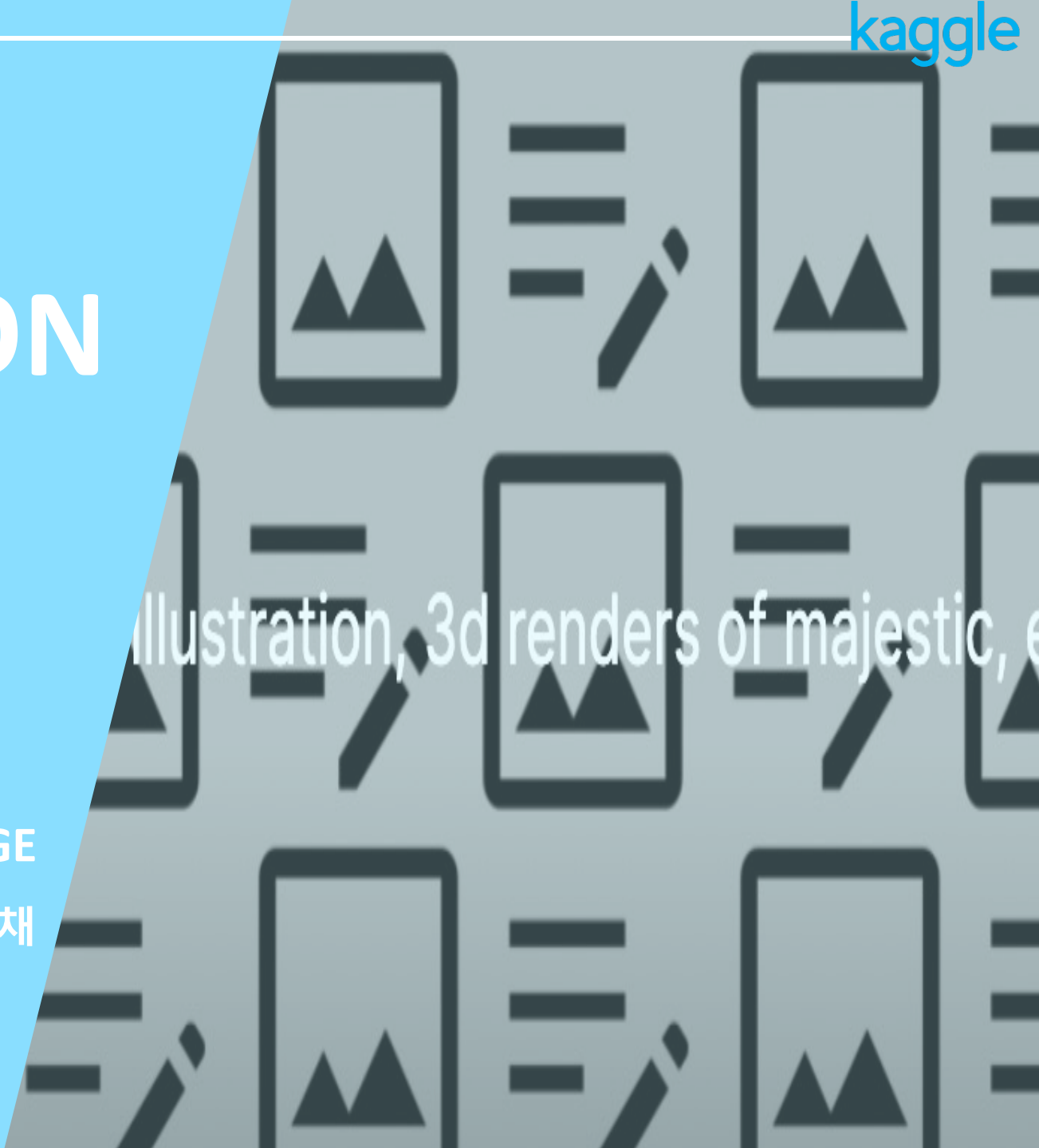
kaggle

STABLE DIFFUSION

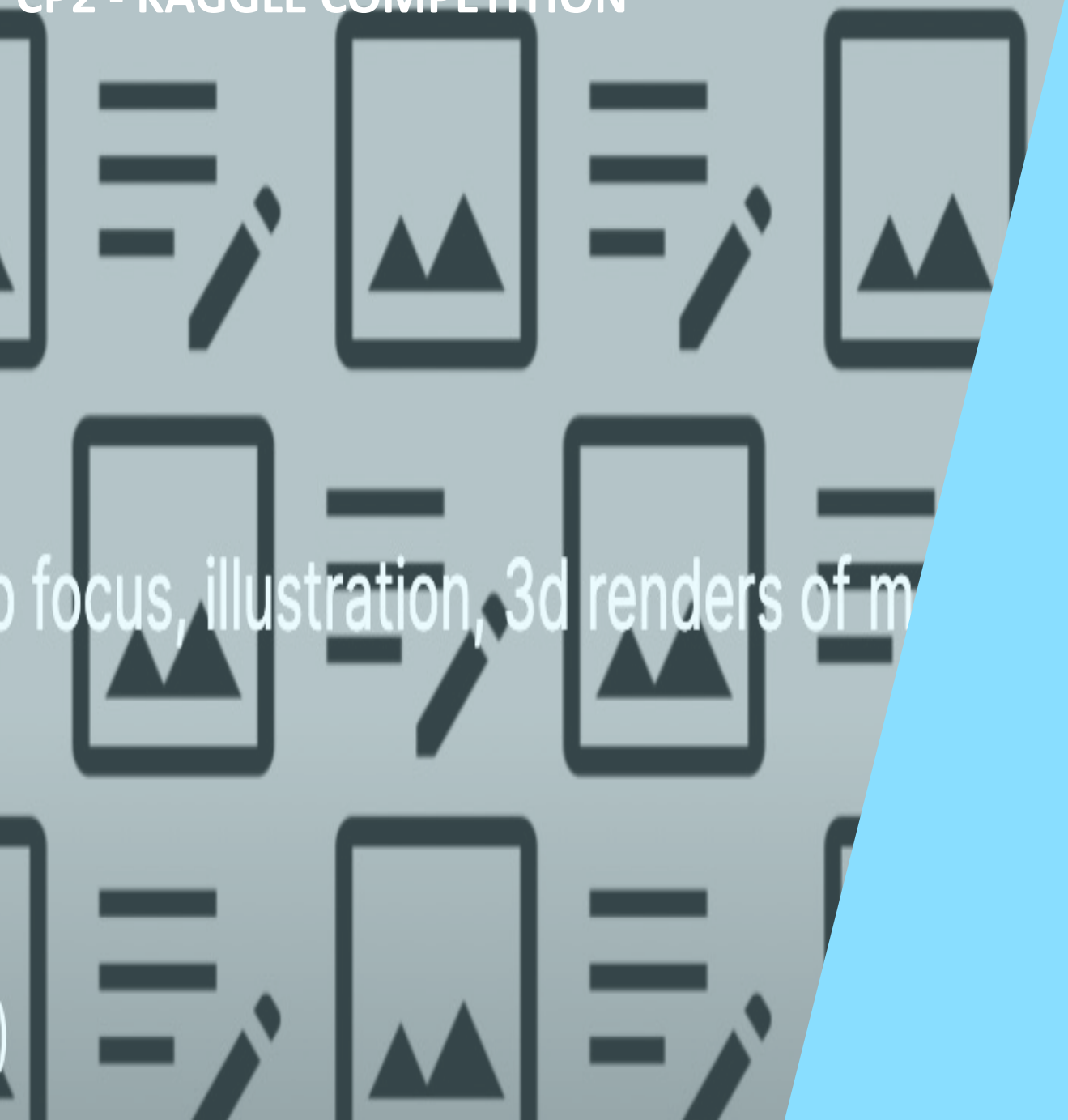


16기 COMPETITION-CHALLENGE

김슬기 이연주 유영채

고용노동부
KOREATECH 직업능력심사평가원

kaggle



CONTENTS

1. 프로젝트 개요

"포트폴리오로 쓸 수 있는 프로젝트를 진행한다."

경진대회에 참가하여 다양한 실험을 하고 기록

대회를 통해서 얻을 수 있는 것:

- 협업 모델링
- 다양한 참여자들의 CODE/DISCUSSION 을 공부하면서 인사이트를 강화
- 각자의 실력 수준 파악
- 주최측에서 원하는 목표/문제에 대해 해결하는 역량을 갖춤
- CV,NLP,ML 등의 다양한 기술 경험
- 입상 및 상위권 진입 시 취업 시장에서 매우 큰 어필 포인트

