

모델 가져다 쓰기

# 개요

- 자신의 모델을 만드는 방법
- 기존 모델을 가져다 쓰는 경우가 존재 (\*오늘 강의)
  - Case1 : 기존 모델 틀만 가져다 쓰는 경우
  - Case2 : 기존에 학습된 모델을 가져다 쓰는 경우
    - 학습된 모델을 그대로 활용
    - 학습된 모델을 이용하여 새로운 데이터에 적용

# Case1 : 기존 모델 가져다 쓰기

- 기존 모델 : 연구자들이 이미 연구를 통해 검증이 끝난 모델
- 연구 논문을 보거나 해당 github를 살펴보면 쉽게 획득 가능
- [paperswithcode.com](https://paperswithcode.com)
- [keras.applications](https://keras.applications)
- tensorflow Hub ([tfhub.dev](https://tfhub.dev))

# 기존 모델 가져다쓰기

## part1 : 일단 따라해보자

# 일단 따라해보자 :

## Keras Application [KA]

[참고] <https://keras.io/applications/>

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201

# 일단 따라해보자 :

## Keras Application [KA]

```
!wget http://www.personal.psu.edu/afr3/blogs/SIOW/elephant.jpg
```

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

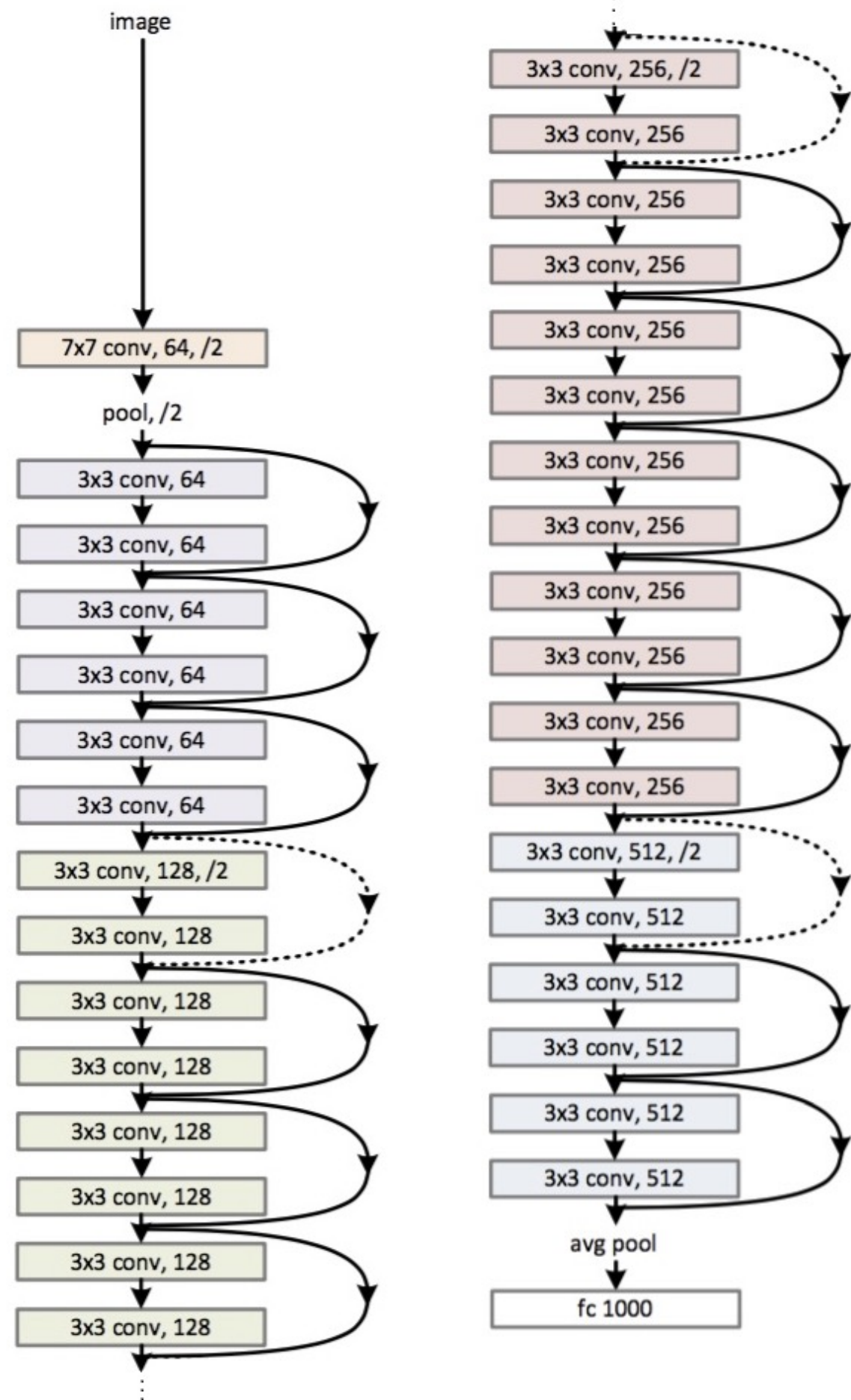
model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

