

# Mask R-CNN을 기반으로 한 태양광 패널을 설치할 수 있는 옥상 면적 계산 알고리즘

제태성(G202245008), 선석근(G20224302), 지성우(G202132007), 노소영(G202236002)

## 1. 연구의 배경 및 목적

본 연구는 위성사진을 활용하여 건물 옥상 중 태양광 패널을 설치할 수 있는 면적을 계산하는 모델을 설계한다. 이를 통해 생산 가능한 태양광 패널 전력 생산량을 예측하고, 전기 사용량과 비교하여 필요한 생산량 추이를 알아본다.

## 2. 연구의 범위 및 방법

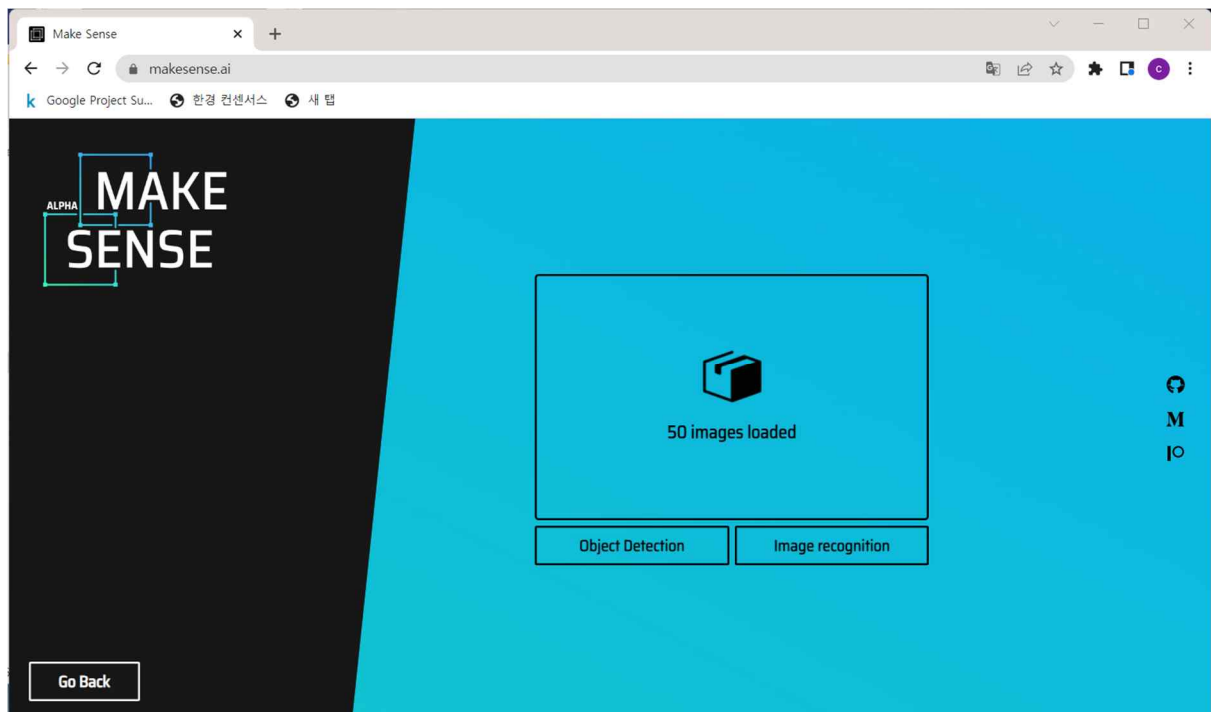
이를 위해 서울시 동대문구를 공간적 범위로 모델링을 진행하였으며, 구득 가능한 위성사진 기준 시기인 2021년을 연구의 시간적 범위로 설정하였다. 연구의 분석 단계는 다음과 같다. 먼저 서울시 동대문구 위성사진을 구득한 뒤 이를 임의 크기로 나누어 데이터 라벨링을 진행하였다. 이미 지 데이터 라벨링의 경우 오픈소스 프로젝트를 활용하여 태양광 패널의 설치가 가능할 것으로 판단되는 옥상 영역에 마스크(mask)와 주석(annotation)을 설정하였다. 이후 Mask R-CNN(Mask Region-based Convolutional Neural Network)으로 태양광 패널 설치 가능 옥상 면적 계산 모델을 설계하기 위해 해당 데이터셋을 이용해 옥상 형태를 학습시켰다. 이때 학습된 모델은 Mask R-CNN을 기반으로 태양광 패널 설치 가능 옥상 영역을 인식하고 해당 영역에 마스크(masking)를 수행한다. 모델의 학습이 종료된 이후 실측 면적과 모델이 계산한 면적의 차이를 통해 성능을 평가하였으며, 이후 모델이 계산한 마스크의 픽셀 면적을 계산하는 알고리즘을 통해 최종적으로 태양광 패널 설치가 가능한 옥상 영역의 면적을 계산하는 모델을 완성하였다. 임의의 지역에 대한 위성사진에 해당 모델을 적용하여 태양광 패널 설치 가능 옥상 면적을 계산한 뒤, 설치 가능한 태양광 패널을 모두 설치하였을 때 예상 전력 절약 크기(태양광 패널 전력 생산량)를 계산함으로써 실제 현실에서 해당 모델의 활용 가능성을 보였다.