1. 프로젝트 개요

KAGGLE COMPTETITION 인 STABLE DIFFUSION 참가

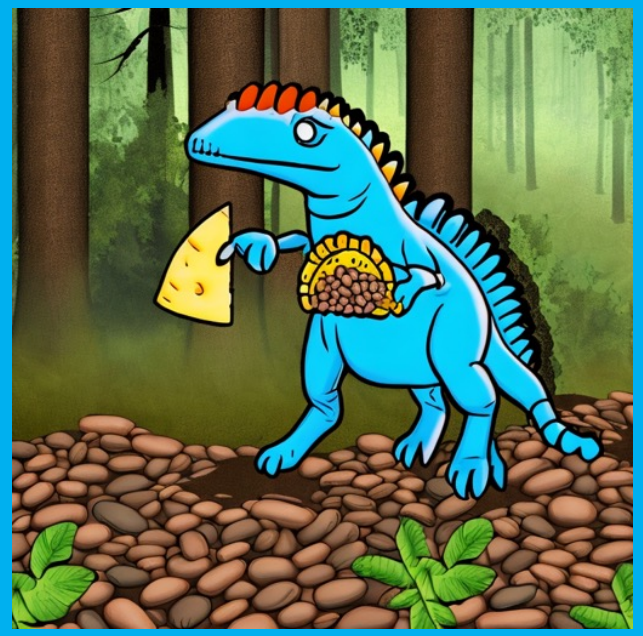
IMAGE TO PROMPTS

굉장히 디테일하고, 선명한 초점의 3D 렌더링 이미지의 프롬프트를 추론하는 것

텍스트-이미지 모델의 일반적인 방향을 뒤집어 , 생성된 이미지가 주어졌을 때 해당하는 텍스트 프롬프트를 예측할 수 있는 모델을 만드는 것

Stable Diffusion 2.0에서 생성된 다양한 프롬프트, 이미지 쌍 데이터 셋에 대해 반대의 상황에는 어떻게 그 요인이 관련성을 가지고 있는지 파악하는 것이 목표

!최종 200등 내 진입을 목표로!



ultrasaurus holding a black bean taco in the woods, near an identical cheneosaurus