# ResNet

34-layer residual



# [KA] : 모델 그대로 사용하기

- 학습된 모델 그대로를 사용하고 싶은 경우
- `weights = 'imagenet', include_top=True`

```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
# Predicted: [(u'n02504013', u'Indian_elephant', 0.82658225), (u'n01871265', u'tusker', 0
```



# 실습 : ResNet을 이용한 응용만들기

- 기 학습된 ResNet50 모델을 기반으로 실제 응용을 만들어 보자
- 카메라에서 영상을 입력 받아서 해당 입력이 무엇인지 맞추어 보자
- 준비물 : Colab, 웹캠, 학습된 모델(resnet50)

# 실습 : ResNet을 이용한 응용만들기

- Colab에서 웹캠 접근하기 (Ipython과 Javascript간의 데이터를 주고 받을 수 있는 라이브러리가 있음. 아래 링크 참조)

[https://colab.research.google.com/notebooks/snippets/advanced\\_outputs.ipynb#scrollTo=2viqYx97hPMi](https://colab.research.google.com/notebooks/snippets/advanced_outputs.ipynb#scrollTo=2viqYx97hPMi)

- [절차]
  1. 기존의 학습된 ResNet50 모델을 가져온다.
  2. 웹캠에서 이미지를 입력 받는다 (스마트폰을 이용하시면 편합니다.)
  3. 모델에 의해서 분류된 최종 결과를 화면에 보여준다.

```
[ ] from IPython.display import display, Javascript
    from google.colab.output import eval_js
    from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
    js = Javascript('''
        async function takePhoto(quality) {
            const div = document.createElement('div');
            const capture = document.createElement('button');
            capture.textContent = 'Capture';
            div.appendChild(capture);

            const video = document.createElement('video');
            video.style.display = 'block';
