매김.

```
59 # 사람과 웅덩이 관계
60 def S14(puddle_result, person_result):
61     if not len(puddle_result["class_ids"]):
62         return 1
63     person_yx = y_x(person_result["masks"])
64     for i in range(len(puddle_result["class_ids"])):
65         yx = y_x(puddle_result["masks"][:, :, i])
66
67         if yx[2]<person_yx[2]<yx[3] or yx[2]<person_yx[3]<yx[3]:
68             return 3
69
70     return 2
```

물체와 사람 간의 관계를 판단하여 점수를 매기는 함수. 사람의 양 끝 x좌표가 웅덩이의 좌표 안에 포함되는가를 판단하여 점수를 매김.

자원 점수를 산출하는 함수 정의

→ 자원 척도의 항목 중 일부를 가져와, 항목별로 점수를 계산하는 함수를 만들어 줌.

```
1 # 직접보호를 개수
2 def R1(umbrella_result):
3     if not len(umbrella_result["class_ids"]):
4         return 1
5     elif len(umbrella_result["class_ids"]) == 1:
6         return 2
7     return 3
8 # 보호물의 적절성
9 def R4(umbrella_result, person_result):
10
11     if (not len(umbrella_result["class_ids"])) or (not len(person_result["class_ids"])):
12         return 1
13     else:
14         umbrella_y_min = y_x(umbrella_result['masks'])[0]
15         person_y_min = y_x(person_result['masks'])[0]
16         # 우산이 사람위에서 비를 막아주는 경우
17         if umbrella_y_min <= person_y_min:
18             return 3
19         else:
20             return 2
21 # 내용의 기준
22 def R7(answer):
23     if answer == "좋다":
24         return 3
25     elif answer == "그저그렇다":
26         return 2
27     return 1
```

우산이 비로부터 사람을 적절하게 보호하고 있는가를 판단하여 점수를 매기는 함수.

사람과 우산의 y좌표를 비교하여 판단한다.

```
28 # 인물의 크기
29 def R8(person_result):
30     y,x,_ = image.shape
31     a,b,c,d = y_x(person_result["masks"])
32
33
34     if size(a,b,c,d) == "크다":
35         return 3
36     elif size(a,b,c,d) == "보통":
37         return 2
38     return 1
39 # 인물의 위치
40 def R9(person_result):
41     y,x,_ = image.shape
42     y_min, y_max, x_min, x_max = y_x(person_result['masks'])
43     y_cen = (y_min+y_max)/2
44     x_cen = (x_min+x_max)/2
45
46 # 가운데
47     if ((x_cen > x*1/5) and (x_cen < x*1/5)) and ((y_cen > y*1/5) and (y_cen < y*1/5)):
48         return 3
49     # 중심이탈
50     elif ((x_cen > x*1/5) or (x_cen < x*1/5)) and ((y_cen > y*1/5) and (y_cen < y*1/5)):
51         return 2
52     else:
53         return 1
```

인물의 중심좌표가 그림의 특정 비율 내에 존재하는가를 판단하는 함수. 그림 내에서의 인물의 위치를 판단하여 점수를 매김.

앞서 정의한 함수를 활용해 인물의 크기를 판단하여 점수를 매김.

스트레스와 자원점수 합산 및 최종 대처능력점수 출력

```
1 def StressScore(image):
2
3     score = 0
4     score+= S1(answer_result[0])
5     score+= S5(answer_result[1])
6     score+= S7(cloud_result)
7     score+= S8(cloud_result)
8     score+= S12(puddle_result)
9     score+= S13(puddle_result)
10    score+= S14(puddle_result , person_result)
11    return score
12
13 def ResourceScore(image):
14     score=0
15     score+= R1(umbrella_result)
16     score+= R4(umbrella_result, person_result)
17     score+= R7(answer_result[2])
18     score+= R8(person_result)
19     score+= R9(person_result)
20
21     return score
```

앞서 항목별로 만들었던 함수들의 결과값들을 모두 합산해서, 스트레스 점수와 자원 점수를 계산하는 함수를 만들어 줌.