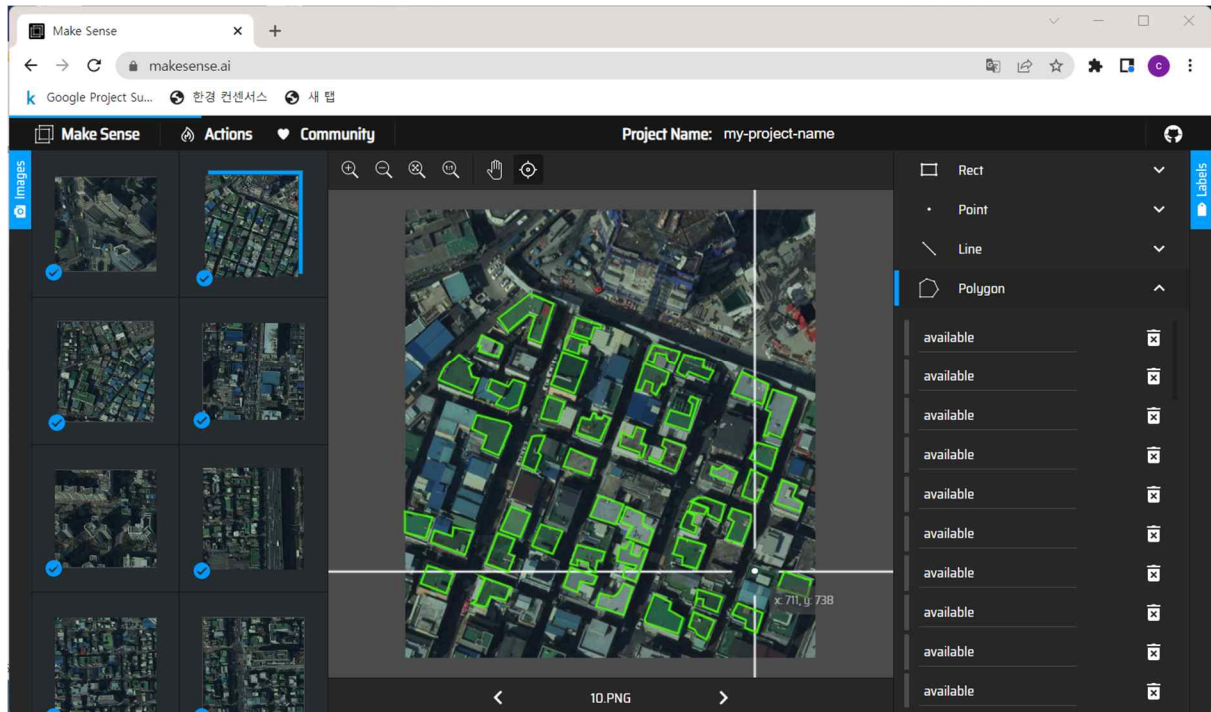
## 3. 분석데이터

모델 설계를 위해 사용한 데이터는 국토정보플랫폼 국토정보맵의 항공사진 데이터이다. 2021년 기준 서울특별시 동대문구의 항공사진을 구득하여 총 50개의 세부 이미지로 잘라서 사용하였다.

학습을 위해서 모델을 통해 감지해야 하는 물체의 외곽선 정보(Ground Truth, GT) 설정이 필요하다. 태양광 패널은 옥상(평지붕)형과 경사지붕(박공지붕)형으로 나뉘는데, 경사지붕형 패널의 경우 패널 설치 면의 경사각 등 고려해야 할 사항이 옥상형보다 많다. 그러나 데이터 구득의 한계로 패널 설치 면의 경사각 등 고려 사항에 대한 정보가 부재해 **본 연구는 옥상형 패널로 분석 영역을 한정하여 구득한 항공사진에서 옥상(평지붕)에 해당하는 것으로 판단되는 영역을 대상으로 Piotr Skalski가 제공하는 오픈소스 프로젝트(Make Sense)를 사용해 외곽선 정보를 Tagging 함으로써 데이터 전처리**를 진행하였다. 처음에는 이미지 라벨링을 수작업으로 진행해야 한다는 부분에서 모델의 신뢰도에 대해 큰 의심을 했는데, 라벨링을 직접 하는 과정에서 위성 사진 중에 어느 부분이 옥상이고 아닌지, 태양광 패널을 설치할 수 있는 공간인지 아닌지 등을 판단하는 것이 육안으로도 판단하기도 애매한 곳들이 많아서 수작업을 할 수밖에 없겠다고 생각했다. 그래도 여전히 신뢰도에 대한 의심은 남아있었고, 또한 육안으로 판단하기도 어렵다는 것은 사용자가 라벨링 단계에서 판단의 오류가 있을 수도 있다는 생각이 들었다. 그렇다면 **알고리즘의 논리나 성능과는 별개로 모델도 학습에 한계가 있다는 의미가 아닌지 의문이 추가**로 생기기도 했다.