            const stream = await navigator.mediaDevices.getUserMedia({video: true});

            document.body.appendChild(div);
            div.appendChild(video);
            video.srcObject = stream;
            await video.play();

            // Resize the output to fit the video element.
            google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

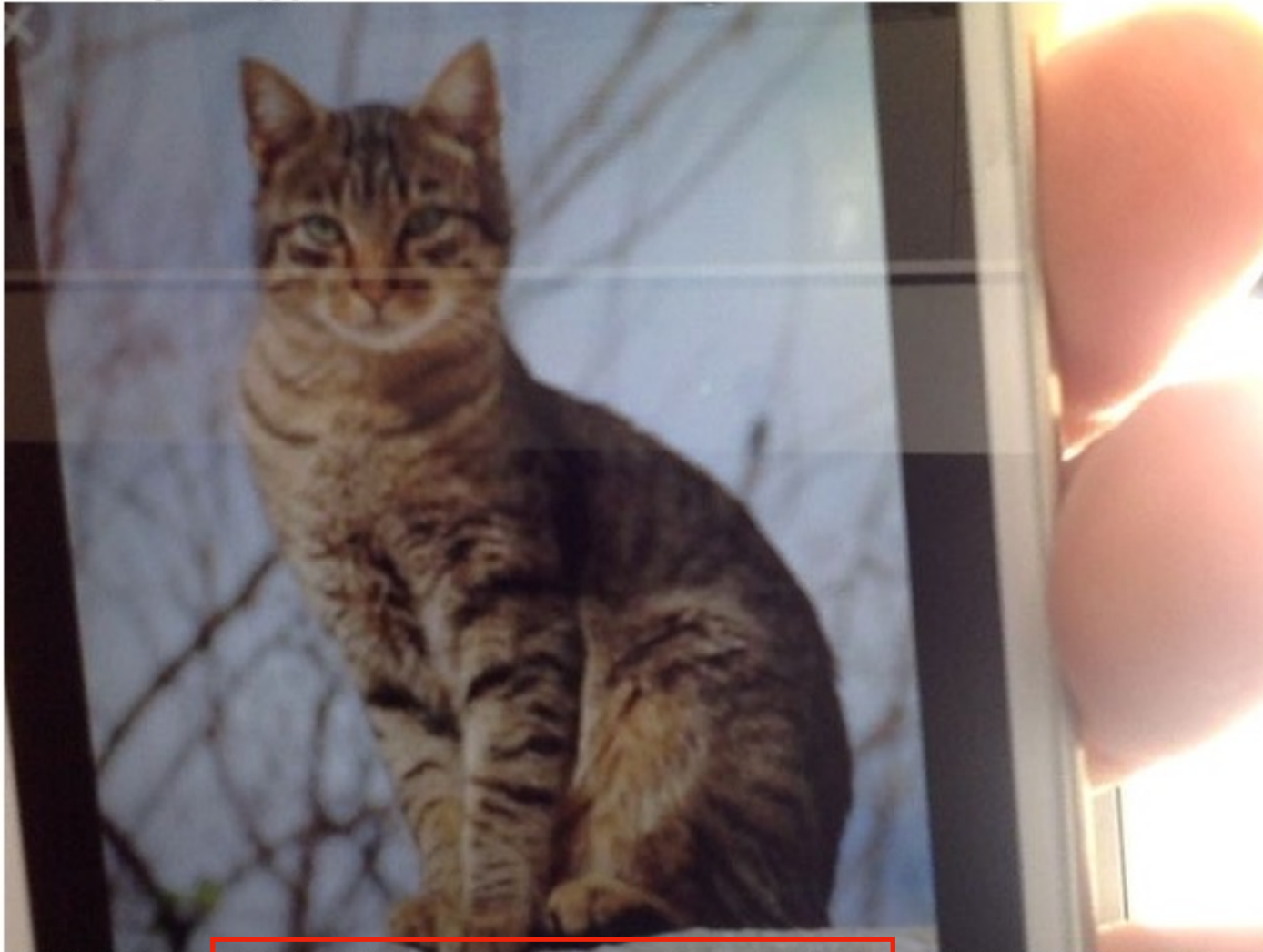
            // Wait for Capture to be clicked.
            await new Promise((resolve) => capture.onclick = resolve);

            const canvas = document.createElement('canvas');
            canvas.width = video.videoWidth;
            canvas.height = video.videoHeight;
            canvas.getContext('2d').drawImage(video, 0, 0);
            stream.getVideoTracks()[0].stop();
            div.remove();
            return canvas.toDataURL('image/jpeg', quality);
        }
    ''')
    display(js)
    data = eval_js('takePhoto({})'.format(quality))
    binary = b64decode(data.split(',')[1])
    with open(filename, 'wb') as f:
        f.write(binary)
    return filename
```

```
[ ] from IPython.display import Image
try:
    filename = take_photo()
    print('Saved to {}'.format(filename))

    # Show the image which was just taken.
    display(Image(filename))
except Exception as err:
    # Errors will be thrown if the user does not have a webcam or if they do not
    # grant the page permission to access it.
    print(str(err))
```