대처능력 = 자원점수 - 스트레스점수

```
1 def PITRScore(image):
2
3     return ResourceScore(image) - StressScore(image) + 3
```

1 PITRScore(image)

0

대처능력 산출함수를 정의하고
최종 대처능력 점수를 출력해본다.

그림에 대한 최종 결과 보고서 출력

```
1 # 해석결과 출력함수 정의
2 def print_report(image):
3     pitr_score = PITRScore(image)
4     stress_score = StressScore(image)
5     resource_score = ResourceScore(image)
6
7     print(Final_Report, end="\n\n")
8     print("- 대처능력점수 해석", end="\n\n")
9     print('나의 스트레스 점수는 {}점이고, 자원 점수는 {}점이며 대처능력점수는 {}점입니다.'.format(stress_score,
10
11     if pitr_score > 0:
12         print(doc1)
13     elif pitr_score < 0:
14         print(doc2)
15         if pitr_score <= -5:
16             print("\n")
17             print(doc3)
18     else:
19         print(doc4)
```

점수가

```
1 print_report(image)
<최종 결과 보고서>
- 그림 내용 해석
비는 '스트레스'를 가능하는 지표로 빗줄기의 양은 스트레스의 양을 나타냅니다.
그림 속에 비가 없거나 빗물이 아주 적은 경우는 내담자가 스트레스에 무딘 경우라는 것을 나타내고,
반대로 빗줄기의 양이 많다면 받고 있는 스트레스의 크기도 그만큼 많다는 것을 나타냅니다.

비에 대한 대응은 스트레스에 대한 대응을 뜻하고 우산을 쓰거나 간접적인 보호물(나무, 집 등) 아래에 숨어
비에 맞지 않고 있다면 내담자가 스트레스에 적절히 대응하고 있다고 볼 수 있습니다.
그러나 우산이 지나치게 클 경우 스트레스에 대처하는데 과도한 에너지를 사용한다고 볼 수 있습니다.

한편, 그림 속 사람의 크기는 자신의 자아에 대한 크기를 나타냅니다.
가로등을 그린 경우, 애정과 자지에 대한 관심을 드러낸다고 볼 수 있습니다.

만약 천둥이나 번개를 그릴 경우, 지금 상당한 스트레스에 직면해 있음을 나타냅니다.
*****
- 대처능력점수 해석
나의 스트레스 점수는 15점이고, 자원 점수는 12점이며 대처능력점수는 0점입니다.

현재 자신이 받는 스트레스와 대처자원의 정도가 균형을 이루고 있는 상태입니다.
어떻 부면 문제가 없는 상태로 부의 수 있지만 현재 상황에서 스트레스가 조금마 더 많아지면 어제든지 스트레스로 이해 할되어지 수 있습
```

미리 정리해둔 해석 코멘트를 참고하고, 최종 대처능력 점수의 크기를 판단하여 최종 결과 보고서를 출력하는 함수를 정의.

테스트 그림에 대한 최종 결과보고서 출력.

Mask된 이미지 출력함수 정의

```
def get_ax(rows=1, cols=1, size=16):
    _, ax = plt.subplots(rows, cols, figsize=(size*cols, size*rows))
    return ax

def Custom_Mask(r1,r2,r3,r4, IMAGE):
    # Load validation dataset

    ax = get_ax(1)
    class_names = ["", "custom"]
    #Must call before using the dataset

    visualize.display_instances(image, r1["rois"], r1['masks'], r1['class_ids'],
                                r1["class_names"], r1['scores'], ax=ax,
                                title="Predictions")
    visualize.display_instances(image, r2["rois"], r2['masks'], r2['class_ids'],
                                r2["class_names"], r2['scores'], ax=ax,
                                title="Predictions")
    visualize.display_instances(image, r3["rois"], r3['masks'], r3['class_ids'],
                                r3["class_names"], r3['scores'], ax=ax,
                                title="Predictions")
    visualize.display_instances(image, r4["rois"], r4['masks'], r4['class_ids'],
                                r4["class_names"], r4['scores'], ax=ax,
                                title="Predictions")
```

