

305 – Unveiling the Illusion

Team Information

Team Name: kimbabasaksaksak

Team Member: Jaeheon Kim, Donghyun Kim, Soyoung Yoo, Minhee Lee

Email Address: uaaooong@gmail.com

Instructions

Description DFC has been hosting an annual photography exhibition contest consistently each year. After last year's winning entry was revealed to be created by an AI, to avoid repeating the same mistake, they have entrusted you with the task of determining whether the submitted artworks are generated by human or AI. Please discern whether each submitted artwork is a human-generated or an AI-generated creation.

Target	Hash (MD5)
ImageSet.ad1	083C11A3D77C073AC1A4A32AF1230531

Questions

- 1) Explain the approach for detecting AI-generated creation and submit the code for it. (100 points)
 - We provide you with the information about the photos submitted for the 2022 DFC Contest, indicating which ones were revealed to be generated by human and generated by AI.
 - Scores will be assigned based on the "F1-Score" when using the submitted code to classify the photos of the artworks for the 2022 DFC contest.

- Additional points may be awarded for the submitted code and ideas.
- 2) Use the method devised in the previous question to determine whether each image has been generated by AI or not. (200 points)
- We provide you with the photos of the artworks submitted for the 2023 DFC Contest.
 - Scores will be assigned based on "F1-Score" when using the submitted code to classify the photos of the artworks for the 2023 DFC contest.

Teams must:

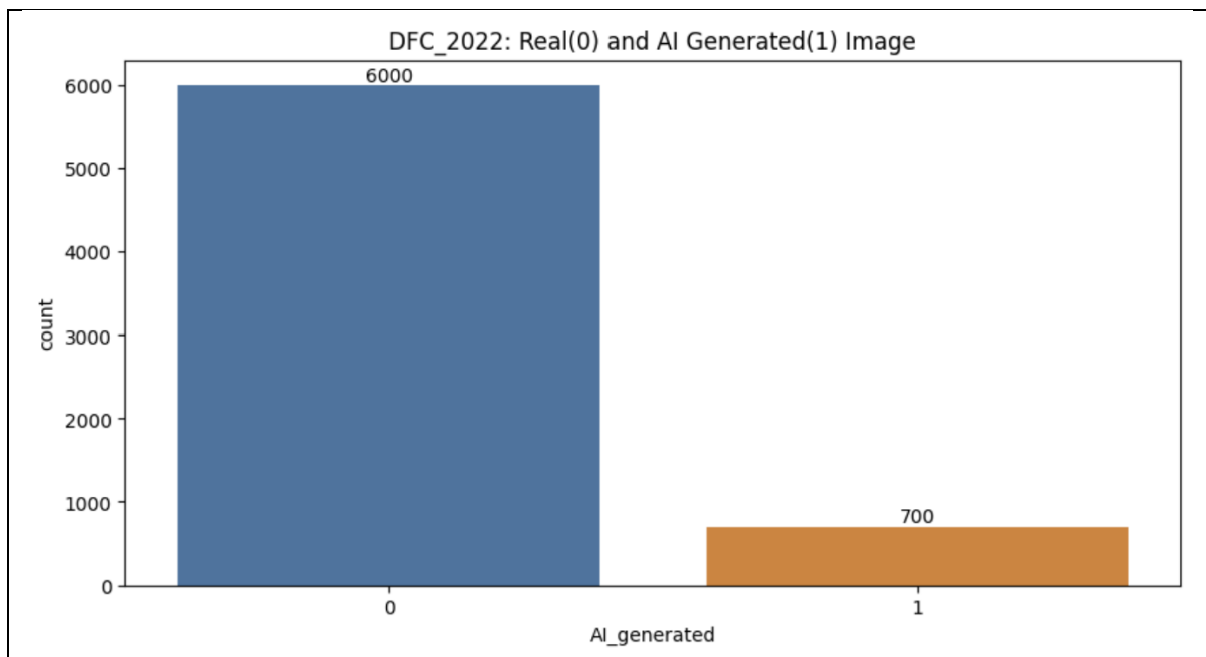
- Develop and document the step-by-step approach used to solve this problem to allow another examiner to replicate team actions and results.
- Specify all tools used in deriving the conclusion(s).

Step-by-step methodology:

Q1. Explain the approach for detecting AI-generated creation and submit the code for it. (100 points)

모델을 생성하기 앞서, 학습 데이터로 사용할 2022_DFC 이미지 데이터 분석을 진행하였다. 정확도가 높은 모델을 만들기 위해 해결해야 할 포인트를 2가지 발견할 수 있었다.