Saved to photo.jpg



<최종 결과물의 예>

Predicted: (('n02124075', 'Egyptian\_cat', 0.7297629), ('n02123159', 'tiger\_cat', 0.14424737), ('n02123045', 'tabby', 0.107179

# 기존 모델 가져다쓰기

## part2 : 이론 & 사용법

# [KA] 모델 활용 방법들

- 해당 모델들은 imagenet을 기반으로 학습되어 있음 (weights='imagenet')
  - 학습된 모델 그대로를 사용하고 싶은 경우
  - 학습된 모델에서 마지막 layer를 제외하고 사용하고 싶은 경우
  - 학습된 모델의 특정 layer 출력을 사용하고 싶은 경우
  - 학습된 모델을 자신의 application에 맞게 Fine-Tune
  - 모델의 꺾데기만 사용하고 싶은 경우

# [KA] : 모델 그대로 사용하기

- 학습된 모델 그대로를 사용하고 싶은 경우
- `weights = 'imagenet', include_top=True`

```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
# Predicted: [(u'n02504013', u'Indian_elephant', 0.82658225), (u'n01871265', u'tusker', 0
```



# [KA] 특징만 추출하고 싶을때

- 학습된 모델에서 특징만 추출하고 싶은 경우
- weights = 'imagenet', include\_top=False

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import numpy as np

model = VGG16(weights='imagenet', include_top=False)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
```



# [KA] 특징만 추출하고 싶을때

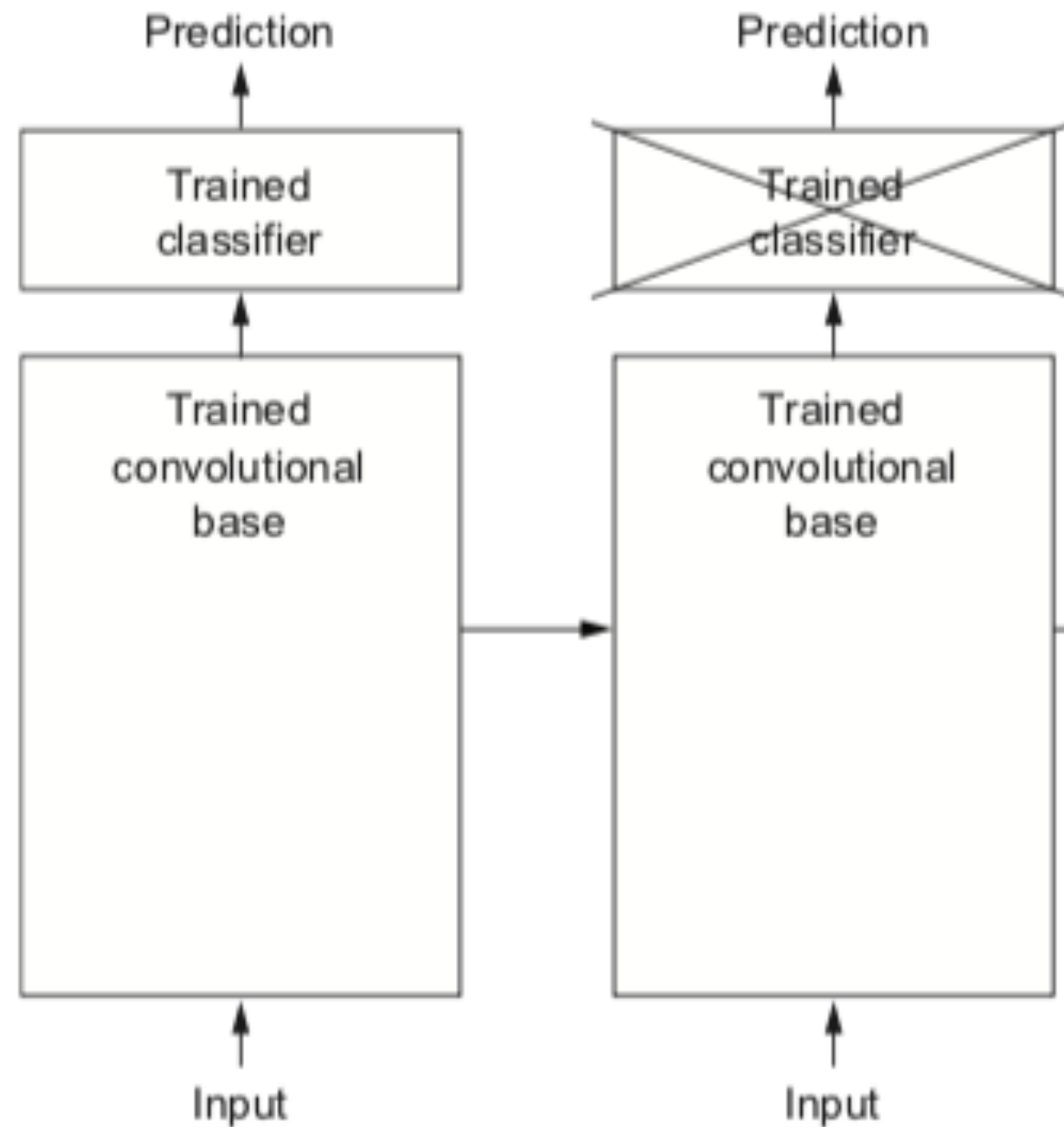


Figure 5.14 Swapping classifiers while keeping the :

# [KA] 특정 layer의 출력을 추출

- 학습된 모델의 특정 layer 출력을 사용하고 싶은 경우
- weights = 'imagenet', include\_top=True
- model = Model(inputs=..., outputs=...)

