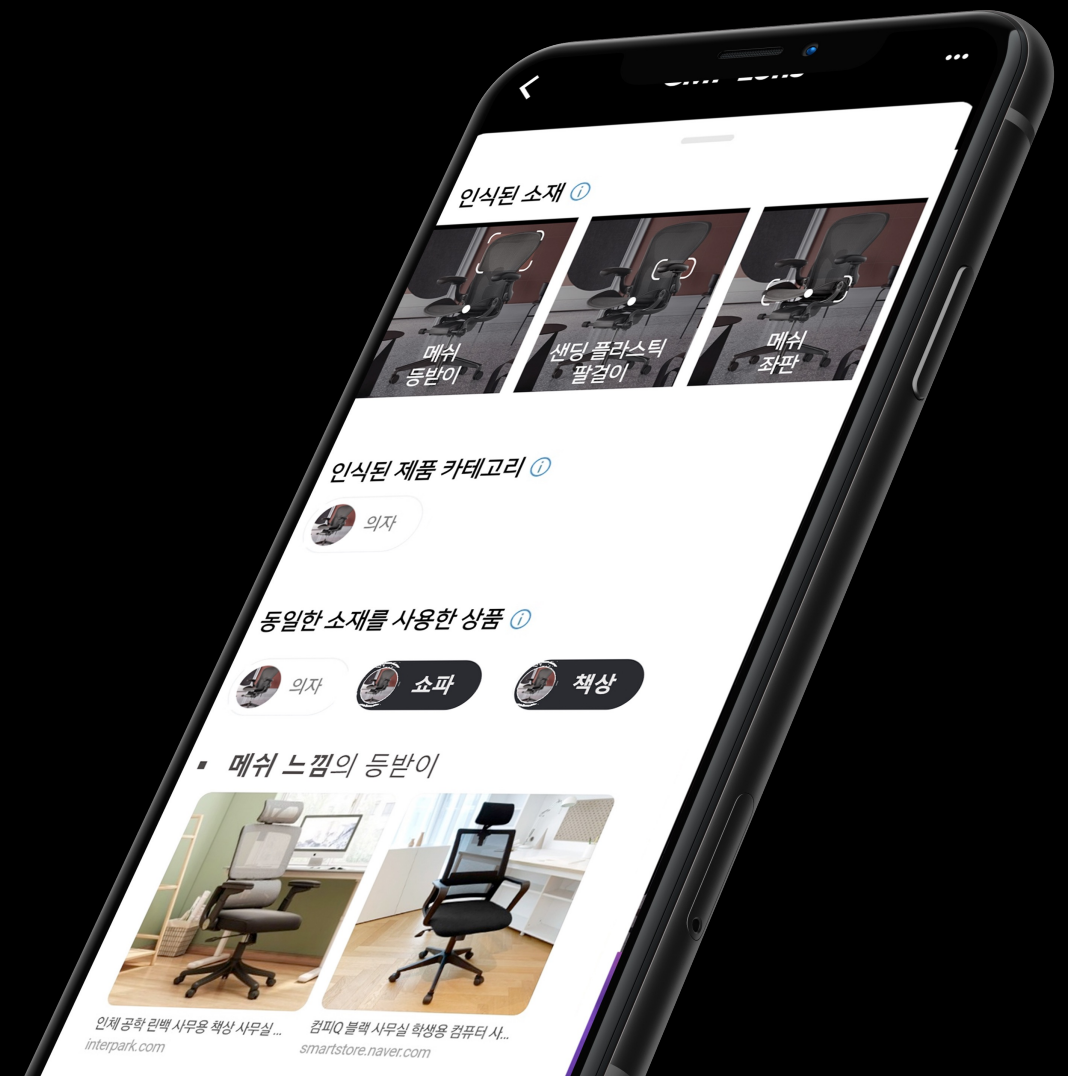
원하는 소재와 상품을

# #이미지로 검색

# #CMF-Lens




**배 진**

중앙대학교 경영학과

Market trend
painpoints

Semantic segmentation
Semantic part segmentation
Image classification

**Market Trend**

Mobile-First 시대의 트렌드

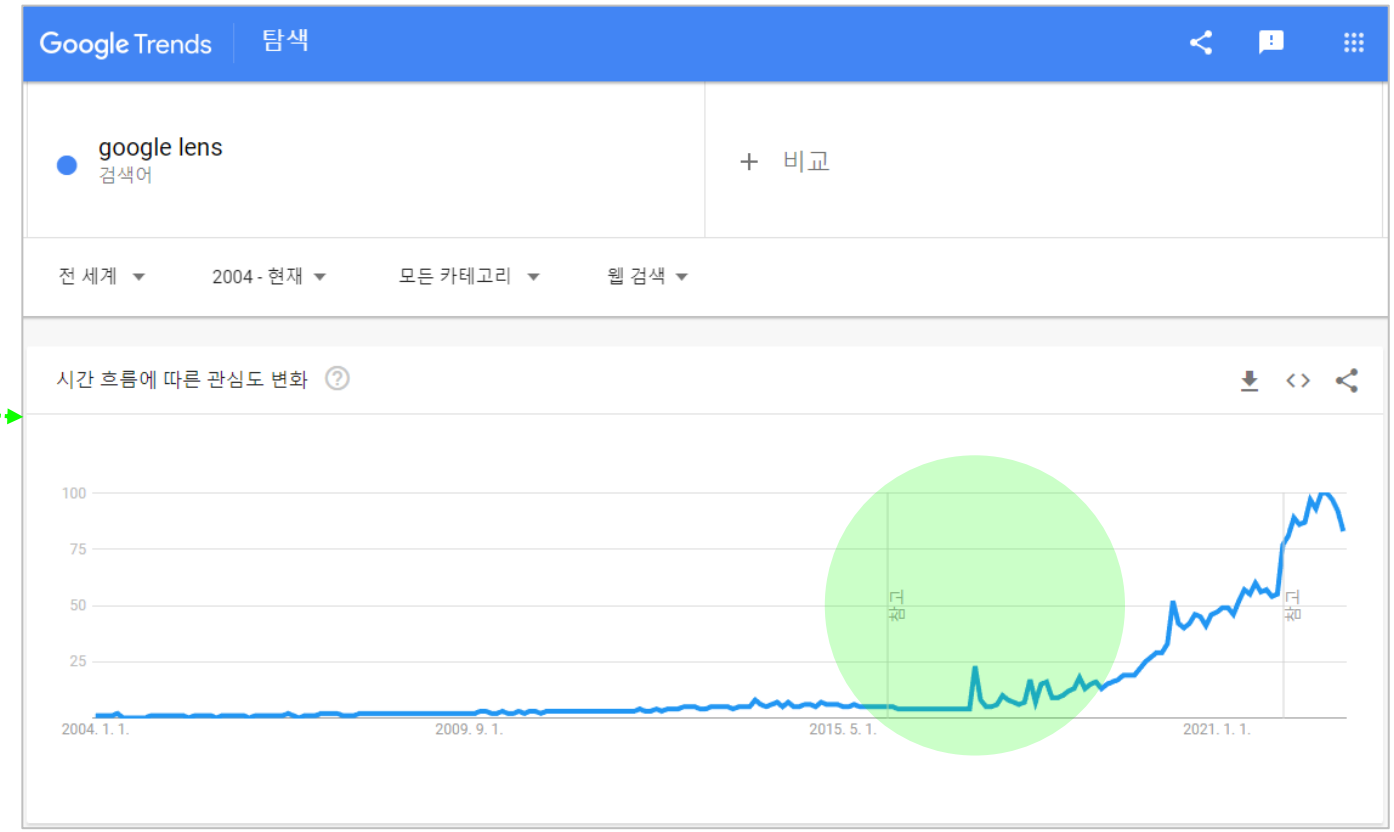
**#이미지로 검색**

**#search by image**

Google Lens
**1B+**
Downloads

Naver SmartLens
**100M+**
Downloads

모야모
(식물 식별, 병충해 진단)
**1M+**
Downloads



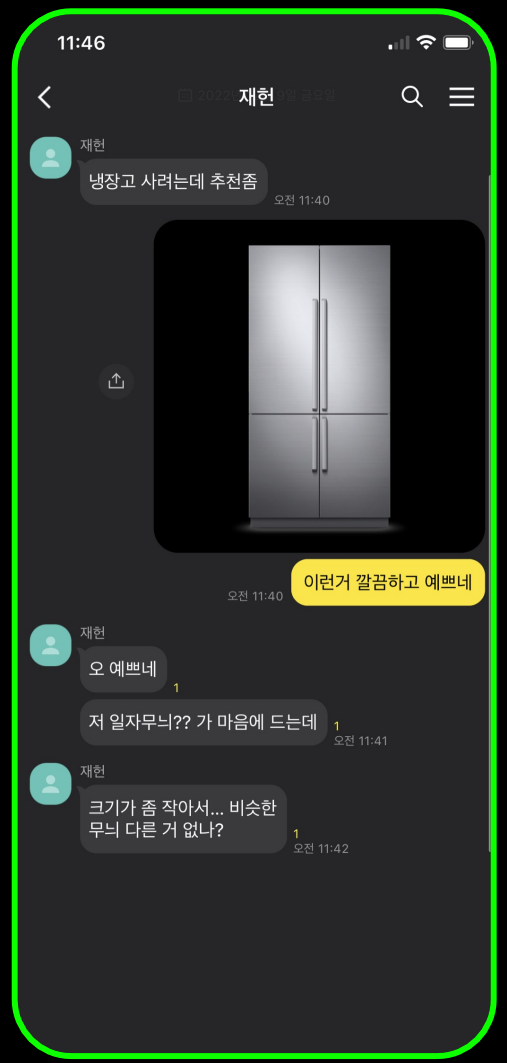
painpoint 01

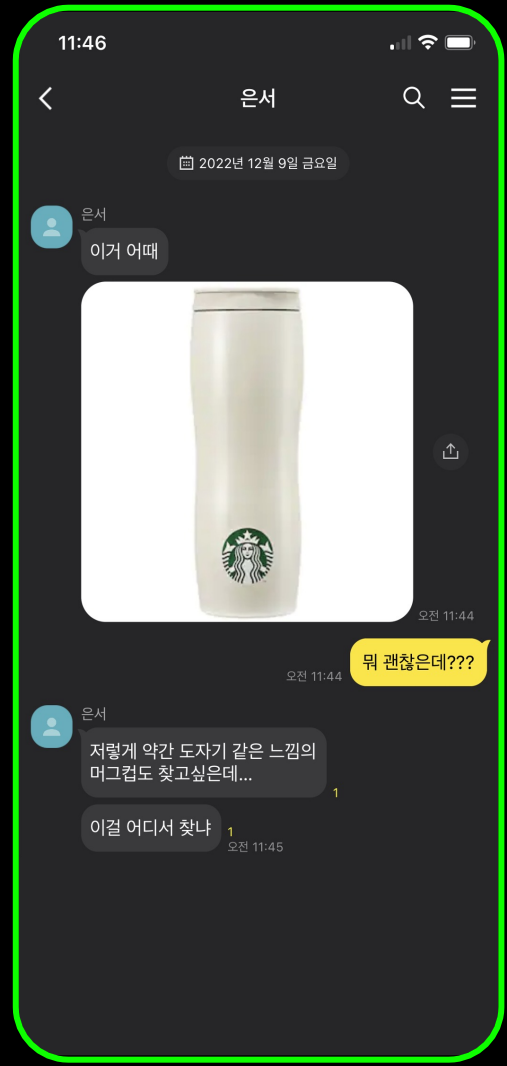
알고 싶어도
검색어를 몰라서
못 찾는