첫번째 포인트는 데이터 불균형이다. 이 문제의 경우, AI가 생성한 이미지와 실제 이미지를 구분하는 분류 문제다. 분류 문제를 해결하기 위해서는 학습에 사용되는 데이터의 클래스별 분포가 중요하다. 2022_DFC 데이터에서 이미지 분포를 확인해본 결과, 실제 이미지(6000개, 약 89.6%)에 비해 AI 생성 이미지(700개, 약 10.4%)의 비중이 매우 작은 것을 확인할 수 있었다. 이러한 데이터셋을 이용해 모델을 생성하는 경우, AI 생성 이미지에 대한 학습을 충분히 하지 못하게 된다.



[그림 1] 2022_DFC 데이터의 AI 생성 이미지와 실제 이미지 비율

이를 해결하기 위해 학습 데이터를 추가로 수집하여 균형을 맞춰주었다. 라벨링이 되어있는 AI, 실제 이미지 데이터셋*을 학습 데이터셋에 추가하여 실제 이미지와 AI 생성 이미지의 비율을 56.4%, 43.6%으로 비교적 균등하게 맞춰주었다.

* 데이터 출처: <https://huggingface.co/datasets/competitions/aiornot>

(해당 데이터의 train 데이터 셋만을 이용)

두번째 포인트는 데이터 셋의 크기이다. 2022_DFC 데이터의 경우, 6700건이며 추가로 수집한 데이터를 합쳐도 약 25000건 정도이다. 학습 데이터 셋의 크기가 작은 경우, 모델 학습이 성공적으로 이루어지더라도 학습시키지 않은 새로운 데이터셋에 대해 좋지 않은 예측 결과를 보이는 과적합 문제가 발생할 수 있다. 모델의 정확도를 높이기 위해서는 더 다양하고 많은 데이터셋의 학습이 필요하다. 이를 위해 이미지에 변형을 주는 증강 기법으로 학습 데이터 셋의 규모를 키우고 과적합을 방지하도록 하였다. 또한, 전이학습으로 대량의 이미지 데이터를 학습한 사전 훈련 모델에 ai-real 데이터 학습 layer를 추가해 해당 모델의 이미지 분류 지식을 ai-real 이미지 분류에 이용해 부족한 데이터 문제를 해결하고자 하였다.

다양한 사전 훈련 모델 중, 가장 성능이 좋은 모델을 찾기 위해 총 10 종류의 모델에 2022_DFC 데이터를 이용하여 전이 학습을 수행하고 f1 score를 계산해보았다. 아래는 각 모델의 f1 score이며 Swin-Transformer가 가장 높은 점수를 보이는 것을 확인할 수 있었다.

Model	F1 score (2022_DFC 데이터)
Swin-Transformer	0.9513
ResNet50V2	0.8556
DenseNet201	0.8392
MobileNetV2	0.8148
ResNet152V2	0.8104
InceptionV3	0.6239
MobilNetV3Large	0.3568
DenseNet121	0.3210
ResNet50	0.3048
EfficientNetV2B0	0.0000

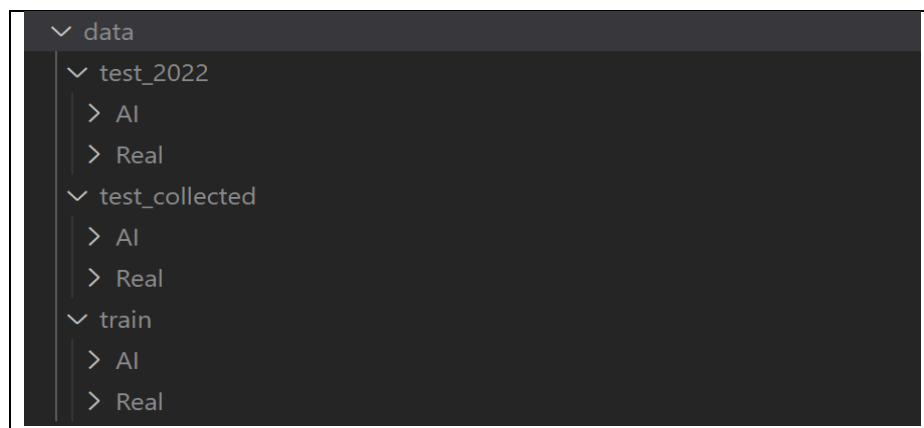
Swin-transformer는 비전 작업을 위한 딥러닝 아키텍처로 이미지를 작은 패치로 분할하여 패치 간 계층적 관계를 학습한다. 각 패치의 특징과 주변 패치와의 관계를 파악하며 계층적으로 그룹화하고 처리하여 이미지의 작은 특징부터 큰 특징까지 다양한 측면을 캡처할 수 있다. 이를 통해 실제 이미지와 구분하기 힘든 ai 생성 이미지의 특징을 다양하게 파악하여 학습할 수 있을 것이라고 보았다. 또한, 비교적 경량화 된 모델로 다른 주요 이미지 인식 아키텍처(CNN, ResNet 등)에 비해 작은 파라미터 크기로도 좋은 성능을 내는 경향이 있어 해당 아키텍처 기반 사전 훈련 모델을 사용하였다.

