

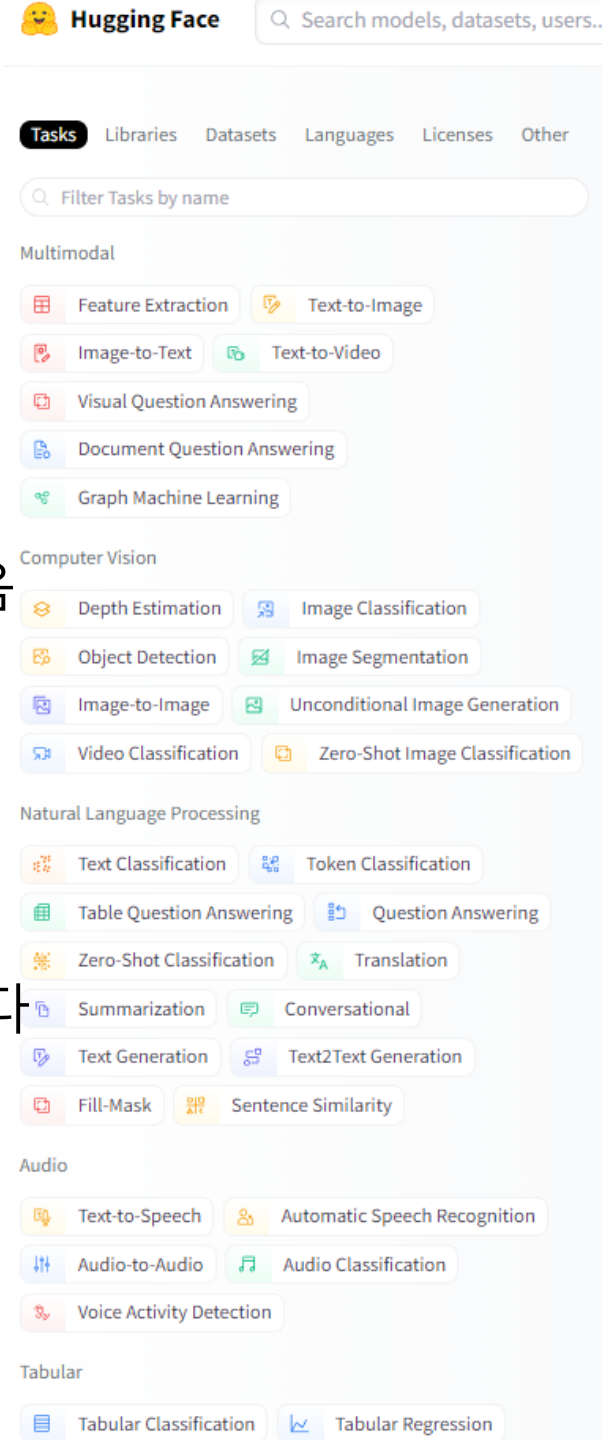


HuggingFace 사전학습모델

최석재 lingua@naver.com

HuggingFace 모델

- 2016년 프랑스 기업인에 의해 설립된 미국 기업
- 파이토치와 트랜스포머를 기반으로 많은 모델들을 만들어 공개하고 있음
- 대용량 컴퓨터 자원을 이용하여 모델을 만듦
- HuggingFace 모델은 인터페이스가 통일되어 있다는 것이 가장 큰 장점
- HuggingFace를 이용해 이미지를 간단히 예측할 수 있는 모델을 알아본다



구글 드라이브 연결

- 이미지 관련 주요 모델을 알아본다
- 이미지 처리 시에는 torchvision이 필요한 경우가 많으니 설치한다
- 이어 구글 드라이브에 연결한다
- `!pip install transformers torchvision`
- `from google.colab import drive`
- `drive.mount('/content/gdrive')`

이미지 출력

- 분석 대상 이미지를 출력해본다
- `from IPython.display import display`
- `from IPython.display import Image as _Imgdis`
- `from PIL import Image`
- `path = '/content/gdrive/MyDrive/pytest_img/opencv/pen_coffee.png'`
- `display(_Imgdis(filename=path, width=600, height=400))`