#color

#material

#finish

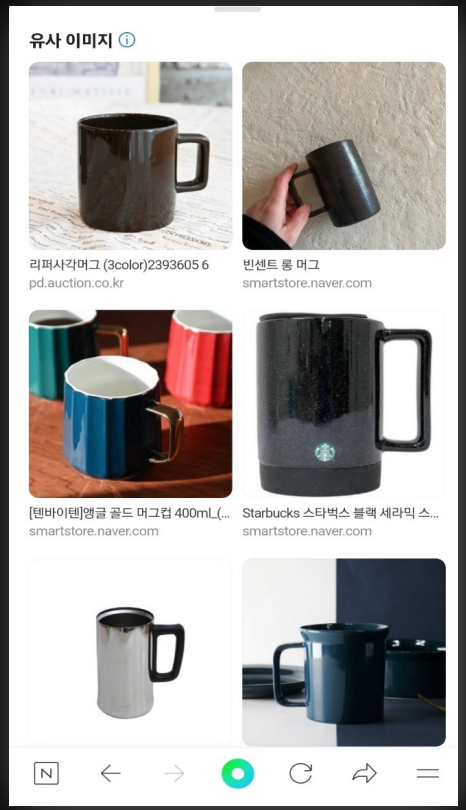



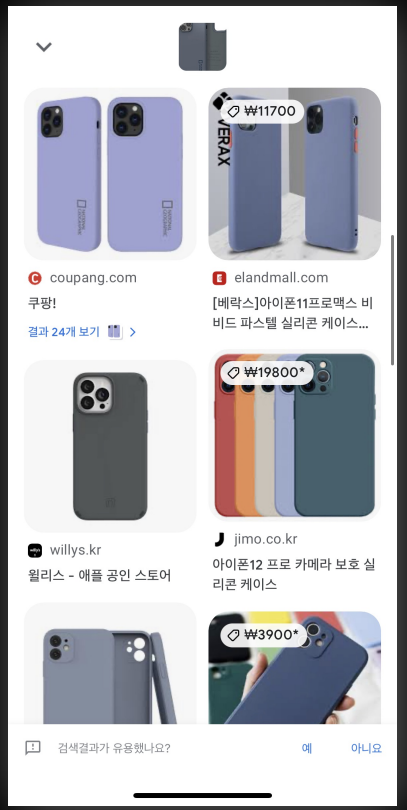
painpoint 02

CMF를
인식하지 못하는
기존의 이미지
검색 서비스들

NAVER SmartLens


Google Lens

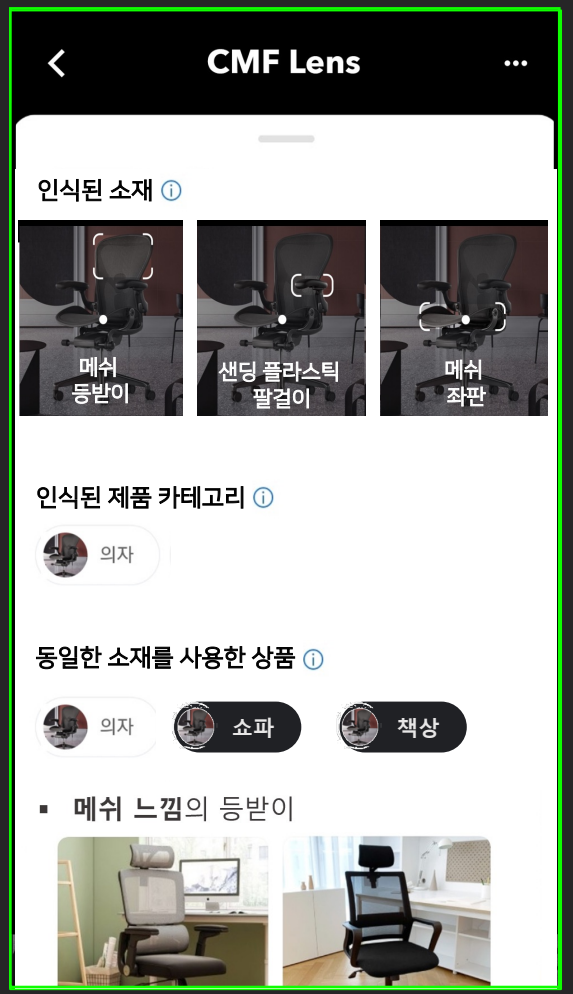
# 원하는 소재와 상품을

**#이미지로 검색**

**#CMF-Lens**

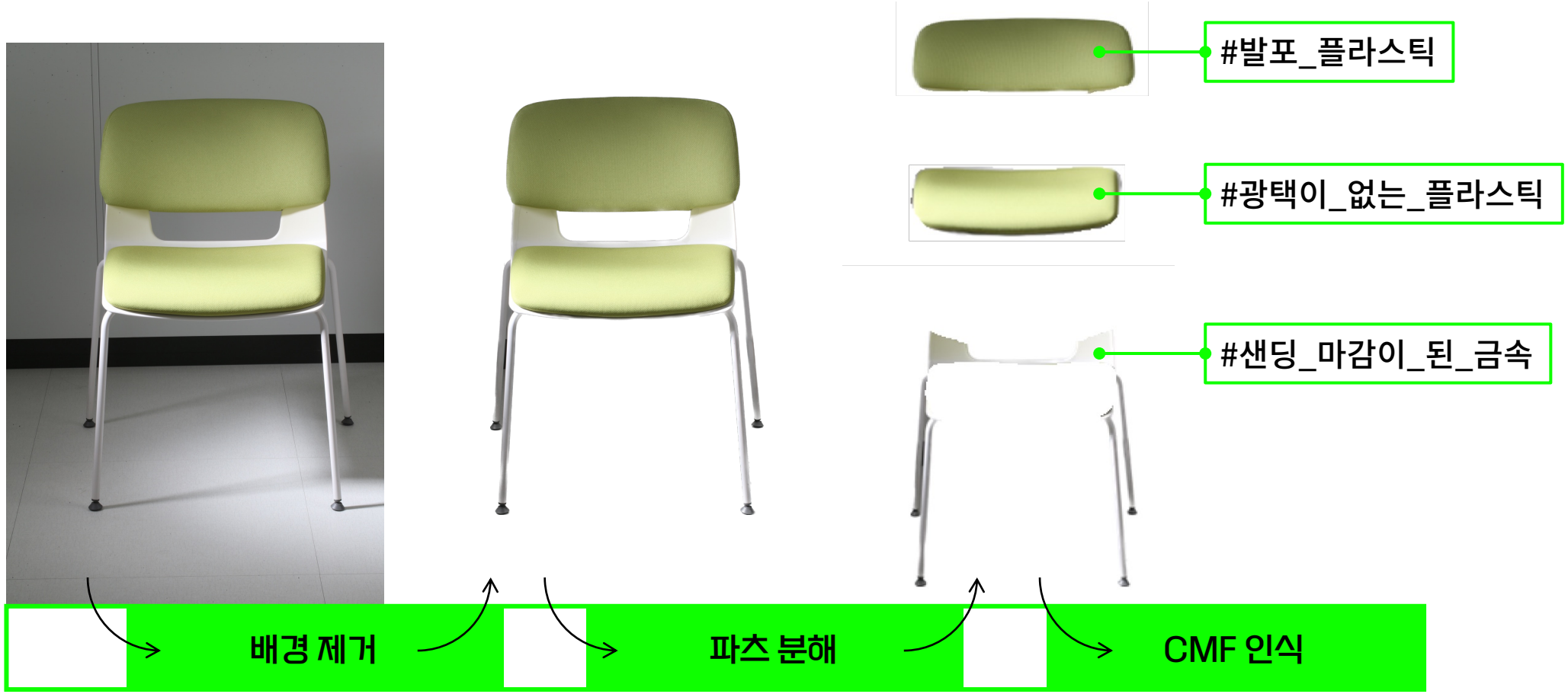
이미지 촬영 및 업로드


결과 화면 1




결과 화면 2

# 서비스 구현을 위한 **이미지 처리 기술**



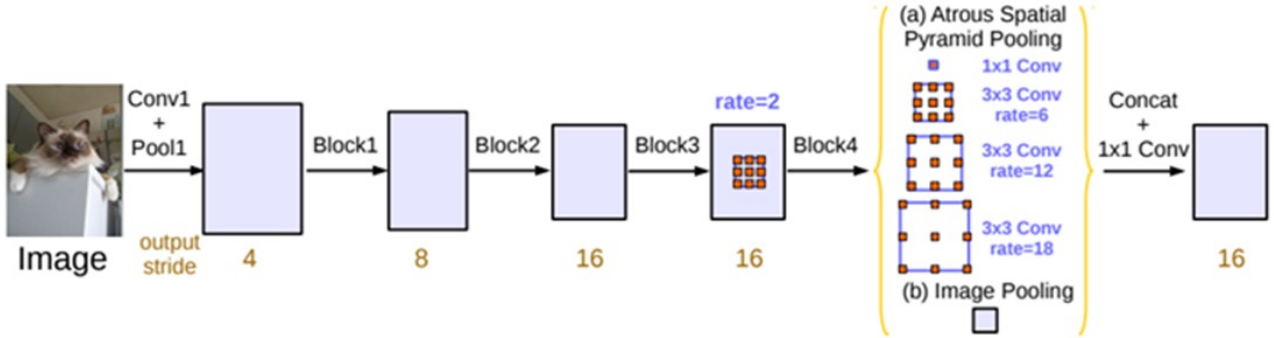
(1) 배경 제거

Semantic segmentation

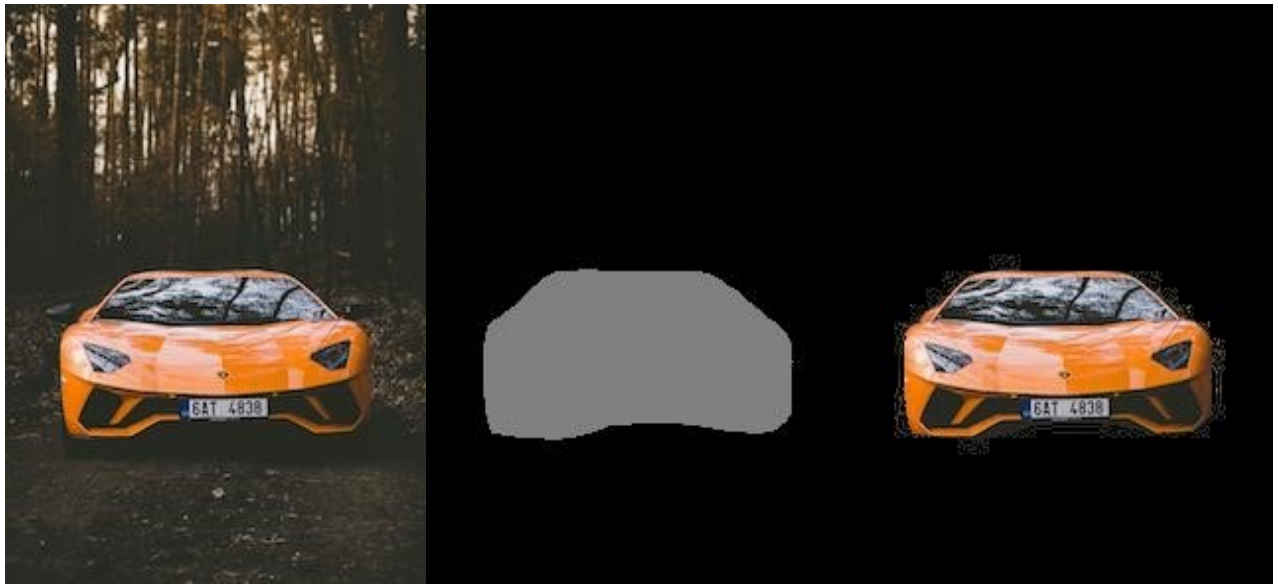
(2) 파츠 분해

(3) CMF 인식

(1) 이미지를 픽셀 단위로 분류 가능한 CNN 기반 딥러닝 알고리즘




