
BEAUTY
TryOnDiffusion
Virtual Fitting
MS
AI School
대한상공회의소

CONTENTS

개요

- 온라인 쇼핑몰의 성장으로 상품을 실착해보지 못하고 단지 모델의 피팅, 상품의 상세 설명으로만 예상을 해서 구매해야 하는 경우가 증가하고 있음.

- 이로 인해 상품의 환불이 증가하고, 구매자의 소비 만족도가 떨어질 수 있음

- AI-Hub의 패션상품 및 착용영상 Dataset(https://aihub.or.kr/aihubdata/data/view.do?currMenu=&topMenu=&aihubDataSe=realm&dataSetSn=78)을 이용하여 Model Image, Garment Image를 학습

- TryOn 모델 중 가장 최신 모델인 TryOnDiffusion Model을 Reference하여 Model 구축

  (2023. 6. 14)

개요

자료 출처 : https://tryondiffusion.github.io/

팀 구성 및 역할

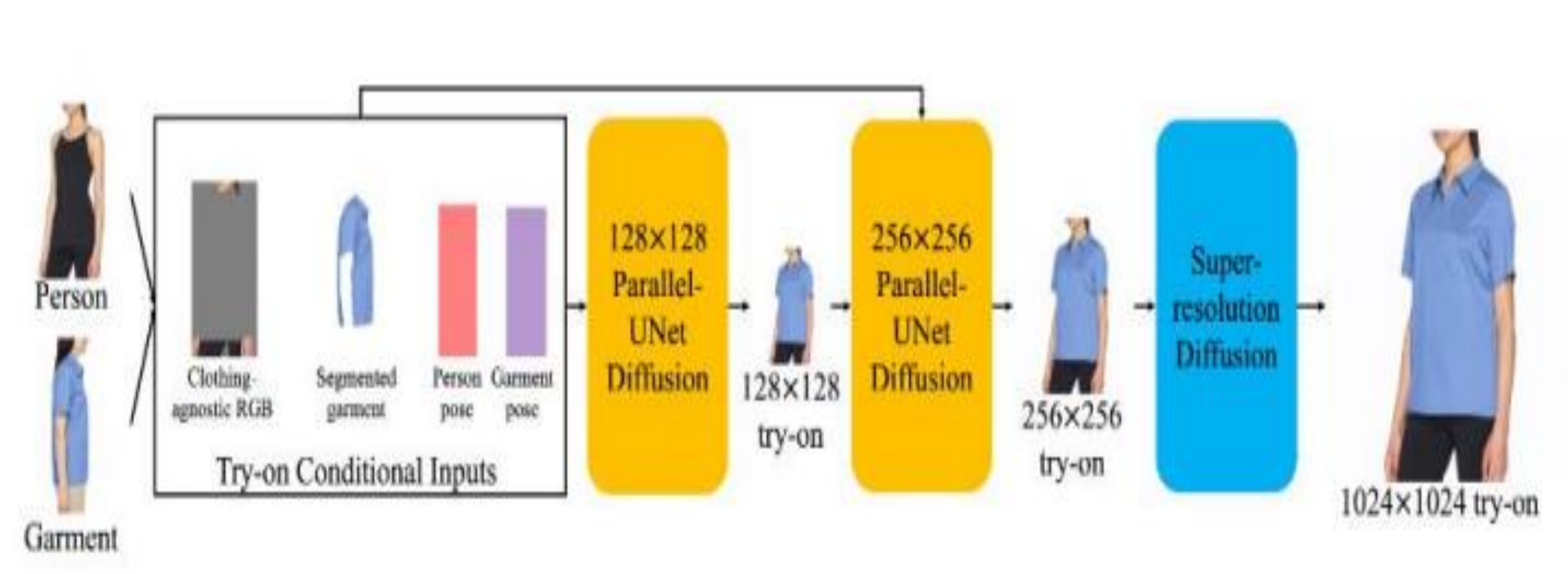
팀원명	역할	담당 업무
배윤주	팀원	Image Preprocessing
전진희	팀원	Image Preprocessing
최준혁	팀원	ParallelUNet, Diffusion model architecture building
이지호	팀장	EfficientUNet model architecture building
최보미	팀원	EfficientUNet model architecture building