초기에는 선행연구를 참고하여 이미지에서 건물의 옥상 및 지붕으로 판단되는 영역에 모두 마스크를 설정하였는데, 라벨링 작업을 하면서 이미 태양광이 설치되어 있는 부분이나 탈부착이 어려운 설치물 등이 있는 부분이 계속해서 포함되는 것을 보고 이 데이터를 기반으로 분석을 진행할 경우 나중에 진행할 태양광 패널 신규 설치 면적과 예상 전력 생산량 계산 등에 현실성이 다소 떨어질 것이라고 생각했다. 그래서 **안테나 등과 같은 이동이 어려운 물체와 이미 설치되어 있는 태양광 패널 등은 GT 내에 포함되지 않도록 설정해서 실제 설치 가능한 면적을 계산하는 모델을 설계하고자 했다.**

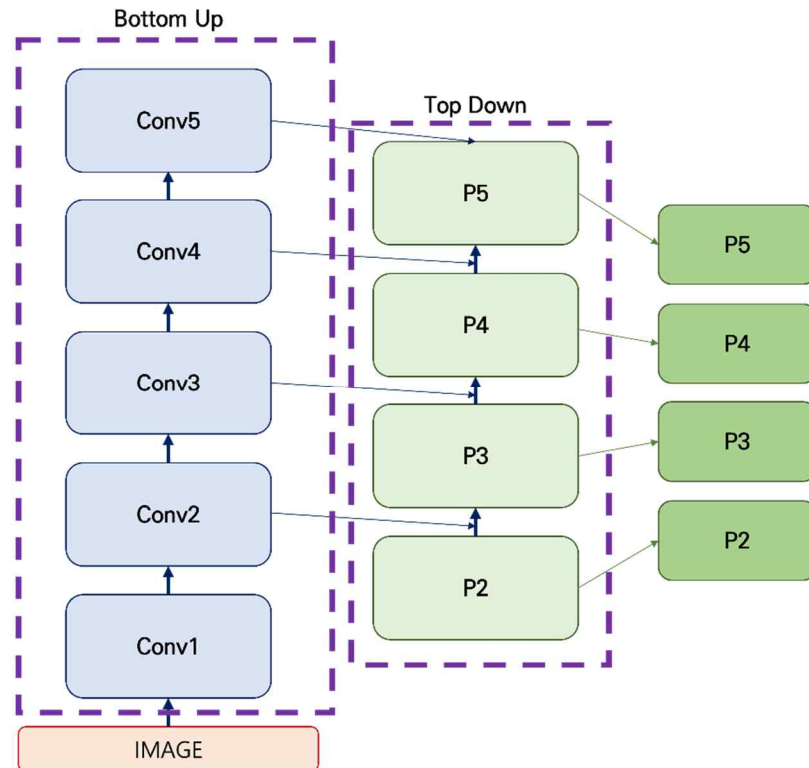




#### 4. 태양광 패널 설치 가능 영역 인식 모델

모델 구조는 다음 그림과 같다. 먼저, ResNet-101을 Backbone으로 하는 FPN(Feature Pyramid Network) 학습 구조를 사용한다. ResNet-101은 101개의 Convolution Layer로 이루어졌다. 이미지 분류 문제의 경우 기존 Neural Net의 학습 목적과는 다르게 입력값과 타깃값을 동일하게 매핑해야 한다. 따라서 결과적으로 입력값에 대응하는 출력값과 입력값의 차이를 최소화하는 것을 목적으로 학습을 진행한다. 결과적으로 ResNet-101은 층(Layer)의 출력값( $F(x)$ )에 입력값( $x$ )을 더하는 구조를 통해 학습의 정확도를 높인다.