```
from keras.applications.vgg19 import VGG19
from keras.preprocessing import image
from keras.applications.vgg19 import preprocess_input
from keras.models import Model
import numpy as np

base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input, outputs=base_model.get_layer('block4_pool').output)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

block4_pool_features = model.predict(x)
```

# [KA] 모든 layer의 출력을 추출

- 학습된 모델의 모든 layer 출력을 얻고 싶은 경우
- weights = 'imagenet', include\_top=True
- model = Model(inputs=..., outputs=...)

```
model = VGG19(weights='imagenet')
```

```
from keras.models import Model
layer_outputs = [layer.output for layer in model.layers]
activation_model = Model(inputs=model.input, outputs=layer_outputs)
activations = activation_model.predict(x)
```

# [KA] 맞춤형 모델 만들기

- 학습된 모델을 자신의 application에 맞게 Fine-Tune
- weights = 'imagenet', include\_top=False

```
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing import image
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras import backend as K

# create the base pre-trained model
base_model = InceptionV3(weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(1024, activation='relu')(x)
# and a logistic layer -- let's say we have 200 classes
predictions = Dense(200, activation='softmax')(x)

# this is the model we will train
model = Model(inputs=base_model.input, outputs=predictions)
```

# [KA] 모델 Fine-Tune

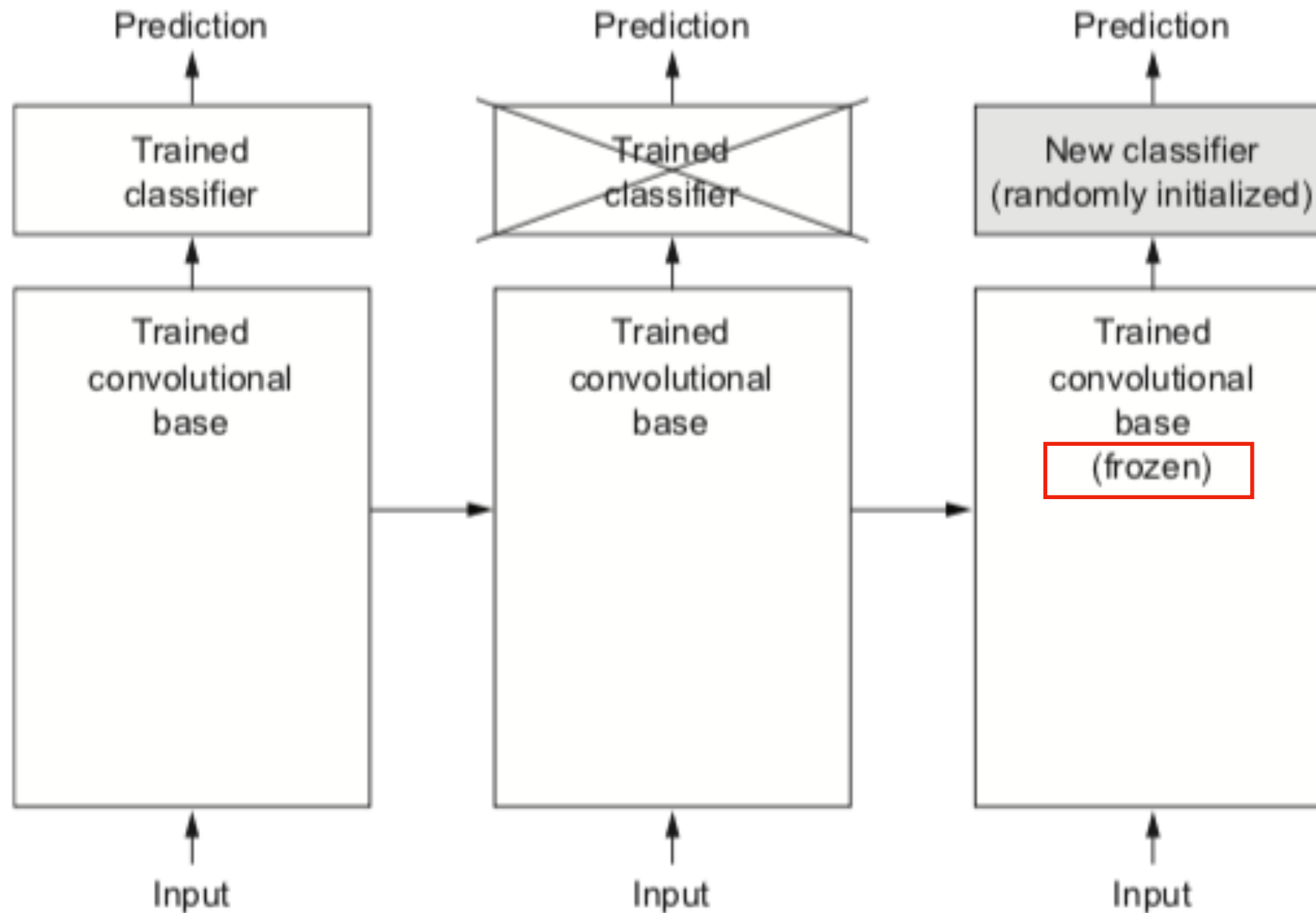


Figure 5.14 Swapping classifiers while keeping the same convolutional base

# [KA] 모델 Fine-Tune

- 학습된 모델을 자신의 application에 맞게 Fine-Tune
- weights = 'imagenet', include\_top=False

```
# first: train only the top layers (which were randomly initialized)  
# i.e. freeze all convolutional InceptionV3 layers  
for layer in base_model.layers:  
    layer.trainable = False  
  