프로젝트 수행 절차 및 방법

구분	기간	활동	비고
사전 기획	➢ 8/14(월) ~ 8/18(금)	➢ 프로젝트 기획 및 주제 선정 ➢ 기획안 작성	➢ 아이디어 선정
데이터 수집	➢ 8/19(토) ~ 8/20(일)	➢ 필요 데이터 및 수집 절차 정의 ➢ 외부 데이터 수집	➢ AI-Hub 데이터 수집
데이터 전처리	➢ 8/21(월) ~ 8/25(금)	➢ 이미지 전처리 및 세그멘테이션	
모델링	➢ 8/26(토) ~ 9/10(일)	➢ 모델 구현	
서비스 구축	➢ 9/11(월) ~ 9/15(금)	➢ 웹 서비스 시스템 설계	➢ 모델 구현하지 못하여 서비스 구축하지 못함
총 개발기간	➢ 8/14(월) ~ 9/15(금)(총 5주)		

모델 소개

Overall Pipeline



자료 출처 : https://tryondiffusion.github.io/

발표 순서

- Image Preprocessing

  배윤주 전진희

- ParallelUNet, Diffusion Model architecture building

  최준혁

- EfficientUNet Model architecture building

  최보미

- Summary and Future Work

  이지호

프로젝트 수행 결과

결과 제시 ① 이미지 전처리

학습 데이터 소개 (Train/dev set)