(2) Atrous convolution layer를 활용하여 픽셀 주변의 맥락을 넓게 파악



(1) 배경 제거

Semantic segmentation

(2) 파츠 분해

(3) CMF 인식

## (3) Deeplabv3를 활용한 Semantic Segmentation 모델 프로토타이핑

```
import torch
#Pre-trained 모델 불러오기
model = torch.hub.load('pytorch/vision:v0.10.0', 'deeplabv3_resnet50', pretrained=True)
model.eval()
```

```
DeepLabHead(
  (0): ASPP(
    (convs): ModuleList(
      (0): Sequential(
        (0): Conv2d(2048, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (1): ASPPConv(
        (0): Conv2d(2048, 256, kernel_size=(3, 3), stride=(1, 1), padding=(12, 12), dilation=(12, 12), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (2): ASPPConv(
        (0): Conv2d(2048, 256, kernel_size=(3, 3), stride=(1, 1), padding=(24, 24), dilation=(24, 24), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (3): ASPPConv(
        (0): Conv2d(2048, 256, kernel_size=(3, 3), stride=(1, 1), padding=(36, 36), dilation=(36, 36), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (4): ASPPPooling(
        (0): AdaptiveAvgPool2d(output_size=1)
        (1): Conv2d(2048, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (3): ReLU()
      )
    )
    (project): Sequential(
      (0): Conv2d(1280, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): Dropout(p=0.5, inplace=False)
    )
  )
  (1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): ReLU()
  (4): Conv2d(256, 21, kernel_size=(1, 1), stride=(1, 1))
)
```

```
FCNHead(
  (0): Conv2d(1024, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU()
  (3): Dropout(p=0.1, inplace=False)
  (4): Conv2d(256, 21, kernel_size=(1, 1), stride=(1, 1))
)
```