DATA SET

IMAGE: Prompt로부터 생성된 이미지로 768x768 사이즈로 50번에 걸쳐 만들어 대회 데이터 셋은 512x512 사이즈로 다운사이징 RE-RUN 테스트 폴더에는 16,000개

PROMPTS: 이미지 셋의 샘플로 사용되어진 예시

SAMPLE_SUBMISSION_CSV: 제출 시 파일, prompt.csv 파일 내의 임베딩 값을 의미하고 이 파이프라인을 검증하는데 사용

2. 프로젝트 구성

김슬기

- BASELINE 모델 검토, 성능개선, 최종 선정, 모델링,
- STREAMLIT 웹 최종 마무리
- 최종 발표
- 주간업무보고, SUBMISSION

이연주

- BASELINE 모델 검토, 성능개선, 최종 선정, 모델링,
- STREAMLIT 웹 최종 마무리,
- 최종 발표
- 주간업무보고, SUBMISSION

유영채

- BASELINE 모델 검토, 성능개선, 최종 선정, 이미지 셋 생성,
- STREAMLIT 웹 BASE 작성, 최종보고서 작성
- 주간업무보고, SUBMISSION

3. 프로젝트 수행 절차 및 방법

1주차	2주차	3주차	4주차
• 데이터셋 EDA • BASELINE 모델 성능 및 구조 확인 • 개선 모델 라인 리스트업, 오픈소스 코드 임포트	• 각 모델별 실험 진행 • 웹/앱 아이디어 구현 • 모델 구조 성능 평가 및 개선	• 최종모델 선정 • 모델 구조 및 초매개변수 조정 • 이미지셋 생성	• 최종 모델링 마무리 • Streamlit 을 이용한 웹 개발

3. 프로젝트 수행 절차 및 방법

	월	화	수	목	금
1주차	27	28	29	30	31
	SUBMISSION				
	팀빌딩/기획안	데이터셋 EDA		개선 모델라인 리스트업	
	선행연구	BASELINE 모델 성능 및 구조 확인		오픈소스 코드 임포트	
2주차	3	4	5	6	7
	모델 성능, 구조 확인 및 개선/ 모델 별 실험 진행				모델 선정
			웹 아이디어 구현		
3주차	10	11	12	13	14
	모델 구조,초매개변수 조정 및 이미지셋 생성			STREAMLIT 웹 개발	
				앞선 모델 합치기	
4주차	17	18	19		
	STREAMLIT 웹 구현				
	최종모델링 마무리				
	최종 보고서 작성 및 영상 촬영				

3. 프로젝트 수행 절차 및 방법

2주차 회의 결과

KAGGLE DISCUSSION 의 **HOW TO GET TO 0.58 AND BEYOND, WITH CODE AND DATA** 의 단계로 모델을 개발하기로 최종 결정

1단계 : SD2GPT2 이용해서 dataset(프롬프트, 이미지) 생성 후 문장 임베딩 유사성 기반으로 0.95 이상의 데이터 셋은 필터링

2단계 : 80K(테스트)이미지 셋으로 clip interrogator + vit 을 이용하여 로드 프롬프트 문장 임베딩 간 유사성을 예측하는 모델을 훈련