이러한 2가지 포인트를 중점으로 고려하여 ai 생성 이미지와 실제 이미지를 분류하는 모델을

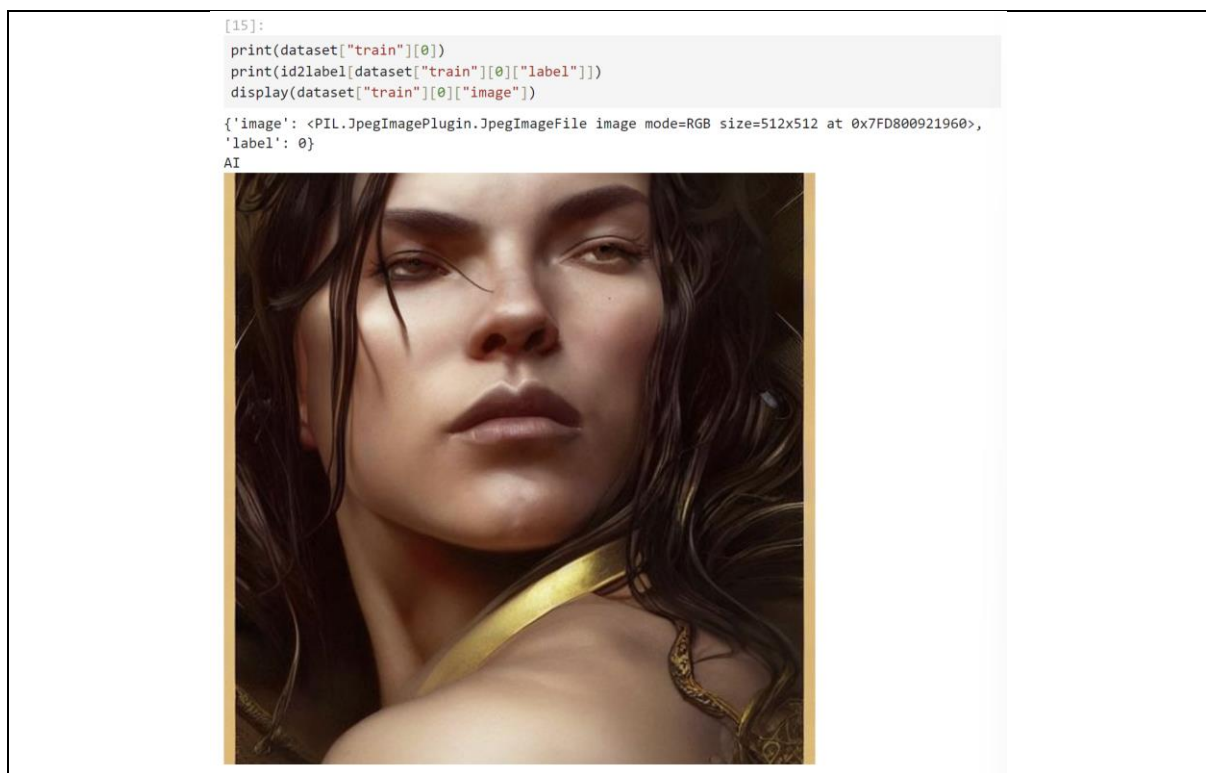
생성하였다.