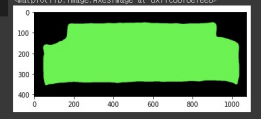
```
# 이미지 불러오기
import urllib
url, filename = ("https://i.pinimg.com/originals/4a/6c/78/4a6c7847b23a0e0e01a4b8f9843058f5.png", "deeplabl.png")
try: urllib.URLopener().retrieve(url, filename)
except: urllib.request.urlretrieve(url, filename)

import matplotlib.pyplot as plt
import matplotlib.image as mpimg
img = mpimg.imread(filename)
imgplot = plt.imshow(img)
```

```
# 대상 제품 segmentation
from PIL import Image
from torchvision import transforms
input_image = Image.open(filename)
input_image = input_image.convert("RGB")
preprocess = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0) # create a mini-batch as expected by the model

# move the input and model to GPU for speed if available
if torch.cuda.is_available():
    input_batch = input_batch.to('cuda')
    model.to('cuda')
print(input_batch.shape)
with torch.no_grad():
    output = model(input_batch)['out'][0]
output_predictions = output.argmax(0)
```

```
torch.Size([1, 3, 409, 1072])
```

```
palette = torch.tensor([2 ** 25 - 1, 2 ** 15 - 1, 2 ** 21 - 1])
colors = torch.as_tensor([i for i in range(21)])[:, None] * palette
colors = (colors % 255).numpy().astype("uint8")

# plot the semantic segmentation predictions of 21 classes in each color
r = Image.fromarray(output_predictions.byte().cpu().numpy()).resize(input_image.size)
r.putpalette(colors)


plt.imshow(r)
# plt.show()
```

<matplotlib.image.AxesImage at 0x7fc081f6e7e0>



```
import numpy as np
import cv2
def make_transparent_foreground(pic, mask):
  # split the image into channels
  b, g, r, a = cv2.split(np.array(pic).astype('uint8'))
  # cv2 split returns b,g,r,a with updated version
  # a = np.ones(mask.shape, dtype='uint8') * 255
  # merge the alpha channel back
  alpha_im = cv2.merge([b, g, r, a], 4)
  # create a transparent background
  bg = np.zeros(alpha_im.shape)
  # setup the new mask
  new_mask = np.stack([mask, mask, mask, mask], axis=2)
  # copy only the foreground color pixels from the original image where mask is set
  foreground = np.where(new_mask, alpha_im, bg).astype(np.uint8)

  return foreground
```

```
# 배경 제거
input_image = Image.open(filename)
foreground = make_transparent_foreground(input_image, r)
plt.imshow(foreground)
```