3단계 : 2단계에서 훈련한 모델(유사성 예측 모델)을 사용하여 샘플(이미지)을 생성

4. 프로젝트 수행 결과

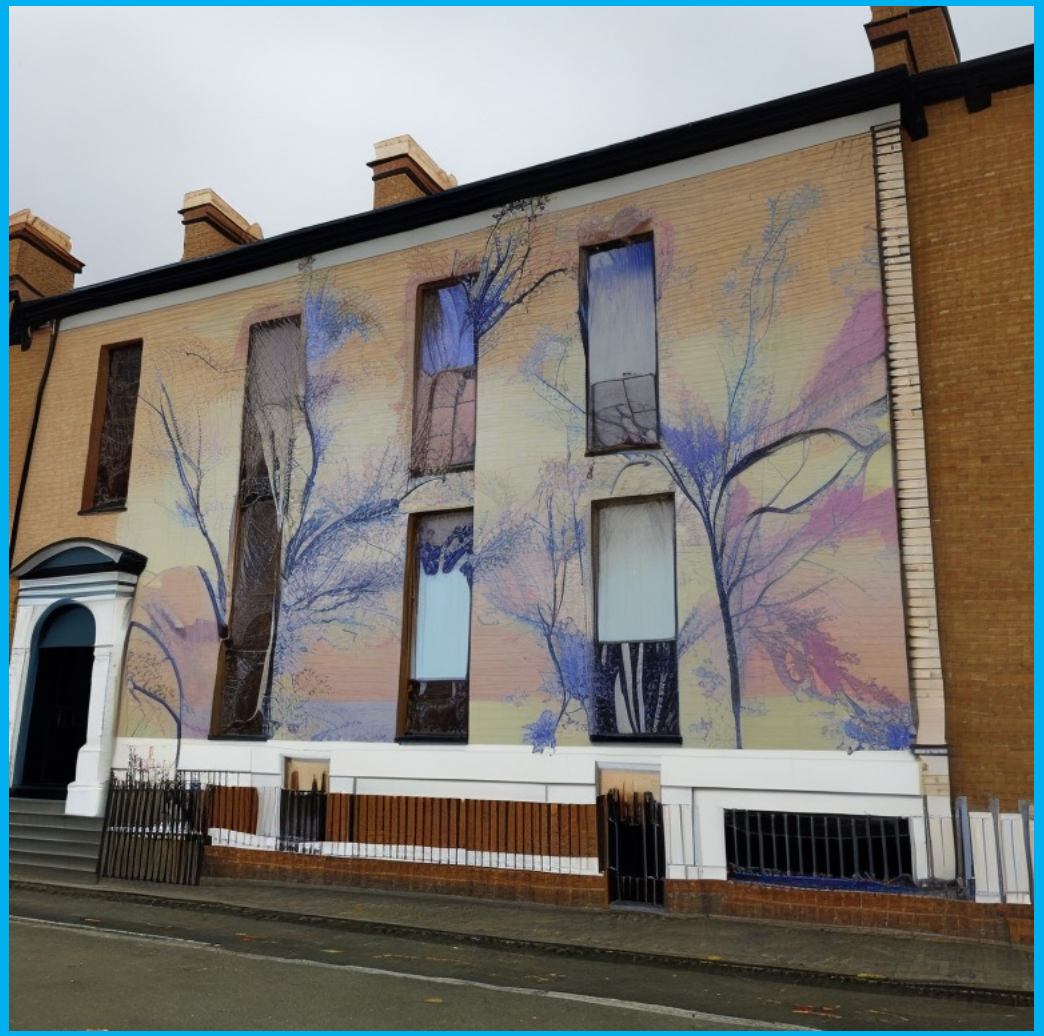
1. 이미지 및 프롬프트 생성

SD2+GPT2 로 이미지 및 프롬프트 1000 개 생성 후 EMBEDDING 기반의 FILTERING 코드를 구현하였으나 유사도 0.95 이상의 데이터 셋 ❌

이미지셋은 35번에 걸쳐 712x712 사이즈로 생성

PROMPTS 최소 10글자 이상으로 생성하였으며, 후에 특수문자를 제거

```
['A picture of her at a concert in 2009 The photo of her at a concert in 2009 right This is my life n
ow and I could tell she had her own life and I can t blame her I can t blame',
 'A picture of Trump s face before you re invited to the rally This has been an unusually quiet month
for Donald Trump From early September he attended rallies outside Alabama Florida Iowa North Carolina
as well as Louisiana Nevada and',
 'A picture of the three men who are accused of carrying out the attempted attack on the school A pic
ture of the three who were accused of carrying out the attempted attack on the school Photo AFP A pic
ture of the three who',
 'A picture of a woman sitting in a chair next to his wife The picture was taken on December 20 2012
Photo Michael Videl AP In the next decade California s state law requires a family member to attend',
 'A picture of the room below is available today It looks exactly like the one you see below It s a l
ot more stylish for the price than you might assume but not bad for an iPhone 7 Plus',
 'A picture of the new look new look shooter is now all over Twitter Logan a 36 year old from Manches
ter has also tweeted about Trump over the weekend in which some people have complained about the ton
e',
```



A building with a mural on the side

4. 프로젝트 수행 결과

CLIP Interrogator

- 최종모델 : CLIP INTERRORGATOR +VIT

```
class CFG:
    device = "cuda"
    seed = 42
    embedding_length = 384
    sentence_model_path = "/kaggle/input/sentence-transformers-222/all-MiniLM-L6-v
2"
    blip_model_path = "/kaggle/input/clip-interrogator-models-x/model_large_captio
n.pth"
    ci_clip_model_name = "ViT-H-14/laion2b_s32b_b79k"
    clip_model_name = "ViT-H-14"
    clip_model_path = "/kaggle/input/clip-interrogator-models-x/CLIP-ViT-H-14-laio
n2B-s32B-b79K/open_clip_pytorch_model.bin"
    cache_path = "/kaggle/input/clip-interrogator-models-x"
```

- 클래스 CFG 정의
  : 다양한 구성 요소 정의

```
sys.path.append('../input/sentence-transformers-222/sentence-transformers')
from sentence_transformers import SentenceTransformer, models

st_model = SentenceTransformer(CFG.sentence_model_path)
```

- SentenceTransformer 모델 불러오기
- 텍스트를 입력으로 받아,
  입력 텍스트를 벡터 임베딩으로 변환

4. 프로젝트 수행 결과

CLIP Interrogator

- 최종모델 : CLIP INTERRORGATOR +VIT

```
# Step 1: Config 객체 생성
model_config = clip_interrogator.Config(clip_model_name=CFG.ci_clip_model_name)
model_config.cache_path = CFG.cache_path
```

```
configs_path = os.path.join(os.path.dirname(os.path.dirname(blip_path)), 'configs')
med_config = os.path.join(configs_path, 'med_config.json')
blip_model = blip.blip_decoder(
    pretrained=CFG.blip_model_path,
    image_size=model_config.blip_image_eval_size,
    vit=model_config.blip_model_type,
    med_config=med_config
)
blip_model.eval()
blip_model = blip_model.to(model_config.device)
model_config.blip_model = blip_model
```

- clip_interrogator를 사용하여 모델 구성(config)

4. 프로젝트 수행 결과

CLIP Interrogator

- 최종모델 : CLIP INTERRORGATOR +VIT

```
# CLIP 모델 생성
clip_model = open_clip.create_model(CFG.clip_model_name, precision='fp16' if model_config.device == 'cuda' e
lse 'fp32')

# #학습된 모델 불러오기
open_clip.load_checkpoint(clip_model, CFG.clip_model_path)
clip_model.to(model_config.device).eval()
model_config.clip_model = clip_model
```

```
clip_preprocess = transforms.Compose([
     transforms.Resize((224, 224)),
     transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(
         mean=[0.5, 0.5, 0.5],
         std=[0.5, 0.5, 0.5]
        #transforms.Normalize((0.48145466, 0.4578275, 0.40821073), (0.26862954, 0.26130258, 0.27577711))
     )
 ])
model_config.clip_preprocess = clip_preprocess
```