1. 데이터 로드

라벨링 csv 파일을 기반으로 train, test 이미지 데이터를 AI, Real 폴더로 분리하여 데이터 폴더를 생성하였다. (create_dataset_folder.py 참조) 이후, datasets 패키지의 load_dataset 함수를 이용해 데이터셋 디렉토리를 생성하였다. 데이터 폴더의 루트 디렉토리를 지정하면 폴더명을 기반으로 train, test 데이터셋을 분리하고 train, test 안의 폴더명을 통해 이미지 데이터의 라벨링을 수행한다.



[그림 2] 데이터 폴더 구조



[그림 3] 로드된 데이터 형식

2. 데이터 전처리

로드된 이미지 데이터를 모델에 입력 가능한 형태로 전처리한다. 학습 데이터를 사전 훈련된 모델의 입력 형식과 맞추기 위해 해당 모델의 이미지 처리기를 로드한다. 학습 데이터를 훈련 데이터(90%)와 검증 데이터(10%)로 분리하고 전처리하였다. 훈련 데이터의 경우, 이미지 내의 임의의 위치에서 입력 사이즈만큼 잘라내는 RandomResizedCrop, 이미지를 수평으로 뒤집는 RandomHorizontalFlip 기법을 사용해 데이터 증강을 수행하였다. 이후, 이미지 처리기에서 추출한 평균과 표준 편차를 이용해 정규화를 진행하였다. 생성한 데이터셋 디렉터리에 전처리된 이미지 데이터인 pixel_values 값이 추가된 것을 확인할 수 있다.

```
[18]:
train_ds[0]

[18]:
{'image': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=512x768>,
 'label': 1,
 'pixel_values': tensor([[[ 1.7352,  1.7009,  1.6667, ...,  1.7865,  1.8208,  1.8379],
 [ 1.7523,  1.7180,  1.6838, ...,  1.8722,  1.9064,  1.9407],
 [ 1.7523,  1.7352,  1.7009, ...,  1.8893,  1.9407,  1.9920],
 ...,
 [ 1.4954,  1.4612,  1.4612, ...,  1.8208,  1.8208,  1.8037],
 [ 1.6495,  1.5810,  1.5468, ...,  1.8208,  1.8208,  1.8037],
 [ 1.7865,  1.7180,  1.6667, ...,  1.8379,  1.8037,  1.7865]],
 [[ 0.0476,  0.0476,  0.0301, ...,  1.2556,  1.2906,  1.3081],
 [ 0.0651,  0.0476,  0.0126, ...,  1.3256,  1.3606,  1.3957],
 [ 0.0651,  0.0476,  0.0126, ...,  1.3081,  1.3606,  1.4132],
 ...,
 [-0.0049, -0.0224, -0.0574, ...,  0.9230,  0.9055,  0.8704],
 [ 0.1176,  0.0651, -0.0049, ...,  0.9055,  0.8880,  0.8529],
 [ 0.2577,  0.1877,  0.1001, ...,  0.9230,  0.8880,  0.8529]],
 [[ 0.6705,  0.6531,  0.6182, ...,  0.4091,  0.4614,  0.5136],
 [ 0.6879,  0.6531,  0.6182, ...,  0.4439,  0.4962,  0.5659],
 [ 0.6705,  0.6531,  0.6182, ...,  0.4091,  0.4614,  0.5311],
 ...,
 [-0.3927, -0.4798, -0.5147, ..., -0.1312, -0.1661, -0.2184],
 [-0.2532, -0.3753, -0.4450, ..., -0.1661, -0.2010, -0.2532],
 [-0.1138, -0.2358, -0.3230, ..., -0.1487, -0.2010, -0.2707]]]])}
```

[그림 4] 전처리된 데이터 형식

3. 사전 훈련 모델 로드 및 파인튜닝

Swin-transformer 아키텍처 기반의 이미지 분류 사전 학습 모델인 **microsoft/swin-small-patch4-window7-224** (<https://huggingface.co/microsoft/swin-small-patch4-window7-224>)를 로드하고 준비된 데이터를 이용해 학습시켰다. 모델 로드 과정에서 학습시키고자 하는 라벨 id 디렉토리를 사전 훈련 모델의 체크포인트와 함께 전달한다. 이를 통해 훈련 가중치가 없는 새로운 사용자 정의 classification head를 생성한다. 사전 훈련 모델을 AI 생성 이미지와 Real 이미지를 분류하는 모델로 업데이트할 수 있도록 변형하는 것이다.

```
def create_model(model_checkpoint):  
    model = AutoModelForImageClassification.from_pretrained(  
        model_checkpoint,  
        label2id=label2id,  
        id2label=id2label,  
        ignore_mismatched_sizes = True,  
    ).to("cuda")  
    return model
```