이미지 분류

- 이미지 분류 파이프라인은 이미지가 전체적으로 어느 카테고리에 속하는지를 판정한다
- `from transformers import pipeline`
- `from keras.preprocessing.image import array_to_img, img_to_array, load_img`
- `image = load_img(path)` # 이미지 로딩
- `classifier = pipeline("image-classification")` # 이미지 분류 파이프라인 생성
- `results = classifier(image)` # 이미지 분류 실행
- `print(results)` # 결과 출력

출력 결과

- 첫 부분에 모델이 특정되지 않아 기본 모델이 사용되었음을 알리고 있다
 - 허깅페이스가 기본으로 제공하는 모델이 사용되며,
 - 특정 모델을 사용할 경우 다음과 같이 사용한다
- `classifier = pipeline(model="microsoft/beit-base-patch16-224-pt22k-ft22k")`

```
No model was supplied, defaulted to google/vit-base-patch16-224 and revision 5dca96d (https://huggingface.co/google/vit-base-patch16-224).
```

```
Using a pipeline without specifying a model name and revision in production is not recommended.
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
```

```
The secret `HF_TOKEN` does not exist in your Colab secrets.
```

```
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
```

```
You will be able to reuse this secret in all of your notebooks.
```

```
Please note that authentication is recommended but still optional to access public models or datasets.
```

```
warnings.warn(
```

```
config.json: 100%  69.7k/69.7k [00:00<00:00, 1.42MB/s]
```

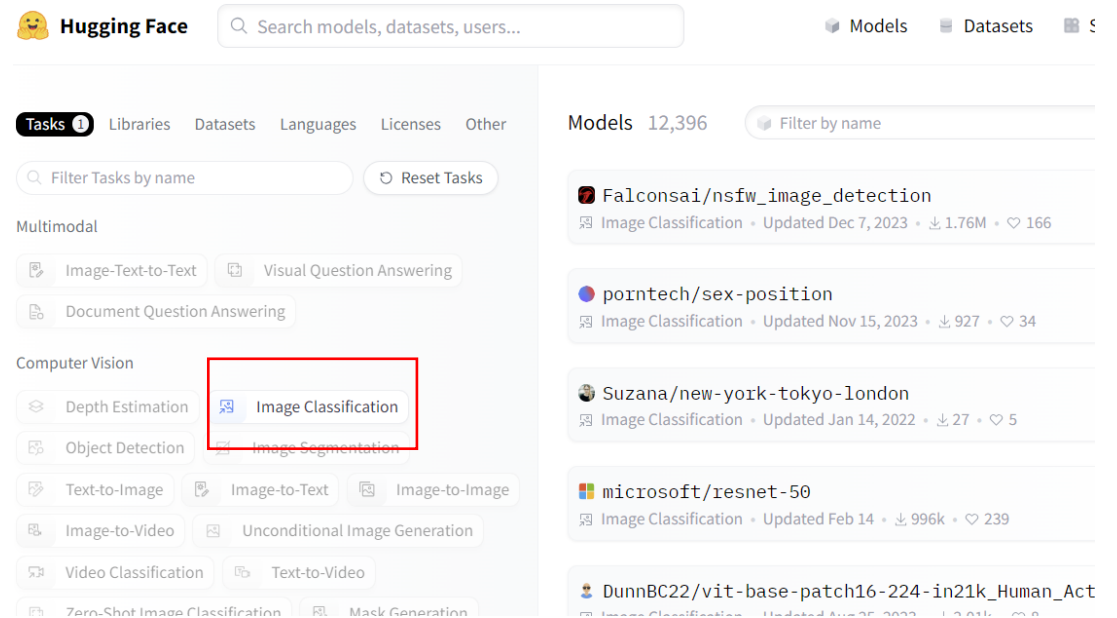
```
model.safetensors: 100%  346M/346M [00:03<00:00, 115MB/s]
```

```
preprocessor_config.json: 100%  160/160 [00:00<00:00, 4.89kB/s]
```

```
[{'label': 'cup', 'score': 0.23439309000968933}, {'label': 'espresso', 'score': 0.23302948474884033}, {'label': 'letter opener, paper knife, paperknife', 'score': 0.10450924932956696}, {'la
```