FPN은 위아래로 연결된 피라미드 구조로, 마지막 Layer의 Feature map에서 이전의 Feature map을 더함으로써 이전 정보를 유지한다. Faster R-CNN의 경우에는 Backbone의 최종 Feature map에서 바로 ROI를 생성하기 때문에 이전 정보의 부재로 다양한 크기의 객체에 대응하는 데 한계가 있다. 그러나 FPN은 이와 달리 이전 정보가 모두 유지되는 구조기 때문에 각 Feature map 크기에 따라 적절한 scale의 anchor를 생성해서 다양한 크기의 객체에 대한 대응력이 높다. 학습 설정 값은 Step 500, Epoch 5로 설정하여 학습을 진행하였으며, 최종 모델은 입력 데이터(이미지)에 대해 바운딩 박스(bounding box)와 주석(class id), 마스크(mask)를 output으로 한다.



```
# Stage 1
x = KL.ZeroPadding2D((3, 3))(input_image)
x = KL.Conv2D(64, (7, 7), strides=(2, 2), name='conv1', use_bias=True)(x)
x = BatchNorm(name='bn_conv1')(x, training=train_bn)
x = KL.Activation('relu')(x)
C1 = x = KL.MaxPooling2D((3, 3), strides=(2, 2), padding="same")(x)

# Stage 2
x = conv_block(x, 3, [64, 64, 256], stage=2, block='a', strides=(1, 1), train_bn=train_bn)
x = identity_block(x, 3, [64, 64, 256], stage=2, block='b', train_bn=train_bn)
C2 = x = identity_block(x, 3, [64, 64, 256], stage=2, block='c', train_bn=train_bn)

# Stage 3
x = conv_block(x, 3, [128, 128, 512], stage=3, block='a', train_bn=train_bn)
x = identity_block(x, 3, [128, 128, 512], stage=3, block='b', train_bn=train_bn)
x = identity_block(x, 3, [128, 128, 512], stage=3, block='c', train_bn=train_bn)
C3 = x = identity_block(x, 3, [128, 128, 512], stage=3, block='d', train_bn=train_bn)

# Stage 4
x = conv_block(x, 3, [256, 256, 1024], stage=4, block='a', train_bn=train_bn)
block_count = {"resnet50": 5, "resnet101": 22}[architecture]
for i in range(block_count):
    x = identity_block(x, 3, [256, 256, 1024], stage=4, block=chr(98 + i), train_bn=train_bn)
C4 = x

# Stage 5
if stage5:
    x = conv_block(x, 3, [512, 512, 2048], stage=5, block='a', train_bn=train_bn)
    x = identity_block(x, 3, [512, 512, 2048], stage=5, block='b', train_bn=train_bn)
    C5 = x = identity_block(x, 3, [512, 512, 2048], stage=5, block='c', train_bn=train_bn)
else:
    C5 = None
return [C1, C2, C3, C4, C5]
```

```

# Top-down Layers
# TODO: add assert to varify feature map sizes match what's in config
P5 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (1, 1), name='fpn_c5p5')(C5)
P4 = KL.Add(name="fpn_p4add")([
    KL.UpSampling2D(size=(2, 2), name="fpn_p5upsampled")(P5),
    KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (1, 1), name='fpn_c4p4')(C4)])
P3 = KL.Add(name="fpn_p3add")([
    KL.UpSampling2D(size=(2, 2), name="fpn_p4upsampled")(P4),
    KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (1, 1), name='fpn_c3p3')(C3)])
P2 = KL.Add(name="fpn_p2add")([
    KL.UpSampling2D(size=(2, 2), name="fpn_p3upsampled")(P3),
    KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (1, 1), name='fpn_c2p2')(C2)])
# Attach 3x3 conv to all P layers to get the final feature maps.
P2 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (3, 3), padding="SAME", name="fpn_p2")(P2)
P3 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (3, 3), padding="SAME", name="fpn_p3")(P3)
P4 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (3, 3), padding="SAME", name="fpn_p4")(P4)
P5 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (3, 3), padding="SAME", name="fpn_p5")(P5)
# P6 is used for the 5th anchor scale in RPN. Generated by
# subsampling from P5 with stride of 2.
P6 = KL.MaxPooling2D(pool_size=(1, 1), strides=2, name="fpn_p6")(P5)

# Note that P6 is used in RPN, but not in the classifier heads.
rpn_feature_maps = [P2, P3, P4, P5, P6]
mrcnn_feature_maps = [P2, P3, P4, P5]

```

```

Epoch 1/5
500/500 [=====] - 1125s 2s/step - loss: 1.7622 - val_loss: 1.3775
WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/keras/callbacks/tensorboard_v1.py:343: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.

Epoch 2/5
500/500 [=====] - 825s 2s/step - loss: 0.9937 - val_loss: 0.6277
Epoch 3/5
500/500 [=====] - 961s 2s/step - loss: 0.7324 - val_loss: 0.5173
Epoch 4/5
500/500 [=====] - 964s 2s/step - loss: 0.5779 - val_loss: 0.3975
Epoch 5/5
500/500 [=====] - 955s 2s/step - loss: 0.4901 - val_loss: 0.4336

```

모델 학습에 사용한 데이터는 총 50장의 이미지로, 이 중 44개가 train data로, 5개가 validation data로 활용되었으며 이미지에서 인식하고 분류할 객체의 수(classes)는 태양열 패널을 설치할 수 있는 공간(옥상)인지를 식별하는 본 모델의 목적에 따라 1개(available)로 설정하였다. 초기 학습의 경우 default 값으로 COCO weight를 통해 진행되며, 이후 한 Epoch가 종료될 때마다 파라미터가 업데이트되어 다음 Epoch에 적용된다. 코드 작성에는 GitHub에 공개된 py 파일을 활용하였다. 상기 이미지로 확인할 수 있는 작성 코드가 GitHub에 공개된 py 파일의 내용이다. 해당 py 파일에서, 설명했던 MRCNN의 알고리즘을 구현하고 있으며 이를 라이브러리와 같이 편리하게 사용할 수 있도록 각종 함수들을 정의해 주고 있다. 정의된 함수들을 활용하여 코드를 작성하였다.