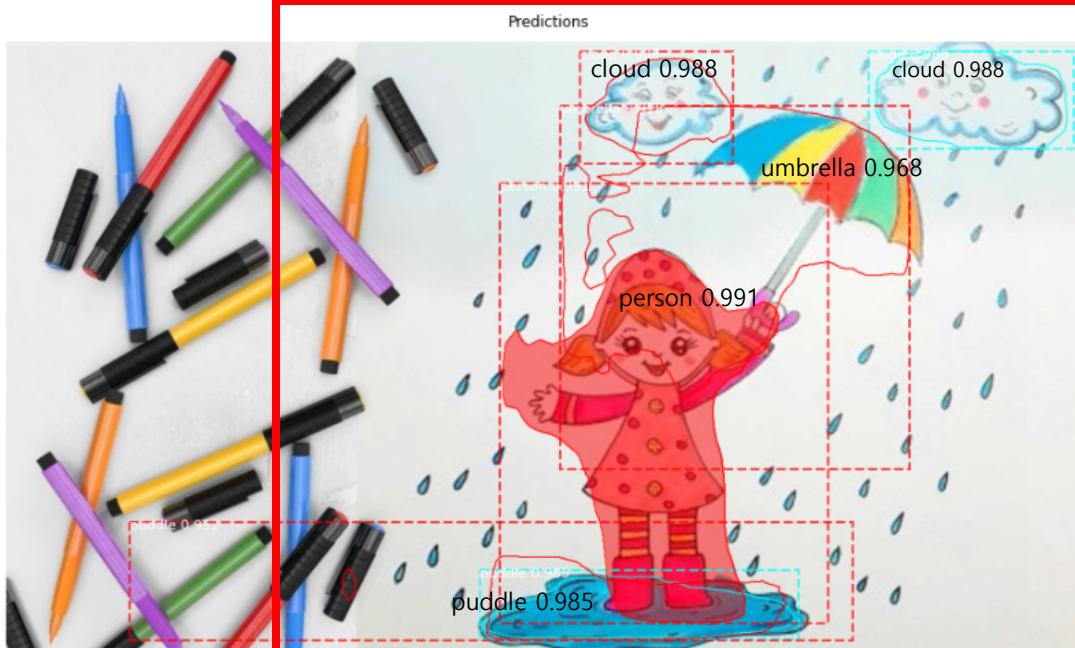
이미지가 출력될 틀을 만들어주는 함수

이미지 출력함수를 정의한다.
함수의 입력값으로 여러 모델의 결과값과 이미지 객체를 넣어주면, 물체별로 인식한 결과를 한 번에 보여준다.

3-2. 테스트 진행

```
title="Predictions")  
visualize.display_instances(image, r3["rois"], r3["masks"], r3["class_ids"],  
    r3["class_names"], r3["scores"], ax=ax,  
    title="Predictions")  
visualize.display_instances(image, r4["rois"], r4["masks"], r4["class_ids"],  
    r4["class_names"], r4["scores"], ax=ax,  
    title="Predictions")
```