모델 찾기

- HuggingFace 모델 허브 (<https://huggingface.co/models>)로 들어가
- Image Classification을 눌러 나오는 모델을 사용할 수 있다
- 모델의 세부 사용 방식이 달라 엉뚱한 결과가 나오거나 바로 진행되지 않는 경우도 있을 수 있다



결과 정리

- `import pandas as pd`

결과를 DataFrame으로 변환

- `results_df = pd.DataFrame(results)`

- `print(results_df)`

	label	score
0	cup	0.234393
1	espresso	0.233029
2	letter opener, paper knife, paperknife	0.104509
3	quill, quill pen	0.078224
4	coffee mug	0.071259

이미지는 컵 또는 에스프레소일 확률이 23%로 판정되었다

객체 감지

- 객체 감지는 이미지에 있는 각 객체의 위치를 파악하는 것이 주 목적이다
 - 먼저 파이토치 모델에 사전훈련된 모델과 구성을 연결하는
 - timm 라이브러리를 설치해야 한다
 - 설치가 끝나면 런타임 > 세션 다시 시작
-
- !pip install timm
 - # 설치 후 런타임 > 세션 다시 시작

이미지 출력

- 세션이 다시 시작되어 이미지 경로 등을 다시 입력해야 한다
- `from IPython.display import display`
- `from IPython.display import Image as _Imgdis`
- `from PIL import Image`
- `path = '/content/gdrive/MyDrive/pytest_img/opencv/pen_coffee.png'`
- `display(_Imgdis(filename=path, width=600, height=400))`



객체 감지

- from transformers import pipeline
- from keras.preprocessing.image import array_to_img, img_to_array, load_img
- image = load_img(path)
- detector = pipeline("object-detection") # 객체 감지 파이프라인 생성
- results = detector(image) # 객체 감지 실행
- print(results)

```
No model was supplied, defaulted to facebook/detr-resnet-50 and revision 2729413 (https://huggingface.co/facebook/detr-resnet-50).
Using a pipeline without specifying a model name and revision in production is not recommended.
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(

model.safetensors: 100%  102M/102M [00:01<00:00, 114MB/s]

Some weights of the model checkpoint at facebook/detr-resnet-50 were not used when initializing DetrForObjectDetection: ['model.back
- This IS expected if you are initializing DetrForObjectDetection from the checkpoint of a model trained on another task or with ano
- This IS NOT expected if you are initializing DetrForObjectDetection from the checkpoint of a model that you expect to be exactly i

preprocessor_config.json: 100%  290/290 [00:00<00:00, 14.4kB/s]

[{'score': 0.9935367703437805, 'label': 'knife', 'box': {'xmin': 216, 'ymin': 88, 'xmax': 314, 'ymax': 105}}, {'score': 0.9983748197
```

결과 정리

- `import pandas as pd`

결과를 DataFrame으로 변환

- `results_df = pd.DataFrame(results)`
- `print(results_df)`

```
   score  label  box
0  0.993537  knife {'xmin': 216, 'ymin': 88, 'xmax': 314, 'ymax': ...
1  0.998375   cup  {'xmin': 218, 'ymin': 9, 'xmax': 270, 'ymax': 62}
2  0.997725  book  {'xmin': 9, 'ymin': 45, 'xmax': 281, 'ymax': 222}
```

이미지 분할

- 이미지를 픽셀 단위로 분석하여 각 객체를 파악한다
- 객체 감지와 유사하나, 픽셀 단위 분석으로 더 정확한 정보를 제공한다
- `from transformers import pipeline`
- `from keras.preprocessing.image import array_to_img, img_to_array, load_img`
- `image = load_img(path)`
- `segmer = pipeline("image-segmentation")` # 이미지 분할 파이프라인 생성
- `results = segmer(image)` # 이미지 분할 실행
- `print(results)`

```
No model was supplied, defaulted to facebook/detr-resnet-50-panoptic and revision fc15262 (https://huggingface.co/facebook/detr-resnet-50-panoptic).
Using a pipeline without specifying a model name and revision in production is not recommended.
Some weights of the model checkpoint at facebook/detr-resnet-50-panoptic were not used when initializing DetrForSegmentation: ['detr.model.backbone.c
- This IS expected if you are initializing DetrForSegmentation from the checkpoint of a model trained on another task or with another architecture (e
- This IS NOT expected if you are initializing DetrForSegmentation from the checkpoint of a model that you expect to be exactly identical (initiali
[{'score': 0.988335, 'label': 'knife', 'mask': <PIL.Image.Image image mode=L size=326x223 at 0x7A4722585720>}, {'score': 0.99682, 'label': 'LABEL_189
```