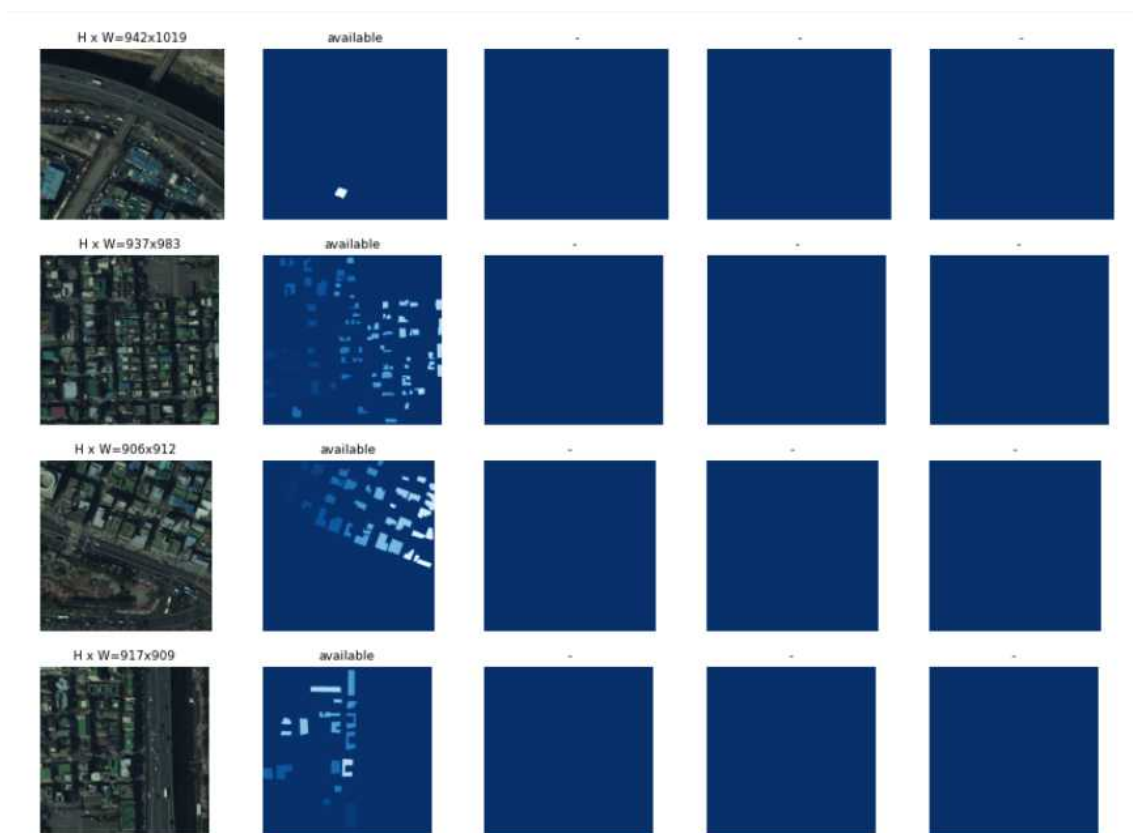
```

Annotation json path: /content/annotations.json
Annotation json path: /content/annotations.json
Train: 44
Validation: 5
Classes: 1

```



Mask R-CNN 구조를 활용해서 이미지에서 특정 개체를 인식하고 검출해 내는 코드를 구현하는 것은 어렵지 않았지만, 정의된 함수를 그대로 활용하는 데도 자꾸 구동이 됐다가 안됐다가 오류가 심했다. 확인해 보니 같은 Mask R-CNN 모델을 내려받거나 클론해서 사용하는 사람들이 많이 겪고 있는 문제였다. 사용자 개인 PC 환경에 따라 함수를 구동하기 위해 텐서플로나 케라스 버전을 정확히 맞춰줘야 하는 문제였다. 본 연구의 경우에는 **텐서플로는 1**이면 세부 버전은 상관없었고, 케라스는 처음에 2.2.X이면 상관없다는 검색 결과를 보고 2.2.5를 설정했다. 이유는 에러 메시지를 검색했을 때 같은 문제를 공유한 사람이 **케라스 버전을 2.2.5로 설정했더니 문제가 해결됐다**고 하는 걸 보고 그렇게 설정했다. 그럼에도 불구하고 같은 구간에서 같은 오류가 지속되어서 2.2.5부터 하나씩 버전을 단계별로 내려가면서 구동시켰고 **결국 2.0.8에서 구동되었다**. 나중에 알게 된 것인데, **GitHub에서 파일을 내려받을 때 포함되어 있던 파일들 중에 requirements.txt 문서에서 이런 조건들을 다 정리해두고 있었다**. 뭔가 파일들을 하나씩 확인해 볼 생각을 미리 못 했던 것이 허무했고, 이렇게 기작성된 자료들을 내려받아 구동시키는 과정에서 작성자가 적어놓은 것들을 잘 확인해 봐야겠다고 생각했다.



상기 이미지는 원본 이미지와 이미지 라벨링 단계에서 설정했던 마스크를 보여준다. **태양광 패널**을 설치할 수 있는 공간을 모델이 인식할 객체로 설정한 뒤 **주석(available)**을 달았고, 여기에 해당하지 않는 나머지 부분은 전부 배경으로 인식하도록 다른 주석을 달지 않았기 때문에 위의 이

미지에서 확인할 수 있는 3, 4, 5열의 경우 비어있는 것을 확인할 수 있다.

학습이 끝난 후 validation data 중 하나를 임의로 선택해 모델을 적용시킨 결과는 다음과 같다.  
(좌 원본, 우 모델링(예측) 결과)



## 5. 태양광 패널 설치 가능 면적 계산 알고리즘

태양광 패널 설치 가능 면적의 계산은 앞서 설계한 **Mask R-CNN 기반 모델에서 생성된 마스크의 픽셀 사이즈를 계산하고, 이후 이 값에 픽셀당 면적을 곱하는 형태로** 진행하였다. 픽셀당 면적은 선행연구를 참고하여 적용하였다.