- CLIP 모델 불러와 eval 모드로 설정
- 디바이스(device)에 맞게 설정
- CLIP 모델 전처리(preprocess) 설정

4. 프로젝트 수행 결과

CLIP Interrogator

- 최종모델 : CLIP INTERRORGATOR +VIT

```
# Step 2: Interrogator 객체 생성
ci = clip_interrogator.Interrogator(model_config)
```

```
Loaded CLIP model and data in 2.10 seconds.
```

- interrogator 객체를 생성

```
def image_to_features(image: Image) -> torch.Tensor:
        ci._prepare_clip()
        images = ci.clip_preprocess(image).unsqueeze(0).to(ci.device)
        with torch.no_grad(), torch.cuda.amp.autocast():
            image_features = ci.clip_model.encode_image(images)
            image_features /= image_features.norm(dim=-1, keepdim=True)
        return image_features
```

CLIP Interrogator 모듈의 image_to_features 함수와 interrogate_classic 함수를 정의

```
def interrogate_classic(image: Image,max_flavors: int=3) -> str:
        """Classic mode creates a prompt in a standard format first describing the image,
        then listing the artist, trending, movement, and flavor text modifiers."""
        caption = ci.generate_caption(image)
        image_features = ci.image_to_features(image)

        medium = ci.mediums.rank(image_features, 1)[0]
        movement = ci.movements.rank(image_features, 1)[0]
        flaves = ", ".join(ci.flavors.rank(image_features, max_flavors))

        if caption.startswith(medium):
            prompt = f"{caption} , {movement}, {flaves}"
        else:
            prompt = f"{caption}, {medium} , {movement}, {flaves}"

        return clip_interrogator._truncate_to_fit(prompt, ci.tokenize)
```

4. 프로젝트 수행 결과

CLIP Interrogator

- 최종모델 : CLIP INTERRORGATOR +VIT

```
# #이미지에 적용
prompts = []

images_path = "../input/stable-diffusion-image-to-prompts/images/"
for image_name in images:
    img = Image.open(images_path + image_name).convert("RGB")

    generated = interrogate_classic(img)

    prompts.append(generated)
```

```
embeddings
```

```
array([-0.00050107,  0.06069836, -0.06835597, ...,  0.06744548,
       -0.0664781 ,  0.04254934], dtype=float32)
```

- 이미지를 모델에 적용하여 프롬프트 생성
- 생성한 프롬프트 임베딩

4. 프로젝트 수행 결과

VIT

- 최종모델 : CLIP INTERRORGATOR +VIT

```python
class DiffusionTestDataset(Dataset):
    def __init__(self, images, transform):
        self.images = images
        self.transform = transform

    def __len__(self):
        return len(self.images)

    def __getitem__(self, idx):
        image = Image.open(self.images[idx])
        image = self.transform(image)
        return image
```

- PyTorch의 Dataset 클래스를 상속하여, 이미지 데이터셋을 구현
- 이미지 분류 모델의 추론(inference)을 수행하는 함수인 predict()를 정의

```python
def predict(
    images,
    model_path,
    model_name,
    input_size,
    batch_size,learning_rate
):
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    transform = transforms.Compose([
        transforms.Resize(input_size),
        transforms.RandomHorizontalFlip(p=0.5),
#         transforms.RandomRotation(degrees=10),

        # transforms.RandomVerticalFlip(p=0.5),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
    ])
    dataset = DiffusionTestDataset(images, transform)
    dataloader = DataLoader(
        dataset=dataset,
        shuffle=False,
        batch_size=batch_size,
        pin_memory=True,
        num_workers=2,
        drop_last=False
    )

    model = timm.create_model(
```

4. 프로젝트 수행 결과

VIT

- 최종모델 : CLIP INTERRORGATOR +VIT

```
data_path = "/kaggle/input/gustavosta-stable-diffusion-prompts-sd2/train.csv"
image_dir = "/kaggle/input/gustavosta-stable-diffusion-prompts-sd2/train_images"
images = list(Path('/kaggle/input/gustavosta-stable-diffusion-prompts-sd2/train_images').glob('*.jpg'))
```

- **80K 이미지셋을 이용하여 학습**
- **Vit 하이퍼파라미터 튜닝 및 best params 추출**
- Input_size = 224
  batch_size = 256
  patch_size = 8
  learning_rate = 0.001

```
import itertools

# 하이퍼파라미터 조합 생성
param_grid = {
    'model_path': ['/kaggle/input/stable-diffusion-vit-baseline-train/vit_base_patch16_224.pth'],
    'model_name': ['vit_base_patch16_224'],
    'input_size': [224],
    'batch_size': [16, 32, 64, 128, 256],
    'patch_size': [8, 16, 32],
    'learning_rate': [1e-5, 1e-4, 1e-3]
}

param_list = list(itertools.product(*param_grid.values()))

# 결과 저장을 위한 딕셔너리 생성
results = {}

# 하이퍼파라미터 조합마다 predict 수행
for params in param_list:
    # 딕셔너리 형태로 변환
    params_dict = {list(param_grid.keys())[i]: params[i] for i in range(len(params))}
    print("Testing params:", params_dict)

    # predict 수행
    embeddings = predict(images, params_dict['model_path'], params_dict['model_name'],
                         params_dict['input_size'], params_dict['batch_size'])

    # 결과 저장
    results[str(params_dict)] = embeddings.mean()

# 결과 출력
print("Results:", results)

# 최적의 조합 찾기
best_params = max(results, key=results.get)
print("Best params:", best_params)
```