결과 정리

- `import pandas as pd`
- `pd.set_option('display.max_colwidth', None)` # 판다스 긴 문자열 출력

결과를 DataFrame으로 변환

- `results_df = pd.DataFrame(results)`
- `print(results_df)`

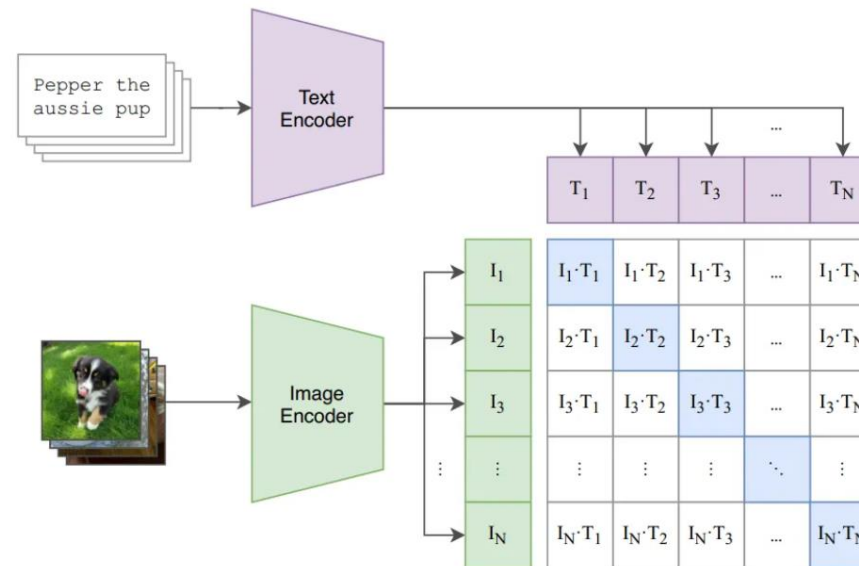
```
                                mask  
0  <PIL.Image.Image image mode=L size=326x223 at 0x7E80507A9000>  
1  <PIL.Image.Image image mode=L size=326x223 at 0x7E80507AAA40>  
2  <PIL.Image.Image image mode=L size=326x223 at 0x7E80507A9360>
```

이미지 예측

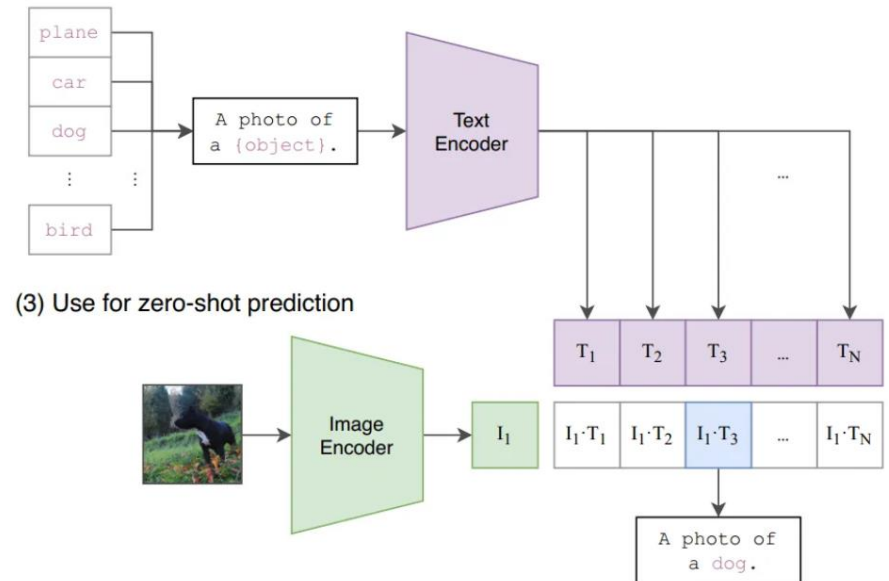
CLIP

- 2021년 OpenAI가 발표한 CLIP은 이미지와 텍스트를 결합하여 학습시킨다
- 텍스트와 이미지를 결합해 모델을 구축함으로써 두 모달리티 간의 의미적 관계를 이해한다
- 이미지와 텍스트를 매칭하려 할 때 사용할 수 있다
- DALL-E와 유사하나, 이미지 또는 텍스트를 생성할 수는 없다