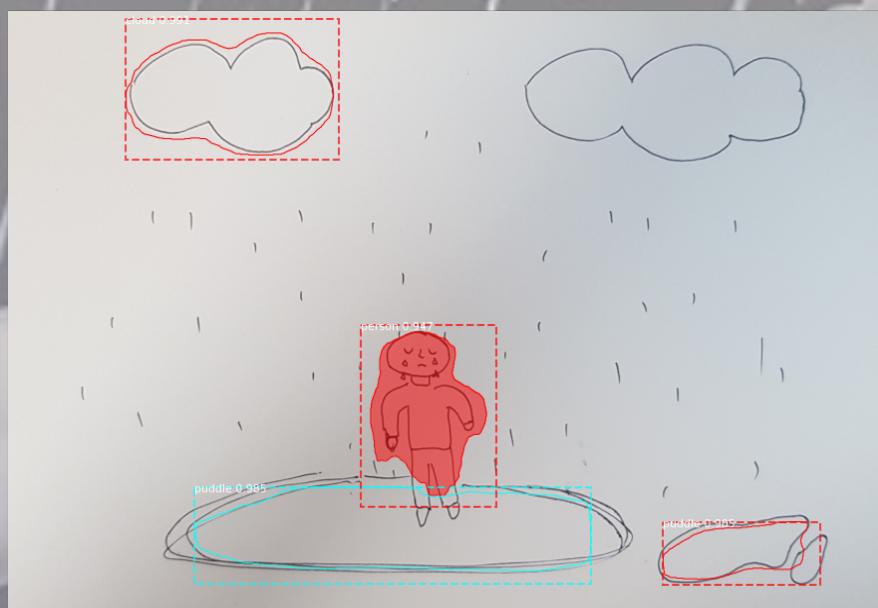
In [20]: Custom_Mask(cloud_result,umbrella_result,puddle_result,person_result, image)



앞서 만든 함수로
테스트 진행

기존에 존재하는
이미지를 가져와
물체 예측

실제 그림을 그려보고 mask된 이미지와 이에 따른 해석 결과를 출력해보았다.



- 대처능력점수 해석

나의 스트레스 점수는 17점이고, 자원 점수는 8점이며 대처능력점수는 -9점입니다.

현재 자신이 가진 스트레스에 대해 적절하게 대처하지 못하고 있습니다.

주변의 스트레스들에 비해 그에 대처할 수 있는 자원이 부족한 상황입니다.

대처자원이 부족하다 보니, 자신에게 스트레스 주는 일이 많다면 이를 잘 이겨내지 못하고 위축되거나 불안해하는 경향을 보입니다.

이러한 상황이 계속된다면 우울증에 빠질 수도 있으니 주의해야 합니다.

평소에 본인이 스트레스를 받을 때 대처할 수 있는 여러 방법들을 찾고, 개발하려는 노력이 필요합니다.

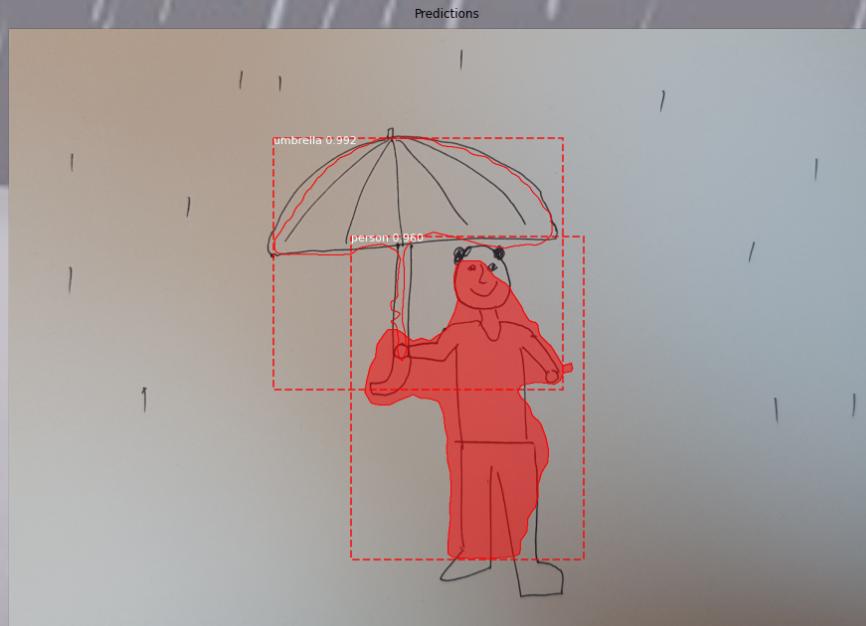
스트레스가 높은 집단을 대상으로 점수를 측정했을 때, 대처능력의 평균점수는 -5점 정도로 나옵니다.