#패션 # 패션상품 # 패션모델 # 사람자세 # 사람영역

패션상품 및 착용 영상

분야 영상이미지 유형 이미지

갱신년월 : 2023-05 구축년도 : 2020 조회수 : 7,916 다운로드 : 1,064 용량 : 36.70 GB

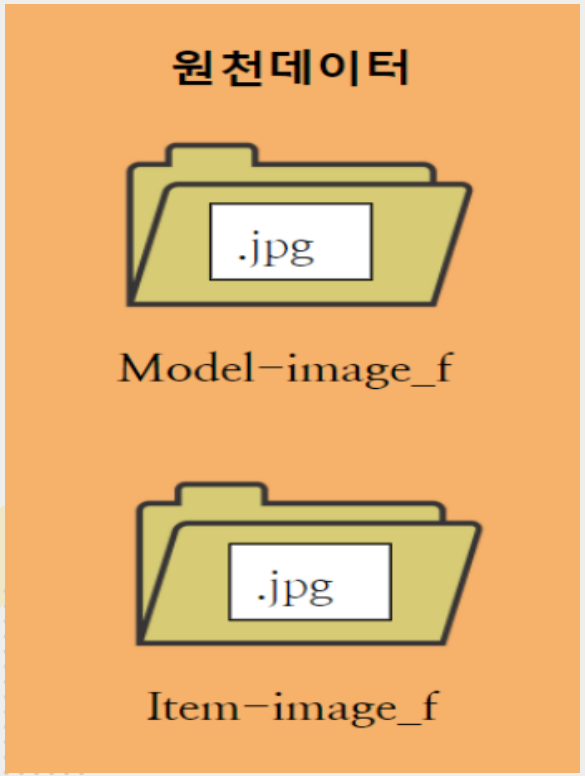
다운로드 샘플 데이터

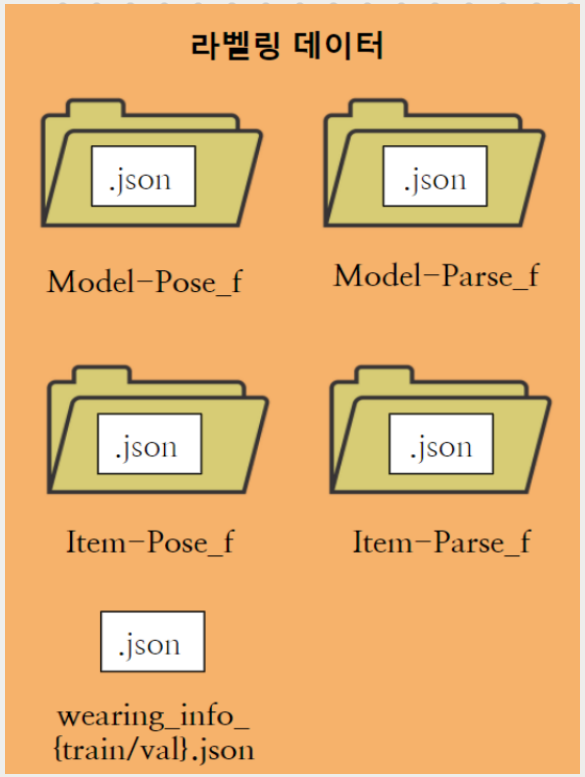
자료 출처 : AI-Hub

프로젝트 수행 결과

결과 제시 ① 이미지 전처리

학습 데이터 소개 (Train/dev set)





데이터 구성

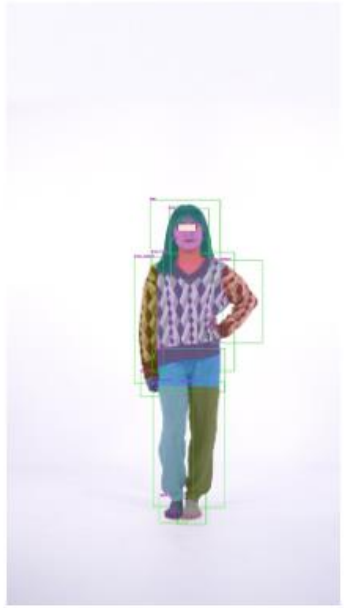
- 사람 패션 이미지: Model-Image
- 사람 패션 이미지 모델 키포인트: Model-Pose
- 사람 패션 영상 모델 영역: Model-Parse
- 패션상품 대표 사진: Item-Image
- 패션상품 키포인트: Item-Pose
- 패션상품 영역: Item-Parse
- 사람 패션 영상 및 패션상품 페어: wearing_info.json

자료 출처 : AI-Hub

프로젝트 수행 결과

결과 제시 ① 이미지 전처리

학습 데이터 소개 (Train/dev set)