4. 프로젝트 수행 결과

VIT

- 최종모델 : CLIP INTERRORGATOR +VIT

```
class CFG:
    model_path = '/kaggle/input/stable-diffusion-vit-baseline-train/vit_base_patch16_224.pth'
    model_name = 'vit_base_patch16_224'
    input_size = 224
    batch_size = 256
    patch_size = 8
    learning_rate = 0.001
```

```
images = list(Path('/kaggle/input/stable-diffusion-image-to-prompts/images').glob('*.png'))
imgIds = [i.stem for i in images]
EMBEDDING_LENGTH = 384
imgId_eId = [
    '_'.join(map(str, i)) for i in zip(
        np.repeat(imgIds, EMBEDDING_LENGTH),
        np.tile(range(EMBEDDING_LENGTH), len(imgIds)))]
embeddings3 = predict(images, CFG.model_path, CFG.model_name, CFG.input_size, CFG.batch_size, CF
```

+ Code + Markdown

```
embeddings3
```

```
array([-0.05055949,  0.03939409, -0.00764806, ...,  0.04537146,
       -0.02931633,  0.03523336], dtype=float32)
```

- best_params를 이용해
- Vit를 이용한 Embedding값 추출

4. 프로젝트 수행 결과

ENSEMBLE THE OUTPUTS

- 최종모델 : CLIP INTERRORGATOR +VIT

```
[ ]: embeddings = embeddings2*ratio_CLIP_Interrogator + ratio_ViT_B_16 * embeddings3
```

```
[ ]: submission = pd.DataFrame(
         index=imgId_eId,
         data=embeddings,
         columns=['val']
     ).rename_axis('imgId_eId')
```

```
[ ]: submission.head()
```

```
[ ]:
```

imgId_eId	val
f27825b2c_0	-0.030706
f27825b2c_1	0.045031
f27825b2c_2	-0.031132
f27825b2c_3	-0.045387
f27825b2c_4	-0.060089

- Clip Interrogator 에서 추출한 Embedding값 과 Vit_B_16 에서 추출한 Embedding값을 결합하여 최종 임베딩값 확인

- ratio_VIT_B_16 = 0.55
  ratio_CLIP_Interrogator = 0.45 로 진행하였으며 베이스라인과 다른 코드를 참고하여 비율 확정

4. 프로젝트 수행 결과

최종 캐글 SCORE

316	**CP2_stable_diffusion**		0.53907	47	13h

Your Best Entry!

Your submission scored , which is not an improvement of your previous score. Keep trying!