[그림 5] image_classification_finetune.ipynb 중 사전 학습 모델을 로드하는 부분

해당 모델을 어떻게 학습시킬지 설정하기 위해서는 trainer 인자를 지정하여 trainer를 생성해야 한다. 각 epoch의 끝에서 학습에 대한 평가를 수행하도록 설정하고 learning rate, batch size, epoch 수 등을 정의하였다. 마지막 훈련된 모델이 항상 최고의 성능을 보이는 모델이 아닐 수 있으므로 훈련 끝에 최적의 모델을 불러올 수 있도록 설정하였다. trainer 인자를 조정해 나가며 최적의 모델을 만들기 위한 값을 찾아 학습을 진행하였다.

```
def trainer_args(model_checkpoint):  
    model_name = model_checkpoint.split("/")[-1]  
    args = TrainingArguments(  
        f"{model_name}-finetuned",  
        remove_unused_columns=False,  
        evaluation_strategy = "epoch",  
        save_strategy = "epoch",  
        learning_rate=5e-5,  
        per_device_train_batch_size=batch_size,  
        gradient_accumulation_steps=4,  
        per_device_eval_batch_size=batch_size,  
        num_train_epochs=3,  
        warmup_ratio=0.1,  
        logging_steps=10,  
        load_best_model_at_end=True,  
        metric_for_best_model="accuracy",  
    )  
    return args
```

[그림 6] image_classification_finetune.ipynb 중 trainer 인자를 설정하는 부분

이렇게 학습시킨 **swin-small-patch4-window7-224-finetuned** 모델의 2022_DFC 데이터에 대한 예측 결과와 f1 score는 아래와 같다.

```
2022 DFC 분류 결과

[15]: pred_results_2022 = predict_test_data(dataset_2022, model_path, "2022_DFC")
pred_results_2022

/transformers/src/transformers/models/vit/feature_extraction_vit.py:28: FutureWarning: The class ViTFeatureExtractor is deprecated and will be removed in version 5 of Transformers. Please use ViTImageProcessor instead.
warnings.warn(
100% ██████████ 6700/6700 [10:55<00:00, 11.03it/s]

[15]:
```

	score	label	filename	answer
0	0.998410	0	./data/test_2022/AI/b20cf7b22fda326daf9be6cb3e...	0
1	0.999930	0	./data/test_2022/AI/2c6c97c46022fa444efd86ab35...	0
2	0.997870	0	./data/test_2022/AI/dbd037a0a9b7f7f41cb942a1c5...	0
3	0.998603	0	./data/test_2022/AI/dbafc892428e73549319216707...	0
4	0.997755	0	./data/test_2022/AI/f89e6a6a2b9b32e47d051e4cbe...	0
...
6695	0.923272	1	./data/test_2022/Real/eb0abdacd3a3218d74285a18...	1
6696	0.999262	1	./data/test_2022/Real/f56caa3b6dfc10ebd216d670...	1
6697	0.987332	1	./data/test_2022/Real/7735a40e6e14fcb9142d64d0...	1
6698	0.999621	1	./data/test_2022/Real/82567bc242531a2f3508d4de...	1
6699	0.999975	1	./data/test_2022/Real/120f36da372a976a0e7eb851...	1

```
6700 rows x 4 columns

[16]: results = f1_metric.compute(references=pred_results_2022["answer"], predictions=pred_results_2022["label"])
results

[16]: {'f1': 0.9719866358262658}
```

[그림 7] swin-small-patch4-window7-224-finetuned 모델로 2022_DFC 데이터를 분류한 결과

4. 사전 학습되지 않은 테스트 데이터 셋(test_collected) 구성

2023년도 데이터의 경우, 라벨링이 되어있지 않아 올바르게 분류가 되었는지 확인이 불가능하다. 학습하지 않은 데이터 셋에 대해서도 높은 정확도를 보이는 범용적 모델을 만들어야 2023년 데이터에 대해서도 좋은 예측 결과를 얻을 수 있을 것이다. 추가로 테스트용 데이터를 수집하여 **swin-small-patch4-window7-224-finetuned** 모델의 성능을 확인하였다.