# compile the model (should be done *after* setting layers to non-trainable)  
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
```

# [KA] 빈 컵 데기 모델

- 모델의 껍데기만 사용하고 싶은 경우
- `weights = None, include_top=True`

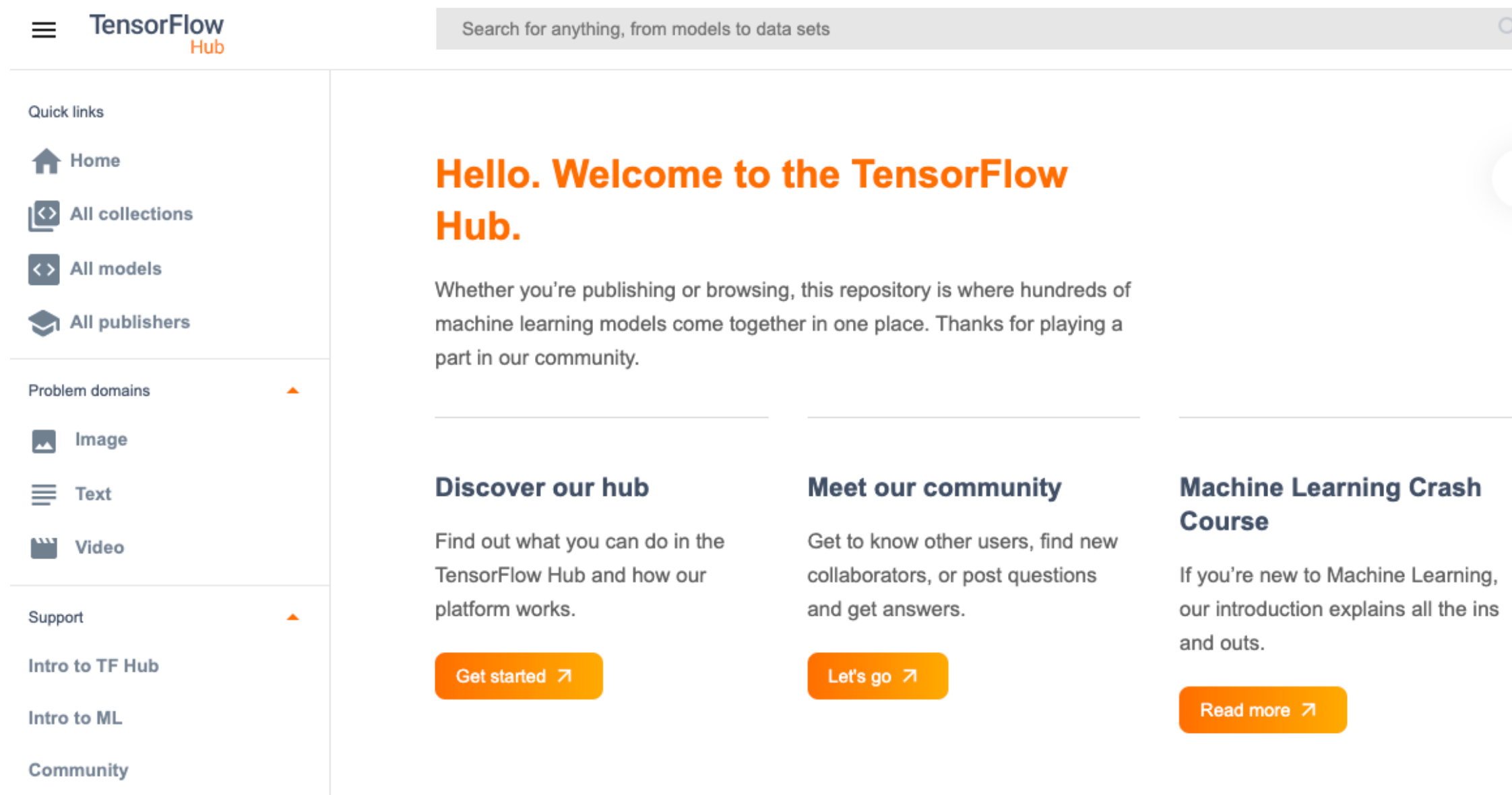
기존 모델 가져다 쓰기

**Part3 : tf-hub**



# Tensorflow Hub (tfhub.dev)

- 앱스토어 처럼 학습된 네트워크를 제공하는 학습된 모델을 제공
- 이러한 모델을 잘 가져다 쓰는 것도 실력 (향후에는 일반화 될 것으로 예상)



# Tensorflow Hub (tfhub.dev)

- 앱스토어 처럼 학습된 네트워크를 제공하는 학습된 모델을 제공
- 이러한 모델을 잘 가져다 쓰는 것도 실력

```
import tensorflow_hub as hub
import tensorflow as tf

model = hub.load("https://tfhub.dev/captain-pool/esrgan-tf2/1")