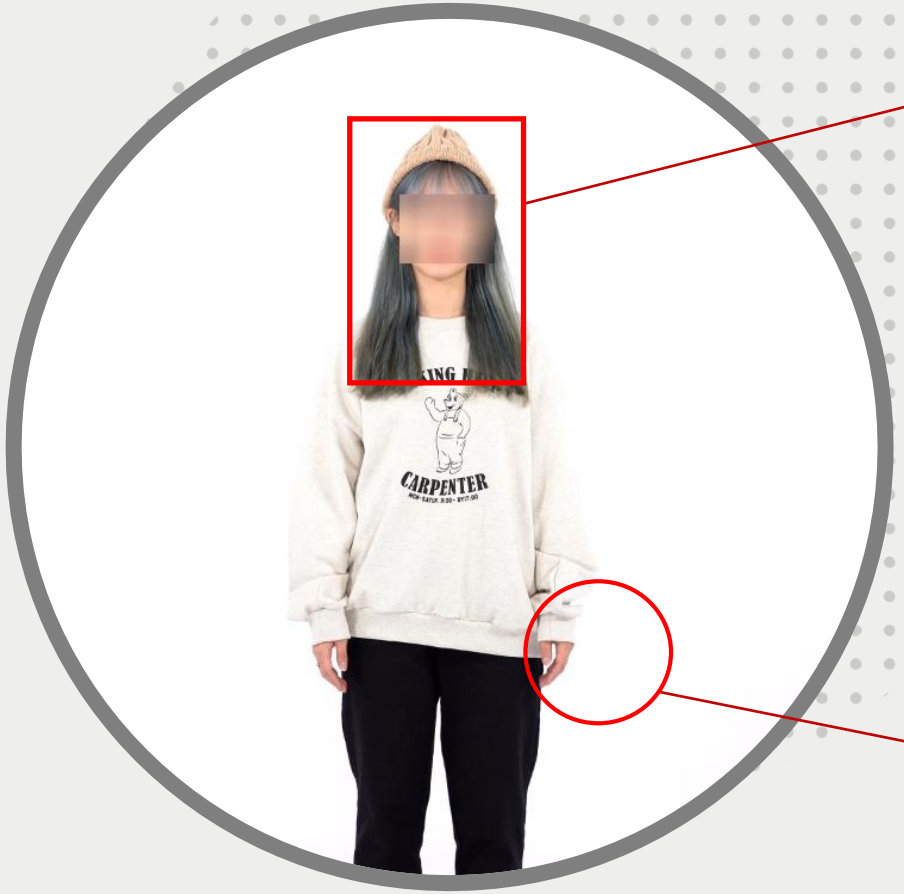
			
모델 키포인트	모델 영역	상품 키포인트	상품 영역

자료 출처 : AI-Hub

프로젝트 수행 결과

결과 제시 ① 이미지 전처리

이미지 전처리 프로세스



Sp(Separated person)

Jp(Joint person)

프로젝트 수행 결과

결과 제시 ① 이미지 전처리

이미지 전처리 프로세스

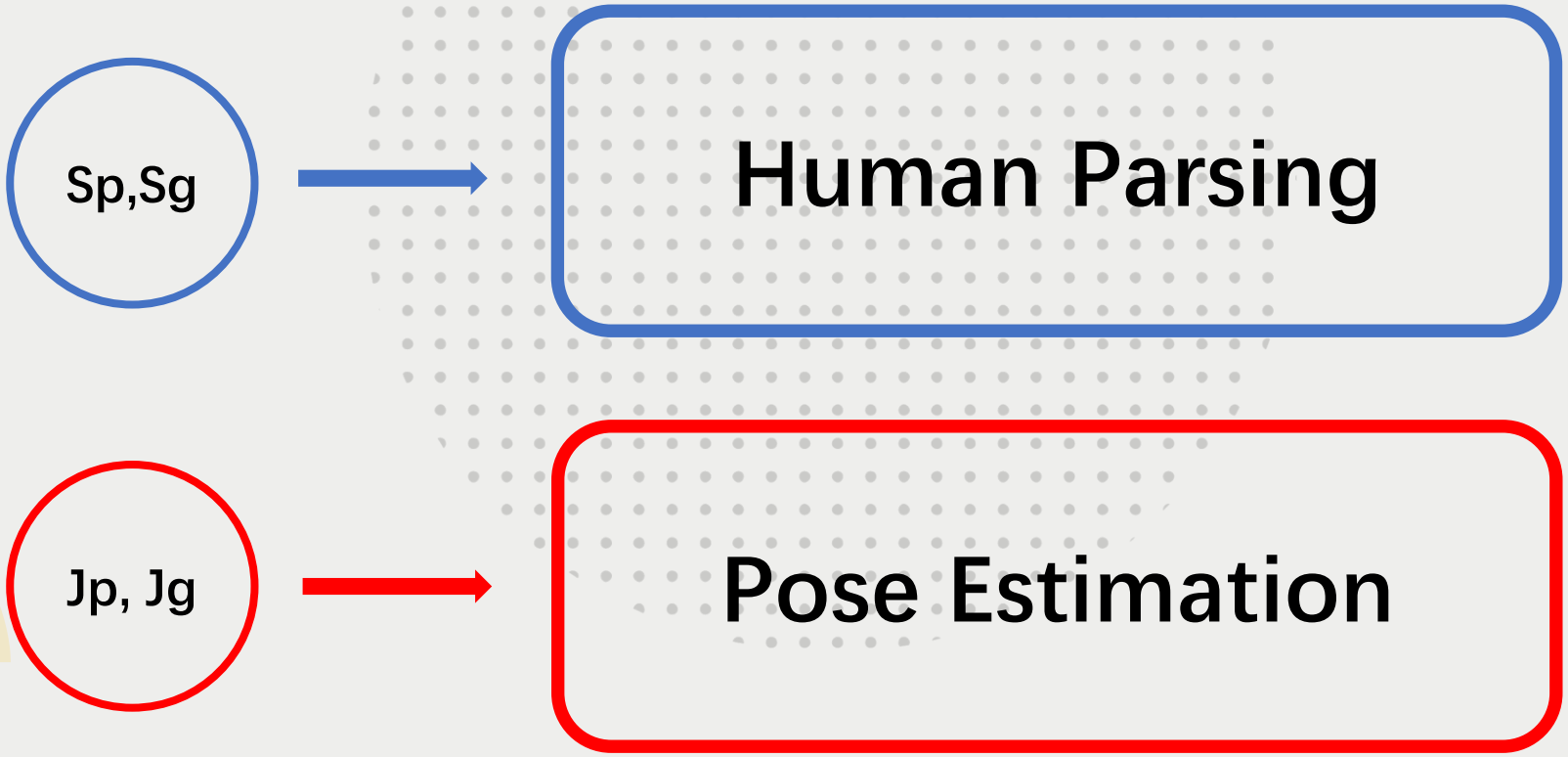
Sg(Separated garment)