픽셀 사이즈를 계산하는 코드 또한 GitHub에 공유된 코드를 활용하였는데, 앞 단계에서 옥상 영역을 검출하는 모델에서 학습을 통해 얻었던 파라미터를 다운로드해 옥상 면적을 검출하는 모델을 작성해서 적용하였다.

```
MODEL_DIR = "sunroof_fin.h5" # 옥상면적 검출모델 학습 결과 파라미터
# 모델 생성
config = InferenceConfig()
sunroof_model = modellib.MaskRCNN(mode="inference", model_dir="", config=config)
# 옥상면적 검출모델 학습 결과 파라미터 적용
sunroof_model.load_weights(MODEL_DIR, by_name=True)
```

먼저 이 모델을 적용해서 면적을 계산하고자 하는 옥상 영역에 대한 마스크를 획득하고, 마스크에 대한 면적을 구해주는 OpenCV 라이브러리를 사용해서 면적을 계산했다.

```

import cv2
from mrcnn_colab_engine import detect_contours_maskrcnn, draw_mask

img = cv2.imread("5.PNG")

print('List of available area')
# 1. 옥상 면적 영역 검출 모델 적용
class_ids, boxes, masks = detect_contours_maskrcnn(sunroof_model, img)
for class_id, box, object_contours in zip(class_ids, boxes, masks):
    # 박스
    y1, x1, y2, x2 = box
    # 외곽선
    cv2.polylines(img, [object_contours], True, colors[class_id], 10)
    img = draw_mask(img, [object_contours], colors[class_id])

    # 면적계산
    #픽셀당 면적 0.04 적용(이현우 외, "딥러닝을 활용한 옥상 면적 계산 모델" 참조)
    area_px = cv2.contourArea(object_contours)
    cv2.putText(img, "A: {}m^2".format(area_px * 0.04), (x1, y1), cv2.FONT_HERSHEY_PLAIN, 2, colors[class_id], 2)
    print(area_px * 0.04)

cv2.imshow(img)

```

원본 이미지에서 GT로 나타나는 실제 태양열 패널 설치가 가능한 것으로 나타나는 옥상 면적과 모델이 계산한 마스크를 통해 나타나는 계산된 옥상 면적의 비교(모델 계산 면적/GT 면적\*100)를 통해 모델의 예측력을 평가하였다. 추가로 구축한 별도의 이미지 데이터(20개)에 대해 모델을 적용한 뒤 예측력 평가를 진행한 결과 20장의 이미지 데이터에 대해 평균 62.85%의 정확도를 보여 모델의 정확도가 다소 낮은 것으로 확인되었는데, 이는 선행연구에서 지적하는 것처럼 옥상 위 물체나 그림자 등으로 인한 왜곡일 수도 있으나 본 연구의 경우 시간상 학습에 사용한 이미지 데이터 개수를 적게 설정하여 충분하지 못했고 Epoch 값도 너무 작게 설정하여 이러한 결과에 영향을 미친 것으로 생각된다.