4. 프로젝트 수행 결과

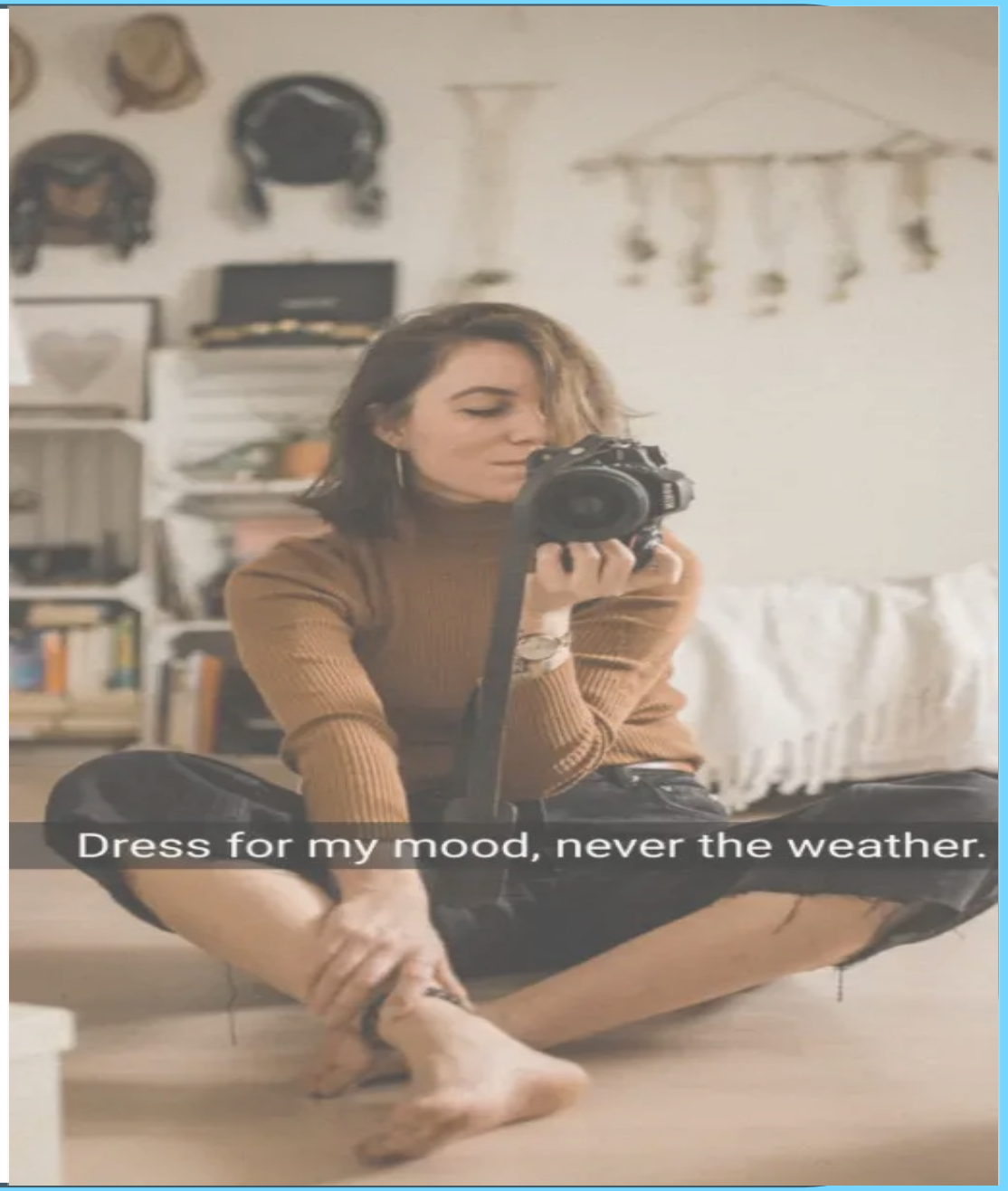
2주차 주간 회의 결과

또한 캐글 대회의 점수 올리기에는 한계 ⭕

IMG2TXT 사업성을 고려하여 웹/앱에 도전하기로 결정

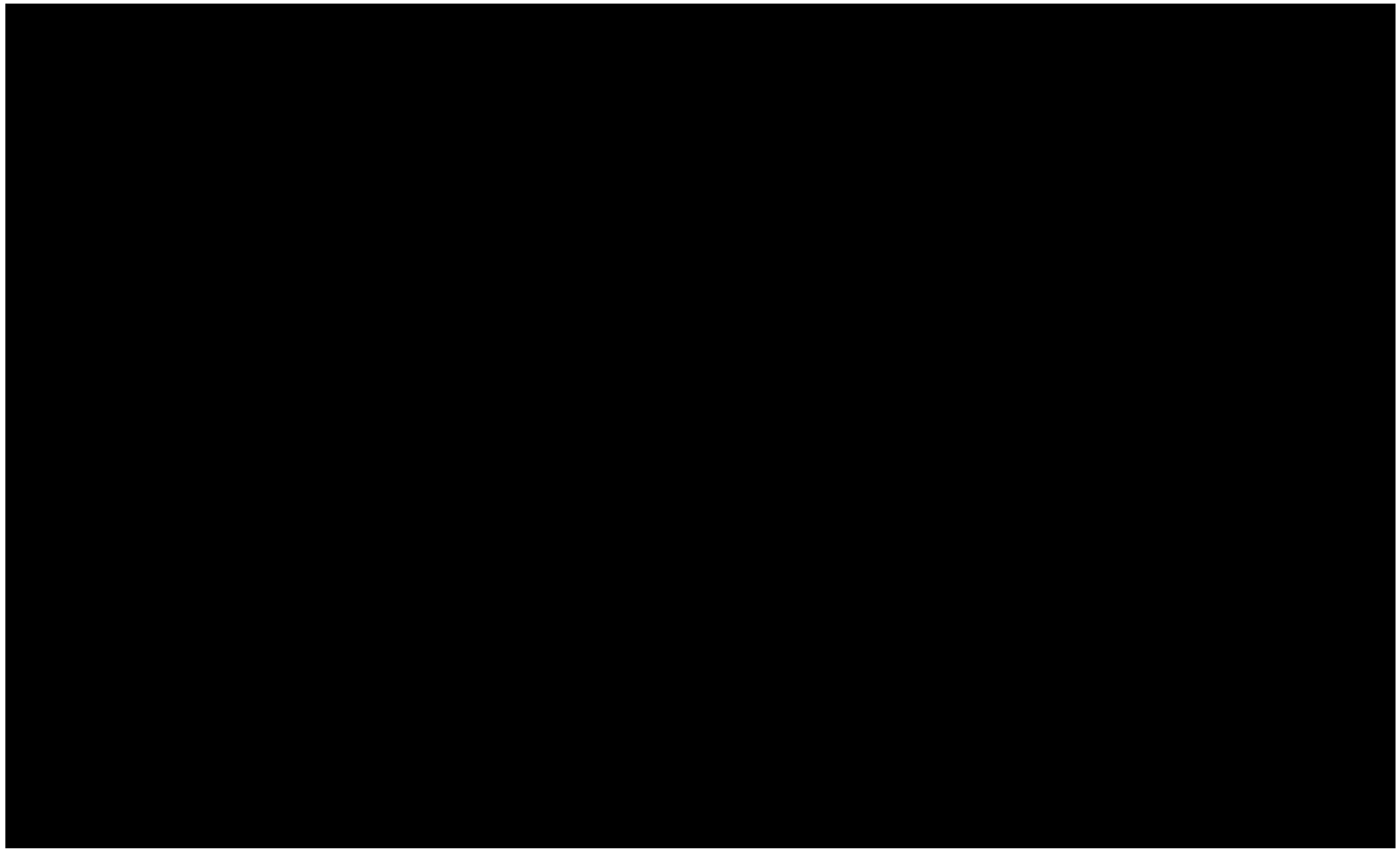
인스타그램이나 스냅챗 등의 소셜미디어에 사진을 업로드 할 때 많은 유저들이 캡션에 대해서 고민

오른쪽과 같이 이미지에 맞는 캡션을 제공할 수 있는 웹/앱을 제작하여 배포하는 것이 목표.