테스트 데이터셋(test_collected)의 구성은 아래와 같다. AI 이미지의 경우, 이미지 생성 AI 모델 중 가장 많이 사용되는 3가지 모델(Stable Diffusion, Midjourney, DALL-E)로 생성된 이미지를 사용하였다.

1. ai 생성 이미지: 1500개

Stable Diffusion, Midjourney, DALL-E 모델로 생성한 이미지

- 출처 1: <https://www.kaggle.com/datasets/rturley/stable-diffusion-100k-custom-prompts-and-images?select=images>

- 출처 2: <https://huggingface.co/datasets/poloclub/diffusiondb/tree/main/images>
- 출처 3: <https://stablediffusionai.org/>
- 출처 4: <https://www.kaggle.com/datasets/nikbearbrown/midjourney-people-photography-week-of-april-9>
- 출처 5: <https://www.kaggle.com/datasets/gauravduttakiit/dalle-recognition-dataset>

2. real 이미지: 1700개

- celebrity 이미지

출처: <https://www.kaggle.com/datasets/j53t3r/celebahq?resource=download>

- landscape 이미지

출처: <https://github.com/universome/alis>

swin-small-patch4-window7-224-finetuned 모델의 테스트 데이터셋(test_collected)에 대한 예측 결과와 f1 score는 아래와 같다.

```

Collected test dataset 테스트

[20]: pred_results_test = predict_test_data(dataset_test, model_path, "collected_data")
pred_results_test

/transformers/src/transformers/models/vit/feature_extraction_vit.py:28: FutureWarning: The class ViTFeatureExtractor is deprecated and will be removed in version 5 of Transformers. Please use ViImageProcessor instead.
  warnings.warn(

100% ██████████ 3200/3200 [10.00<00:00, 6.26it/s]

[20]:
  score  label  filename  answer
0  0.986693   0  ./data/test_collected/Al/gwcddrhwnh.png  0
1  0.999952   0  ./data/test_collected/Al/duqanbbagl.png  0
2  0.953927   0  ./data/test_collected/Al/jthibeevsib.png  0
3  0.999886   0  ./data/test_collected/Al/9779a187933cea3af42ab...  0
4  0.989317   0  ./data/test_collected/Al/ceqdzlzkvhr.png  0
...  ...  ...  ...
3195 0.987701   1  ./data/test_collected/Real/47c27103c59fd1c30b0...  1
3196 0.757473   1  ./data/test_collected/Real/1f62969ca631996427b...  1
3197 0.998283   1  ./data/test_collected/Real/96103a5b122e70ce508...  1
3198 0.983512   1  ./data/test_collected/Real/99ddbb1e41f32aaf492...  1
3199 0.884418   1  ./data/test_collected/Real/05f12d6a22541f0b6ea...  1

3200 rows x 4 columns

[21]: results = f1_metric.compute(references=pred_results_test["answer"], predictions=pred_results_test["label"])
results

[21]: {'f1': 0.9086131386861314}

```

[그림 8] swin-small-patch4-window7-224-finetuned 모델로 test_collected 데이터를 분류한 결과

5. ai, real image를 학습한 사전학습 모델 파인튜닝

학습된 데이터(f1 score: 0.97)에 비해 학습되지 않은 새로운 데이터(f1 score: 0.90)에 대해서는 정확도가 떨어지는 것을 확인할 수 있었다. 이를 해결하고자 단순 이미지 데이터만을 학습한 것이 아닌 대량의 ai, real image에 대해 학습한 사전 학습 모델을 파인 튜닝해 성능을 향상시키 고자 하였다. swin-transformer 아키텍처 기반의 ai, real image 분류 모델인 **umm-maybe/AI-image-detector** (<https://huggingface.co/umm-maybe/AI-image-detector>)를 이용해 이전과 같은 방식으로 fine tuning을 진행하였다.

결과는 아래와 같다. 학습시키지 않은 테스트 데이터셋에 대해서도 비교적 높은 정확도(f1 score: 0.94)를 보이는 것을 확인할 수 있다.

```
Collected test dataset 테스트

[18]: pred_results_test = predict_test_data(dataset_test, model_path, "collected_data")
      pred_results_test

/transformers/src/transformers/models/vit/feature_extraction_vit.py:28: FutureWarning: The class ViTFeatureExtractor is deprecated and will be removed
n version 5 of Transformers. Please use ViTImageProcessor instead.
      warnings.warn(
100% ██████████ 3200/3200 [15:10<00:00, 5.41it/s]

[18]:
```