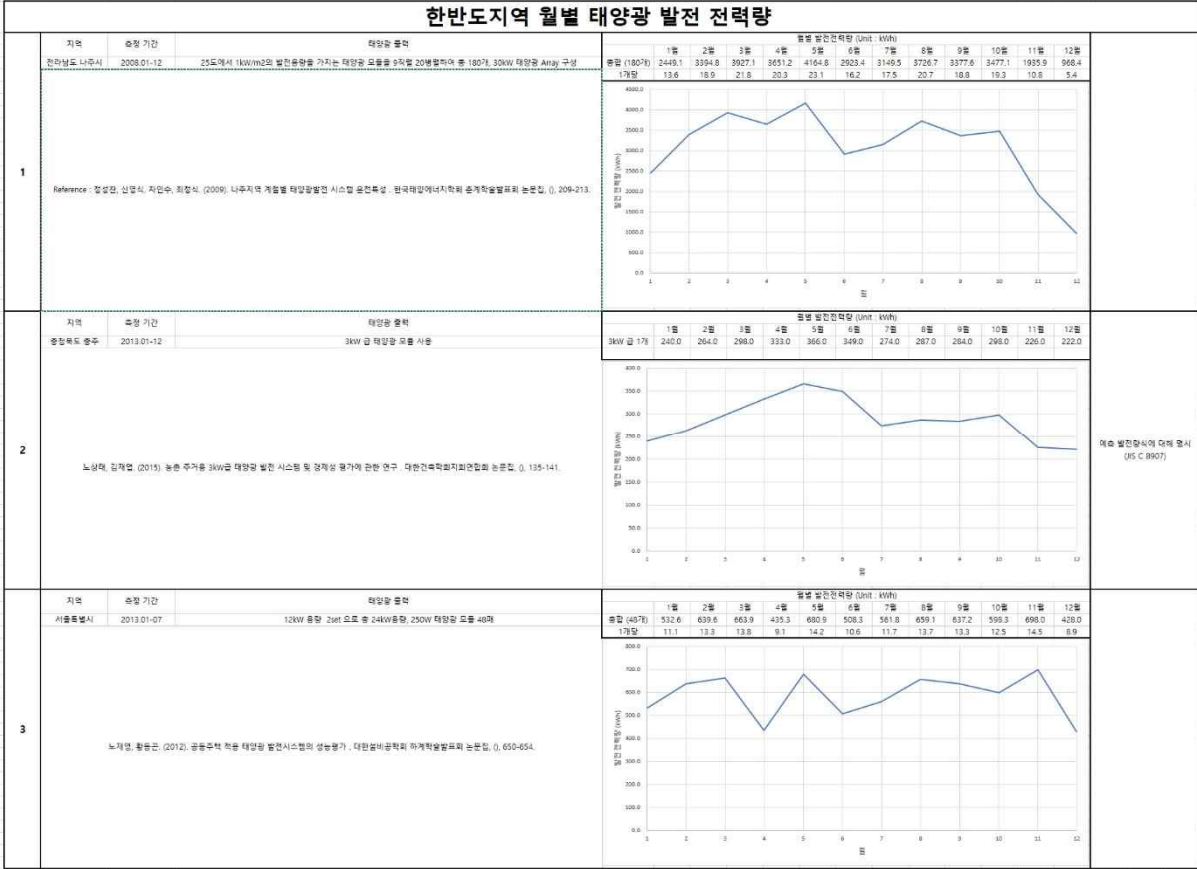
## 6. 모델의 적용 및 예상 전력생산량 계산

### 선행연구. 태양광 패널 전력생산량





국내에서는 태양광 패널에 의한 전력생산량을 1년 4계절을 기준으로 계절별로 상이한 전력 생산량에 대한 많은 연구가 진행되고 있다.

2009년 정성찬 등은 나주지역에 위치한 특정 건물에 설치된 30kW 급 태양광 발전시스템을 2008년 11월 - 2009년 3월까지의 전력 생산량 데이터를 자료화 하여 분석하였다. 2012년 노재영 등은 서울특별시의 공동주택을 대상으로 2013년 1월 - 7월까지의 24kW급 태양광 발전시스템의 전력 생산량을 데이터화 하여 태양광 발전시스템의 적용을 통한 성능평가를 수행하여 적용효과를 예측하고자 하였으며, 2015년 노상태 등은 일본 공업 규격 (JIS C 8907)을 통해 충청북도 충주시의 2013년 1월 - 12월까지의 3Kw급 태양광 발전시스템의 전력 생산량을 수집한 다음, 이를 분석하여 예측 발전량 식에 대한 제안을 하였다. 표 1은 기존 문헌의 내용 중 연간 태양광 발전시스템의 전력 생산량을 나타낸 것이다.





#### 표 2 태양광 발전시스템 설치 방식

설치분류	설치사진
(a) 가대설치형	
(b) 후면개방형	
(c) 지붕설치형	
(d) 견재일체형	

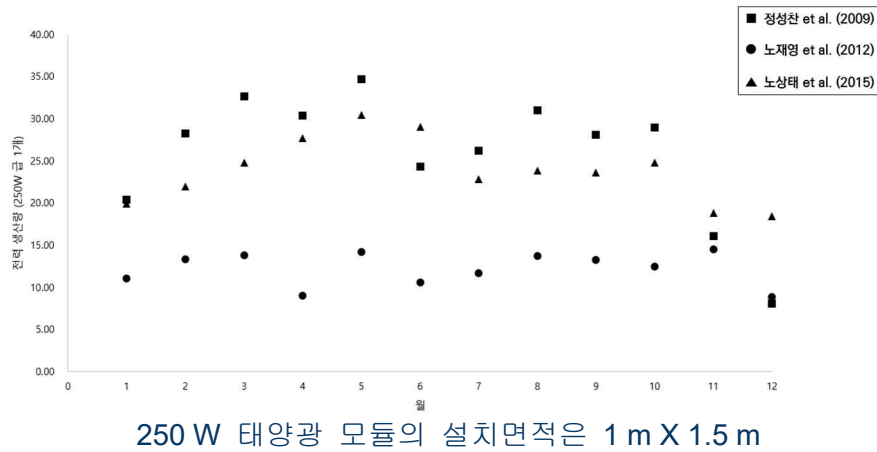
#### 태양광 발전시스템의 발전량 추정식

- 태양광 발전 시스템의 사양과 설치위치 그리고 표준 태양 전지 어레이 출력 (Pas)을 설정한다.
- 기본 설계 계수 K를 통하여 월별 종합 설계 계수 K를 구한다.
- 월 평균 일직선 경사면 일사량 (Hs)을 이용해 월 적산 경사면 일사량 (Ham)을 구한다.

$$E_{Pm} = (K \times P_{AS} \times H_{AM}) \div G_S \quad \dots\dots\dots (1)$$

$E_{Pm}$  : 월간시스템발전 전력량(kWh/month)  
 $K$  : 월별종합설계 계수  
 $P_{AS}$  : 어레이출력(kW)  
 $H_{AM}$  : 월적산경사면일사량(kWh×m<sup>2</sup>/month)  
 $G_S$  : 일사강도(kW/m<sup>2</sup>)

표 1. 기존 문헌 연간 태양광 발전 시스템 전력생산량 (250 W 1개 기준)



따라서 본 연구에서 사용된 데이터는 위의 기존 문헌에서 수집한 데이터의 평균값을 이용하여 연간 태양광 패널의 전력 생산량을 계산하였다.