4. 프로젝트 수행 결과

STREAMLIT 실행 화면



kaggle

4. 프로젝트 수행 결과

STREAMLIT 코드 구현

STREAMLIT을 통해 웹에 이미지를 업로드 하면 하단에 이미지에 알맞는 CAPTION 을 생성하는 코드 구현

만든 모델을 pickling으로 가져오려 했으나, 모델을 가져오는 것에 실패하여 원래 의도한 내용이 구현되는 것에 그침

```
# 왼쪽에 위치시킬 이미지
image = Image.open("insta_2.png")
st.sidebar.image(image, width=250)
st.sidebar.title('Captioning for Instagram')

image = Image.open("insta.png")
st.image(image, width=100)
st.title('Captioning for Instagram')

st.subheader('Img to Text')
uploaded_file = st.file_uploader('', type=['png', 'jpg', 'jpeg','webp'])

if uploaded_file is not None:
    # 이미지파일 열기
    image = Image.open(uploaded_file)

    #이미지파일 업로드
    st.image(image, use_column_width=True, width = 30)

    # model button 생성
    if st.button('Start Captioning'):
        image = transforms.ToTensor()(Image.open(uploaded_file))
        # 저장된 객체 불러오기 및 사용 예시
        processor = ViTImageProcessor.from_pretrained('google/vit-base-patch16-224')
        model = ViTForImageClassification.from_pretrained('google/vit-base-patch16-224')

        inputs = processor(images=image, return_tensors="pt")
        outputs = model(**inputs)
        logits = outputs.logits
        # model predicts one of the 1000 ImageNet classes
        predicted_class_idx = logits.argmax(-1).item()
        st.write("Caption:", model.config.id2label[predicted_class_idx])
```

5. 자체평가의견

초기 기획과 부합 정도

- 협업을 통한 모델링 경험
- 다양한 참여자들의 CODE/DISCUSSION 을 공부하며 인사이트 강화
- 대회를 통한 각자의 수준 파악
- 주최 측에서 원하는 목표/문제에 대해 파악(해결)하는 역량
- CV,NLP,ML 등의 다양한 기술 경험
- submission 문제로 인한 상위권 진입은 이뤄내지 못함

달성도 및 완성도

- 계획 단계에 맞춰 스케줄대로 진행
- CODE/DISCUSSION 을 참조하며 다양한 실험을 시도하였으나 SCORE 향상이나 모델을 이용한 웹 구현에서 원하는 결과값을 이루어내지 못해 아쉬움

실무 활용 가능 정도

- 전체적인 단계를 진행해보며 흐름 파악
[데이터셋팅 → 모델학습 → 서비스 구현] (배포)

5. 자체평가의견

최종 평가 의견 및 느낀 점

김슬기

- 최종모델로 SCORE 및 웹 구현이 이뤄지지 못해 아쉬움
- 하지만 첫 대회 참여와 팀프로젝트 기회에 의의를 두고 이후 지속적인 대회 참여 및 팀프로젝트에 밑거름이 될 것 같음
- 초반 대회에 대한 이해와 사전지식 부족으로 모델자체에 대한 더 깊은 학습이 이뤄지지 못한 것에 부족함을 느낌
- 따라서 CLIP, VIT 를 포함한 CV 역량을 키울 수 있는 스터디가 필요

이연주

- 개인적으로 흥미 있었던 이미지에서 프롬프트를 추출하는 프로젝트를 진행하면서 배운 점이 많음.
- 새로운 모델들을 깊게 접하고 여러 모델을 선정하여 직접 모델을 돌려보며 프로젝트를 진행했다는 점, streamlit 이라는 새로운 기술적인 부분을 스스로 배우며 실행했다는 점에서 성장할 수 있었던 좋은 기회라고 생각
- 하지만 기술적인 부분에서 아쉬운 점을 느꼈다. 최종 모델에 대한 스코어를 내지 못했다는 점
- 웹 구현을 만든 모델이 아닌 다른 모델로 진행하였다는 점에서 부족한 점을 많이 느낌
- 이번 프로젝트를 통해 아쉬웠던 모델에 대한 코드와 구현에 대해 더 깊게 공부하고 싶음

유영채

- 대회 참여 및 팀프로젝트를 통해 자신의 수준을 되돌아 볼 수 있는 좋은 경험
- 사전에 정보를 파악하지 못한 채로 대회 참여를 하고 데이터 분석의 롤을 다 하지 못 한 거 같아 아쉬움
- 하지만 다양한 기술을 경험해 볼 수 있어 좋은 기회 였음
- 추 후 에 는 SQL 로 대 쉬 보 드 를 구현해보고 싶음.

참고자료

캐글 디스커션 링크: https://www.kaggle.com/competitions/stable-diffusion-image-to-prompts/discussion/398529

80K 이미지셋 링크: https://www.kaggle.com/competitions/stable-diffusion-image-to-prompts/discussion/390674

SD2+GPT2링크: https://www.kaggle.com/code/mvvppp/sd2-gpt2-prompt-image-gen-dataset-creation

모델 참고 링크: https://www.kaggle.com/code/jesherjoshua/lb-0-53917-clipinterrogator-vit

kaggle

THANK YOU