Jg(Joint garment)

프로젝트 수행 결과

결과 제시 ① 이미지 전처리

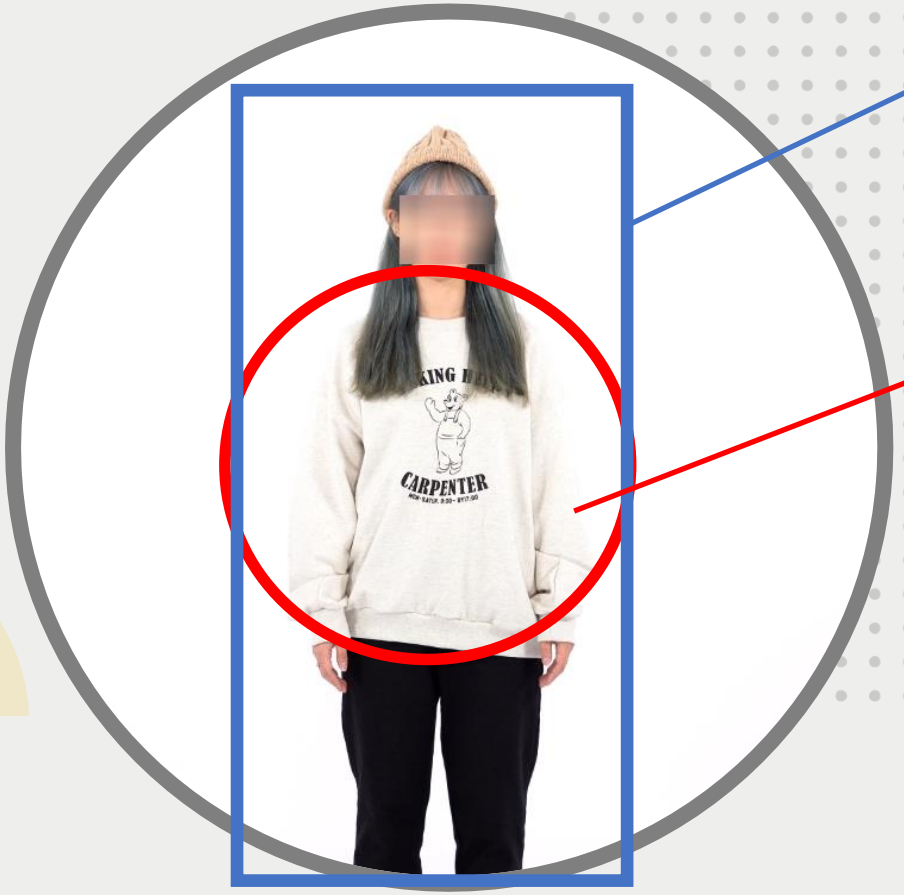
이미지 전처리 프로세스



프로젝트 수행 결과

결과 제시 ① 이미지 전처리

이미지 전처리 프로세스



Ia(Isolated atomy)