계산 면적	설치 면적	전력량	전력 생산량(W)
245.844	1.5	250	40974
107.562	1.5	250	17927
104.3565	1.5	250	17392.75
174.4155	1.5	250	29069.25
180.7455	1.5	250	30124.25

## 7. 결론 및 활용방안

제주의 경우 신재생에너지 생산에만 집중된 정책 추진으로 인해 일조량이 많아지는 태양광 성수기인 4~5월에 출력제어를 걸어야 하는 상황이 발생하고 있다. 출력제어는 태양광·풍력발전에서 생산하는 전력이 수요보다 많아 전력 계통에 과부하가 우려될 때 한전이 전력거래소를 통해 발전사업자에게 발전(전력 생산)을 일시 중단하라고 요청하는 조치다. 전력 계통에 과부하가 걸리면 정전 사태가 발생할 수 있기 때문이다. 발전사업자가 이 조치를 따르지 않으면 발전소 가동은 강제로 중단된다. \*태양광 생산 넘쳐, 특하면 강제 중단 (매일경제, 2022)

출력제어는 수요예측에 실패한 정부의 책임은 물론 발전량에 대한 추계가 완비되어 있지 않기 때문이라고 할 수 있다. 따라서 생산되는 전력량에 대한 정확한 예측을 위해 계절별, 면적별, 그리고 태양광 시설의 입지별로 생산되는 전력량을 실시간으로 모니터링할 수 있는 시스템의 구축이 필요하다.

이런 측면에서 본 연구는 지역단위별 태양광 패널의 생산 전력량의 예측을 하고, 계절과 입지에

다른 일조량 통계를 기반으로 민간 전력 생산량 추이를 추계할 수 있는 방법으로 활용될 수 있다. 즉, 태양광 에너지 생산 및 분배를 위한 관제 시스템 구축에 기여할 수 있을 것이다. 근래에는 지역적으로 분산되어 있는 소규모 태양광, 풍력 등 재생에너지 발전사업자를 효율적으로 연결하고 발전량을 예측·관리하기 위해 가상 발전소(VPP : Virtual Power Plant)라는 개념으로 대두되고 있는 사업과의 연결성을 고려해 볼 수 있다. VPP는 유럽이나 미국 등 신재생에너지 발전량이 많은 나라에서는 에너지 예측과 전원 관리를 위해 활성화돼 있는 방식이다. 전기차 기업 테슬라도 VPP 사업에 뛰어든 업체 중 하나다. 앞서 언급한 것처럼 태양광 에너지의 경우 생산에 대한 예측이 어려울 수 있는데 그것을 간헐성 문제라고 부른다. 재생에너지의 간헐성은 날씨에 따라 발전량이 달라지는 것을 일컫는다. 흔히 재생에너지로 꼽는 태양광·풍력·수력 등은 일조량·풍속 등 날씨의 영향을 받고 낮과 밤의 기온 차이도 크다. 보고서에서 다루고 있는 태양광은 오전 7시쯤부터 낮 동안은 공급이 과다하고 오후 5시 이후 저녁과 밤 동안에는 공급이 뚝 떨어지는 특성이 있다.

현재까지 VPP는 정부가 공개한 공공데이터를 기반으로 한 출력 통계에 기반하여 운영되고 있다. 보고서에서 다루고 있는 면적별 전력 생산의 경우 공공에서 다루지 않는 데이터의 크레바스(빈틈)를 다룰 수 있는 가능성이 크고, 맵의 지속적인 업데이트를 통해 비교적 손쉽게 데이터의 업데이트도 동시에 가능할 수 있다는 장점이 있다. 이는 본 연구가 궁극적으로 우리나라 신재생에너지 관리 방법 및 재생에너지 분배를 위한 시스템 구축에 새로운 실마리를 제공한다고 볼 수 있다. 나아가 지속적인 Real-time 머신러닝으로 위성을 통한 실시간 관제까지 가능할 수 있지 않을까?

## 8. 참고자료 및 문헌

- 1) 국토정보플랫폼 국토정보맵 (<https://map.ngii.go.kr/ms/map/NlipMap.do>)
- 2) Piotr Skalski 제공하는 오픈소스 프로젝트: Make Sense (<https://www.makesense.ai>)
- 3) He, Kaiming, et al. "Mask R-CNN", arXiv 2017, pp. 1-12, 2017.
- 4) 이현우, 양준호, 송명주, 김송, 김재현, 2020, "딥러닝을 활용한 건물 옥상 면적 계산 모델", 아주대학교
- 5) 김영훈, 유은순, 강수환, 박승보, 2019, "MRCNN을 이용한 영화 속 등장인물 면적추출 방법", 한국컴퓨터정보학회 동계학술대회 논문집, 27(1).
- 6) 정성찬, 신영식, 차인수, 최정식. (2009). 나주지역 계절별 태양광발전 시스템 운전특성 . 한국태양에너지학회 춘계학술발표회 논문집, (), 209-213.
- 7) 노재영, 황동근. (2012). 공동주택 적용 태양광 발전시스템의 성능평가 . 대한설비공학회 하계학술발표회 논문집, (), 650-654.

- 8) 노상태, 김재엽. (2015). 농촌 주거용 3kW급 태양광 발전 시스템 및 경제성 평가에 관한 연구 . 대한건축학회지회연합회 논문집, (), 135-141.
- 9) 태양광 생산 넘쳐, 특하면 강제 중단 (매일경제. 2022,  
<https://www.mk.co.kr/news/economy/view/2022/04/379676/>)