<matplotlib.image.AxesImage at 0x7fc03200928D>

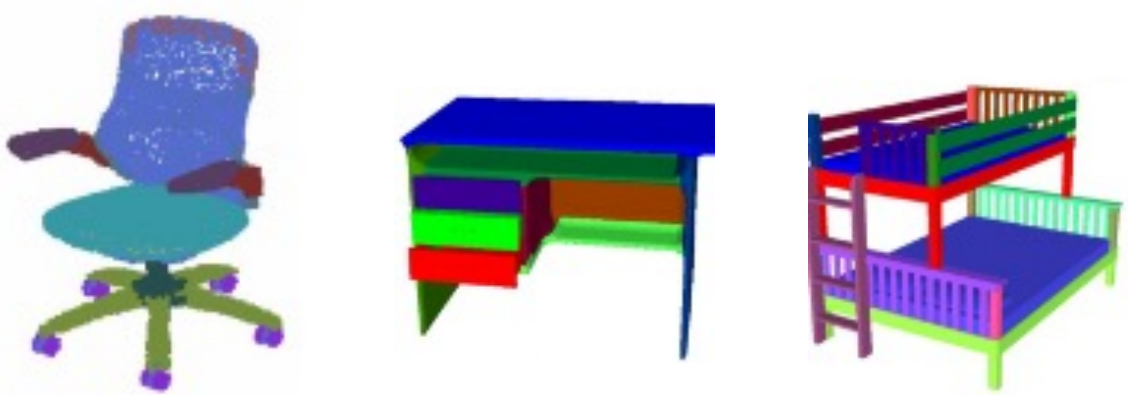
(1) 배경 제거

(2) 파츠 분해

Semantic part segmentation

(3) CMF 인식

(1) 제품 파츠 별 라벨링된 데이터를 학습한 semantic segmantation 모델



(2) PASCAL-Part DataSet, PartImageNet에서 제품 파츠 라벨링 데이터 공개

(1) 배경 제거

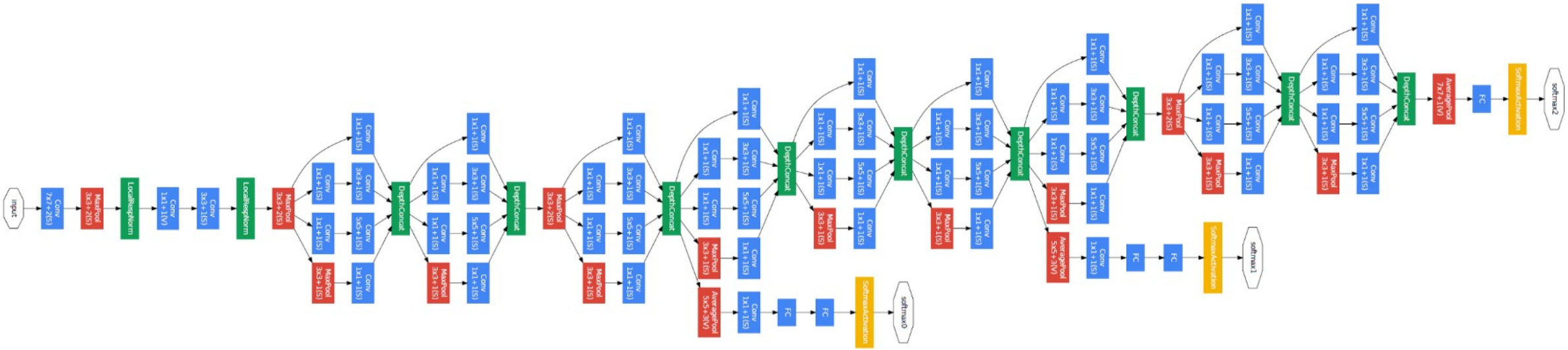
(2) 파츠 분해

(3) CMF 인식

Image classification

(1) CMF 식별 데이터를 학습하여 제품 파츠 별 CMF 이미지를 분류하기 위한 모델

(2) 1,400만장의 ImageNet Dataset으로 Pre-trained 된,
Inception 또는 ResNet 모델의 성능이 준수하며, 이를 Fine-tune 하여 정확도를 향상



(1) 배경 제거

(2) 파츠 분해

(3) CMF 인식

Image classification

-InceptionV3를 활용한 Image Classfication 모델 프로토타이핑

-FMD(Flickr Material Database) 데이터를 활용, 모델을 Fine-Tuning 하였음

```
#Unzip file

!unzip -qq '/content/FMD.zip'
```

```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

```
# 이미지 불러오기
datagen = ImageDataGenerator(rescale = 1.0/255, validation_split = 0.2)
train_data = datagen.flow_from_directory('/content/image/',class_mode = 'categorical',
                                         target_size = (128, 128), subset = 'training',
                                         batch_size = 10)
val_data = datagen.flow_from_directory('/content/image/',class_mode = 'categorical',
                                       target_size = (128, 128), subset = 'validation',
                                       batch_size = 10)
```

```
Found 800 images belonging to 10 classes.
Found 200 images belonging to 10 classes.
```

```
# Pretrained 모델 불러오기
from keras.applications import InceptionV3

base_model = tf.keras.applications.InceptionV3(

    weights='imagenet',
    input_tensor = None,
    input_shape = (128, 128, 3),
    include_top = False,
    pooling = None,
    classes = 1000,
)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 3s 0us/step
```

```
base_model.trainable = False
new_extended_inception_weights = '/content/final_weights.hdf5'
```

```
# 모델 fine-tuning
import keras
from keras import layers
inputs = keras.Input(shape=(128,128,3))
x = base_model(inputs)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(1024, activation = 'relu')(x)
x = layers.Dropout(0.5)(x)

outputs = layers.Dense(10, activation = 'softmax')(x)
model = keras.Model(inputs, outputs)
```