현재 본인의 대처능력점수는 -5점 이하로, 겪고 있는 스트레스의 양이 상당히 많다고 해석될 수 있습니다.

많은 스트레스에 대해 적절하게 대처하지 못하고 있다는 뜻도 있겠지요.

이러한 상태가 오랜 기간 지속되었을 경우, 마음의 병이 생길 수 있습니다.

따라서 실제로 많은 스트레스로 인해 힘든 상황이라면, 까운 상담소나 병원에 찾아가 상담을 받아 보는 것이 좋겠습니다.



- 대처능력점수 해석

나의 스트레스 점수는 9점이고, 자원 점수는 14점이며 대처능력점수는 5점입니다.

현재 자신이 가진 스트레스에 대해 적절하게 대처하고 있습니다.

스트레스 점수가 높더라도 대처능력 점수가 높다면, 그만큼 본인이 스트레스에 대처하는 능력이 좋다고 볼 수 있습니다.

대처능력이 좋은 편이기 때문에 스트레스를 받는 일이 많아도 금방 잘 이겨내는 경우가 많습니다.

스트레스 상황이 생겼을 때 이를 회피하지 않고 문제에 부딪혀 이를 해결하려는 의지력도 가지고 있습니다.

자신의 미래를 낙관적으로 생각하는 경향이 있고, 자신에게 주어진 일을 성공적으로 수행할 수 있다는 믿음을 가리키는 '자기 효능감'이 높은 편입니다.

자신만의 스트레스 극복 & 해소 방법을 주변 사람들에게 공유해주는 것은 어떨까요?

4. 한계점

- 시간 관계상 생략된 데이터 (ex 집, 가로등, 얼굴표정 등의 평가 요소들)
- 그림의 전체적인 느낌, 분위기, 그린 순서, 상세한 내용 등을 내담자와 대화를 통해 파악하지 못함

5. 참고자료

- 인터넷 사이트
 - Github, https://github.com/matterport/Mask_RCNN
 - 티스토리, <https://hansonminlearning.tistory.com/41>
- 단행본
 - 박영숙 외 6인(2019). **현대 심리평가의 이해와 활용**. 학지사
 - 김병철 외 2인(2014). **그림과 심리진단**. 양서원
 - 최외선 외 3인(2006). **미술치료기법**. 학지사
- 논문
 - Lack, H. S.(1996). The Person-In-The-Rain projective drawing as a measure of children's coping capacity, Unpublished doctoral dissertation, the California school of psychology at Alameda.
 - 김갑숙(2013). 청소년의 정신건강 선별도구로서의 빗속의 사람 그림 활용. **Korean Journal of Counseling**, 14(3)
 - 김우민, 이은주(2020). 병원전산실 종사자의 직무스트레스 및 대처 방식과 '빗속의 사람 그림검사' 반응특성 연구. **미술치료연구**, 27(6)
 - 김수진, 한경아(2019). 대학생의 취업스트레스와 진로결정 자기효능감 수준에 따른 빗속의 사람(PITR) 그림검사 반응특성. **미술과 교육**, 20(1)
 - 김미현, 이근매 (2014). 소방공무원의 직무스트레스 정도에 따른 '빗속의 사람' 그림 반응특성 연구. **임상미술심리연구**, 4(2)
 - 정길수(2012). 대학생의 빗속의 사람그림 반응특성과 스트레스 대처방식. **미술치료연구**, 19(5)



모두 수고하셨습니다!