# To add an extra dimension for batch, use tf.expand_dims()

low_resolution_image = load_image() # Low Resolution Image of shape [batch_size, height, width, 3]
low_resolution_image = tf.cast(low_resolution_image, tf.float32)
super_resolution = model(low_resolution_image) # Perform Super Resolution here
```

# Tensorflow Hub (tfhub.dev)

- 앱스토어 처럼 학습된 네트워크를 제공하는 학습된 모델을 제공
- 이러한 모델을 잘 가져다 쓰는 것도 실력

```
import tensorflow_hub as hub
import tensorflow as tf

generator = tf.keras.models.Sequential([
    hub.KerasLayer("https://tfhub.dev/captain-pool/esrgan-tf2/1", trainable=True),
    tf.keras.layers.Conv2D(filters=3, kernel_size=[1, 1], strides=[1, 1])
])
```

# Tensorflow Hub (tfhub.dev)

- 더 많은 예제는 [github.com/tensorflow/hub](https://github.com/tensorflow/hub) 에서 examples를 참고
- [https://github.com/tensorflow/hub/blob/master/examples/colab/object\\_detection.ipynb](https://github.com/tensorflow/hub/blob/master/examples/colab/object_detection.ipynb)

TensorFlow Hub

rcnn

← faster\_rcnn/openimages\_v4/inception\_resnet\_v2

Problem domain

Object detection

Publisher

Google

Architecture

R cnn

Data set

Openimagesv4

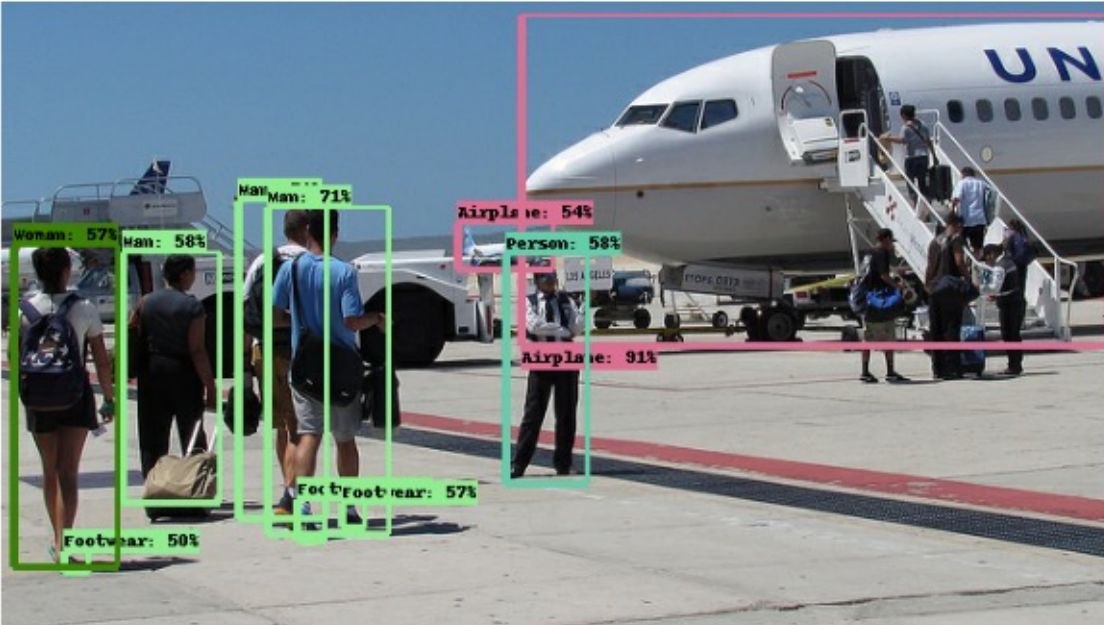
Fine tunable: No

Last updated: 10/04/2019

FasterRCNN+InceptionResNetV2 network trained on [Open Images V4](#).

The module performs non-maxima suppression inside the module. The maximal number of detection outputted is 100. Detections are outputted for 600 boxable categories.

An example detection result is shown below.



# 실습

- 웹캠 + TF Hub의 모델을 받아서 응용 만들기