```
# 모델 학습
model.compile(optimizer = keras.optimizers.Adam(lr = 0.0001),
              loss=keras.losses.CategoricalCrossentropy(),
              metrics=['accuracy'])
model.fit(train_data,
          validation_data = val_data,
          steps_per_epoch = 80 ,
          epochs= 50,validation_steps = 20 )
model.save_weights(new_extended_inception_weights)
```

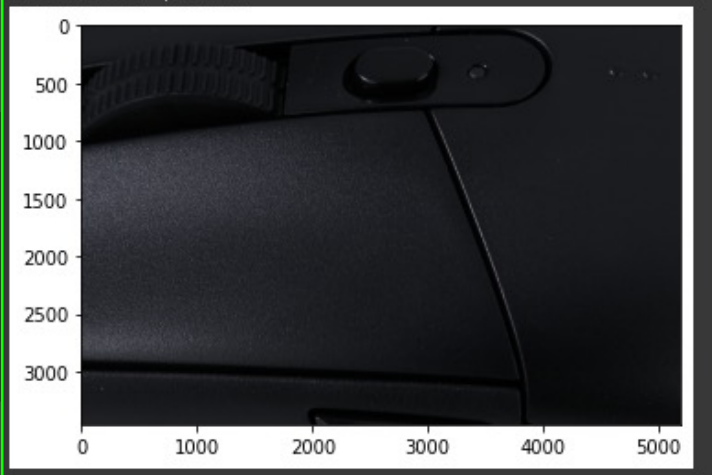
```
Epoch 22/50
80/80 [==============================] - 4s 55ms/step - loss: 0.0102 - accuracy: 0.9975 - val_loss: 2.5802 - val_accuracy: 0.6300
Epoch 23/50
80/80 [==============================] - 5s 67ms/step - loss: 0.0048 - accuracy: 0.9987 - val_loss: 2.4045 - val_accuracy: 0.6050
Epoch 24/50
80/80 [==============================] - 5s 56ms/step - loss: 0.0066 - accuracy: 0.9987 - val_loss: 2.2596 - val_accuracy: 0.5750
Epoch 25/50
80/80 [==============================] - 5s 56ms/step - loss: 0.0061 - accuracy: 0.9975 - val_loss: 2.4238 - val_accuracy: 0.5850
Epoch 26/50
80/80 [==============================] - 4s 55ms/step - loss: 0.0053 - accuracy: 0.9987 - val_loss: 2.2249 - val_accuracy: 0.6100
Epoch 27/50
80/80 [==============================] - 4s 55ms/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 2.2956 - val_accuracy: 0.6200
Epoch 28/50
80/80 [==============================] - 4s 54ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 2.2769 - val_accuracy: 0.6050
Epoch 29/50
80/80 [==============================] - 4s 54ms/step - loss: 7.0177e-04 - accuracy: 1.0000 - val_loss: 2.2668 - val_accuracy: 0.6000
Epoch 30/50
```

```
#CMF 데이터셋에 테스트
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

categories = ['fabric', 'foliage', 'glass', 'leather', 'metal', 'paper', 'plastic', 'stone', 'water', 'wood']
prediction = model.predict(test_data)


index = 0
for i in prediction:
  pred_category = np.argmax(i)
  print('prediction: ' + categories[pred_category] + ', answer: ' + answer.split('/')[0])
  index += 1
```

```
1/1 [==============================] - 1s 959ms/step
prediction: metal, answer: metal
prediction: plastic, answer: plastic
prediction: wood, answer: wood
```

```
answer = str(test_data.filenames[1])
img = mpimg.imread('/content/test/' + answer)
plt.imshow(img)
print('Prediction: ' + categories[np.argmax(prediction[1])])
```

```
Prediction: plastic
```

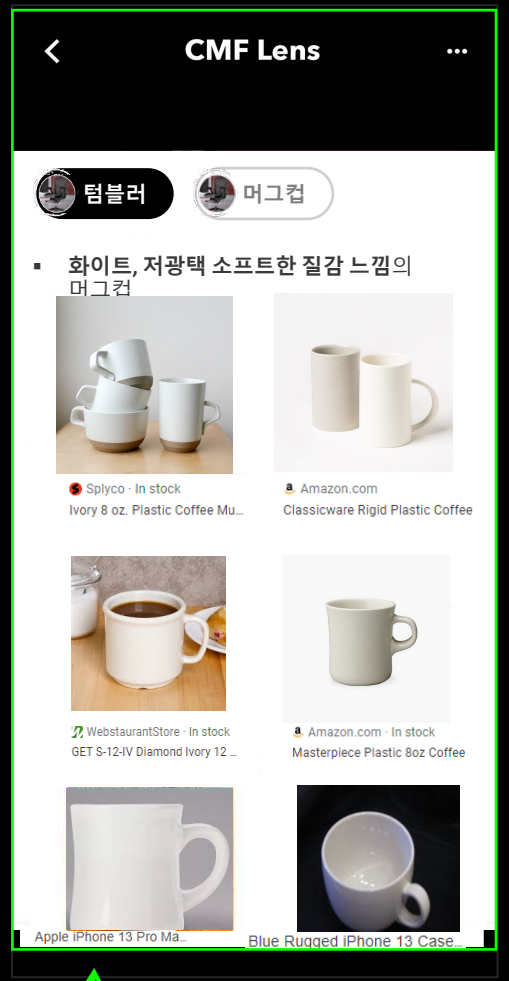


## 소비자

- 이미지로 편리한 상품 검색 UI/UX
- 소재 기반 상품 검색 정확도 향상
- 좋아하는 색상이나 소재를 활용한
  다른 카테고리의 상품도 손쉽게 검색 가능


## 디자이너

- 신속한 시장 리서치
- 카테고리별 CMF디자인 제품 벤치마킹
- CMF디자인 시장 반응/트렌드 분석에 활용

## 디자이너

- 신속한 시장 리서치
- 카테고리별 CMF디자인 제품 벤치마킹
- CMF디자인 시장 반응/트렌드 분석에 활용


스크랩

공유하기

리서치(Professional)

# 감사합니다.

**#좀볼줄아는**

**#CMF_Lens** **#츨카메라**