(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction

구글 드라이브와 연결

```
# from google.colab import auth  
# auth.authenticate_user()
```

- from google.colab import drive
- drive.mount('/content/gdrive')

트랜스포머 설치

- CLIP은 비전 트랜스포머(Vision Transformer) 방식으로 학습된 모델을 사용한다
- 먼저 허깅페이스의 트랜스포머를 설치한다
- `!pip install transformers`









이미지 로드

- 예측 대상이 되는 이미지를 불러와 본다
- 여기서는 PIL의 open() 함수를 이용해본다
- keras의 load_img()와 같은 PIL 객체를 리턴하지만 보다 간단히 사용할 수 있다
- from PIL import Image
- file = '/content/gdrive/MyDrive/pytest_img/cats_dogs/cat.6.jpg'
- image = Image.open(file)
- image



모델 다운로드

- from transformers import CLIPProcessor, CLIPModel
- model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
- processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")

```
config.json: 100%  4.19k/4.19k [00:00<00:00, 113kB/s]
pytorch_model.bin: 100%  605M/605M [00:05<00:00, 157MB/s]
/usr/local/lib/python3.10/dist-packages/torch/_utils.py:831: UserWarning: TypedStorage is deprecated.
  return self.fget.__get__(instance, owner)()
preprocessor_config.json: 100%  316/316 [00:00<00:00, 12.9kB/s]
tokenizer_config.json: 100%  592/592 [00:00<00:00, 24.4kB/s]
vocab.json: 100%  862k/862k [00:00<00:00, 3.28MB/s]
merges.txt: 100%  525k/525k [00:00<00:00, 4.06MB/s]
tokenizer.json: 100%  2.22M/2.22M [00:00<00:00, 6.64MB/s]
special_tokens_map.json: 100%  389/389 [00:00<00:00, 14.3kB/s]
```

후보 텍스트 생성

- 이미지와 관련이 있을 후보 텍스트를 생성한다
- candidates = ['a cat is eating a meal', 'a photo of a cat', 'a photo of a dog', 'a cat is lying with woman', 'a cat is lying with woman on the bed']
- inputs = processor(text=candidates, images=image, return_tensors='pt', padding=True)

모델에 데이터 주입

- `model.eval()`
- `outputs = model(**inputs)` **inputs와 같이 하면 inputs가 가지고 있는 키-값 쌍을 model에 자동으로 전달된다 (딕셔너리 언패킹)
(예: `input_ids`, `attention_mask`, `token_type`, `ids`, `special_tokens_mask` 등의 정보)
- `outputs.keys()`

```
odict_keys(['logits_per_image', 'logits_per_text', 'text_embeds', 'image_embeds', 'text_model_output', 'vision_model_output'])
```

- `logits_per_image`: 이미지와 각 텍스트 입력 사이의 유사성 점수
- `logits_per_text`: 텍스트와 각 이미지 입력 사이의 유사성 점수
- `text_embeds`: 이미지 입력의 임베딩
- `text_model_output`: 텍스트 처리 후의 언어 모델 내부 표현
- `vision_model_output`: 비전 처리 후의 비전 모델 내부 표현

추론 점수

- logits_per_image = outputs.logits_per_image
- print(logits_per_image)

```
tensor([[19.7183, 24.0501, 19.4739, 27.2155, 28.7516]], grad_fn=<TBackward0>)
```

Softmax 변환

- Softmax 함수로 확률 형태로 변환한 뒤,
 - 가장 높은 확률을 가지는 텍스트를 출력한다
-
- `import torch`
 - `probs = logits_per_image.softmax(dim=1)`
 - `print(candidates[torch.argmax(probs).item()])`
`a cat is lying with woman on the bed`