	score	label	filename	answer
0	0.985869	0	./data/test_collected/AI/gwcdrrhwnih.png	0
1	0.999976	0	./data/test_collected/AI/duqanbbgagl.png	0
2	0.895173	0	./data/test_collected/AI/jthibeevsib.png	0
3	0.999964	0	./data/test_collected/AI/9779a187933cea3af42ab...	0
4	0.999548	0	./data/test_collected/AI/ceqdzlzkvhr.png	0
...
3195	0.812649	1	./data/test_collected/Real/47c27103c59fd1c30b0...	1
3196	0.999528	1	./data/test_collected/Real/1f62969ca631996427b...	1
3197	0.999973	1	./data/test_collected/Real/96103a5b122e70ce508...	1
3198	0.999944	1	./data/test_collected/Real/99ddb1e41f32aaf492...	1
3199	0.999785	1	./data/test_collected/Real/05f12d6a22541f0b6ea...	1

```
3200 rows × 4 columns

[19]: results = f1_metric.compute(references=pred_results_test["answer"], predictions=pred_results_test["label"])
      results

[19]: {'f1': 0.9419317187044063}
```

[그림 9] AI-image-detector-finetuned 모델로 collected test 데이터를 분류한 결과

또한, 이 모델(**AI-image-detector-finetuned**)의 DFC_2022 데이터 분류 결과의 f1 score는 **0.9935418938186699** 로 이전 모델에 비해 상당히 높은 점수를 보이는 것을 확인할 수 있었다.

```

2022 DFC 분류 결과

[21]: pred_results_2022 = predict_test_data(dataset_2022, model_path, "2022_DFC")
      pred_results_2022

/transformers/src/transformers/models/vit/feature_extraction_vit.py:28: FutureWarning: The class ViTFeatureExtractor is deprecated and will be removed in version 5 of Transformers. Please use ViTImageProcessor instead.
  warnings.warn(
0%|          | 0/6700 [00:00<?, ?it/s]

[21]:
   score  label  filename  answer
0  0.999529    0  ./data/test_2022/Al/b20cf7b22fda326daf9be6cb3e...    0
1  0.999994    0  ./data/test_2022/Al/2c6c97c46022fa444efd86ab35...    0
2  0.999991    0  ./data/test_2022/Al/dbd037a0a9b7f7f41cb942a1c5...    0
3  0.999995    0  ./data/test_2022/Al/dbafcb892428e73549319216707...    0
4  0.999845    0  ./data/test_2022/Al/f89e6a6a2b9b32e47d051e4cbe...    0
...      ...      ...      ...
6695  0.998433    1  ./data/test_2022/Real/eb0abdacd3a3218d74285a18...    1
6696  0.999310    1  ./data/test_2022/Real/f56caa3b6dfc10ebd216d670...    1
6697  0.999969    1  ./data/test_2022/Real/7735a40e6e14fcb9142d64d0...    1
6698  0.999988    1  ./data/test_2022/Real/82567bc242531a2f3508d4de...    1
6699  0.999998    1  ./data/test_2022/Real/f120f36da372a976a0e7eb851...    1

6700 rows x 4 columns

[24]: results = f1_metric.compute(references=pred_results_2022["answer"], predictions=pred_results_2022["label"])
      results

[24]: {'f1': 0.990833403414347}

```

[그림 10] AI-image-detector-finetuned 모델로 2022_DFC 데이터를 분류한 결과

따라서 **AI-image-detector-finetuned** 모델을 최종 모델로 선정하게 되었다.

DFC_2022 이미지 전체 분류 결과를 csv 파일로 저장하였으며 다음은 생성된 csv 파일의 컬럼명과 설명이다.

Column	Summary
score	모델이 예측한 라벨에 대한 점수
label	이미지에 대한 모델의 예측 결과 (0: AI, 1: Real)
filename	이미지 경로

Q2. Use the method devised in the previous question to determine whether each image has been generated by AI or not. (200 points)

최종 모델(**AI-image-detector-finetuned**)을 이용하여 DFC_2023 데이터를 분류하였고 그 결과를 csv 파일로 저장하였다.

다음은 생성된 csv 파일의 컬럼명과 설명이다.

Column	Summary
score	모델이 예측한 라벨에 대한 점수
label	이미지에 대한 모델의 예측 결과 (0: AI, 1: Real)
filename	이미지 경로

2023_DFC 분류 결과

```
[31]: pred_results_2023 = predict_2023_data(dataset_2023, model_path, "2023_DFC")
      pred_results_2023
```

```
/transformers/src/transformers/models/vit/feature_extraction_vit.py:28: FutureWarning: The class ViTFeatureExtractor is deprecated and will be removed in version 5 of Transformers. Please use ViTImageProcessor instead.
warnings.warn(
  0%|          | 0/3300 [00:00<?, ?it/s]
```

```
[31]:
```

	score	label	filename
0	0.999190	0	./DFC_2023/970bc25332a406de7e3ef64c924a8821.png
1	0.999997	0	./DFC_2023/943623b4d1069b6882b642a14e9b6696.png
2	0.999940	1	./DFC_2023/db90581abec8754485853d5fa44d8d79.png
3	0.999691	1	./DFC_2023/ef5c1f6d2f6678b4d8b8bd53c497f496.png
4	0.999997	0	./DFC_2023/000ce383be6c9e1cefa95afce4ec5ea.png
...
3295	0.999983	0	./DFC_2023/129af062eb64a9d96bab8325bdf006.png
3296	0.999503	0	./DFC_2023/f411c3a02c2452188aa883ba23346b75.png
3297	0.999483	1	./DFC_2023/ddb7442ed2d5941698fec25a670bea05.png
3298	0.999854	1	./DFC_2023/f878d64baeaffd0266cc8b276105f6ef.png
3299	0.999997	0	./DFC_2023/bacdb88d9d72c150a4d774d4011dae1c.png

3300 rows x 3 columns

[그림 11] AI-image-detector-finetuned 모델로 2023_DFC 데이터를 분류한 결과

다음은 첨부된 파일 목록과 설명이다.

Name	Summary
requirements.txt	코드에서 사용된 패키지 목록 및 버전
create_dataset/create_dataset_folder.py	라벨링 정보를 담은 csv 파일을 이용해 학습과 테스트를 위한 로컬 데이터 폴더를 생성하는 코드
tested_model/image_classification-swin.ipynb	성능 비교를 위해 사용된 swin-small-patch4-

	window7-224-finetuned 모델을 생성하고 테스트하는 코드
tested_model/swin-tiny-patch4-window7-224-finetuned	tested_model/image_classification-swin.ipynb 를 실행하여 학습시킨 모델의 상태 데이터와 가중치, 설정 파일 등이 담긴 폴더 (비교 모델)
finetune_pretrained_model.ipynb	모델을 생성하고 테스트하는 코드. 최종 채택 모델인 AI-image-detector-finetuned 모델을 생성한다.
Image_classification.ipynb	AI-image-detector-finetuned 모델을 이용해 2022_DFC, 2023_DFC 데이터 셋을 분류하고 분류 결과를 csv 파일로 만드는 코드
classification_result/ 2022_DFC_dataset_predict_result.csv	AI-image-detector-finetuned 모델로 2022_DFC 데이터셋을 분류한 결과
classification_result/ 2023_DFC_dataset_predict_result.csv	AI-image-detector-finetuned 모델로 2023_DFC 데이터셋을 분류한 결과
AI-image-detector-finetuned/	finetune_pretrained_model.ipynb 를 실행하여 학습시킨 모델의 상태 데이터와 가중치, 설정 파일 등이 담긴 폴더 (최종 채택 모델)



이미지 분류 코드 실행 방법

1. 실행 환경 구성

requirements.txt 를 이용해 패키지를 설치한다. 이후, 데이터 셋을 분류하는 코드인 image_classification.ipynb 와 같은 경로에 2022_DFC, 2023_DFC 이미지 폴더가 위치하도록 한다.

2. 데이터 분류

프로젝트 루트 경로에서 `jupyter notebook` 명령어로 주피터 노트북을 실행한다. 이후 image_classification.ipynb 을 실행하면 이미지가 분류되며 classification_result 폴더에 분류 결과가 csv 파일의 형태로 출력된다.

바탕 화면 > Secu_win > 0. ctf > DFC > 305 – Unveiling the Illusion > 305_소스코드 > classification_result				
이름	수정된 날짜	유형	크기	
 2022_DFC_dataset_predict_result.csv	2023-08-27 오후 9:29	Microsoft Excel 실효...	483KB	
 2023_DFC_dataset_predict_result.csv	2023-08-27 오후 9:43	Microsoft Excel 실효...	237KB	

[그림 12] 2022_DFC, 2023_DFC 분류 결과 csv 파일