Ic(Isolated clothing)

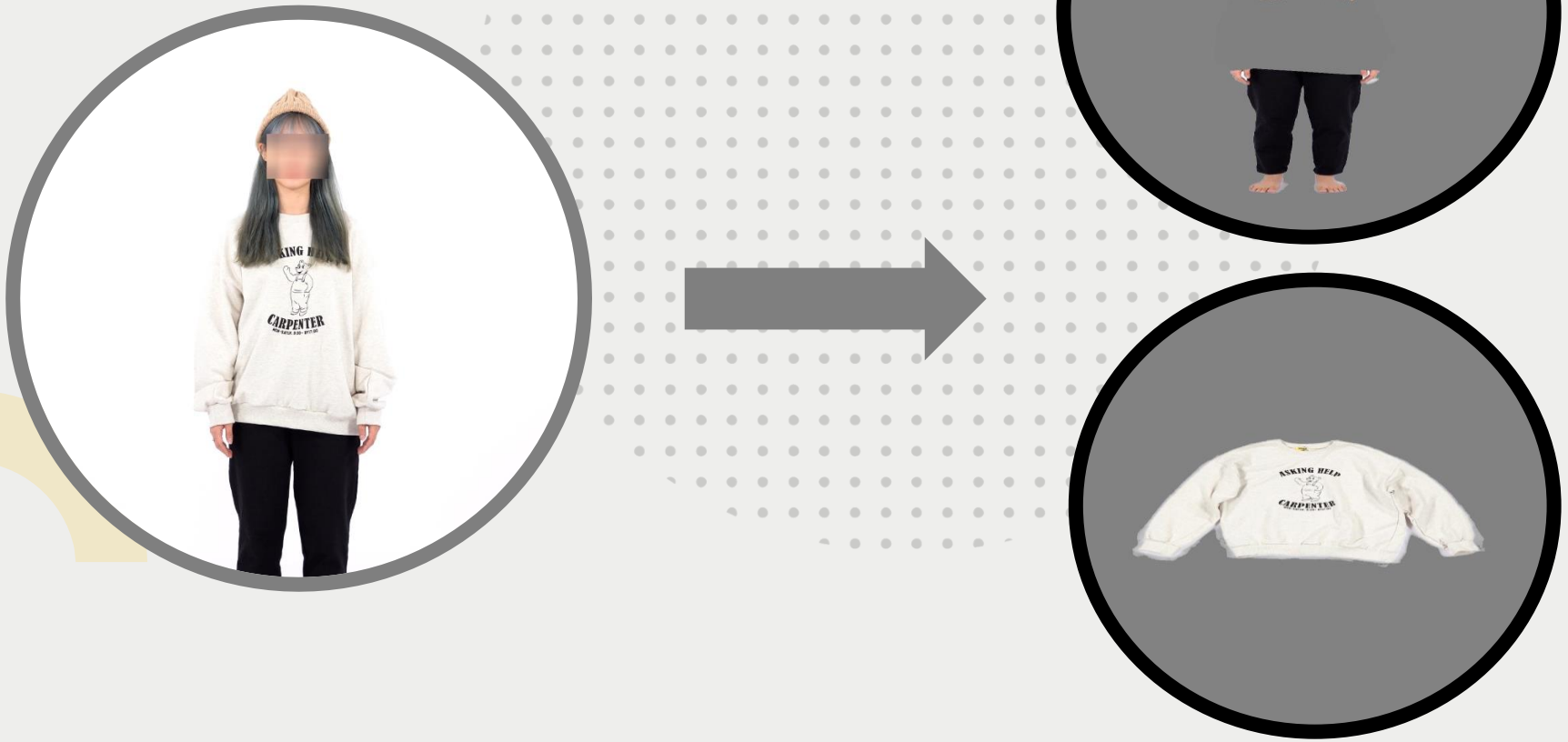
이후 Jp와 Jg의 포즈 키포인트를 [0, 1] 범위로 정규화하여 네트워크 입력으로 사용

ctryon = (Ia, Jp, Ic, Jg)로 정의된 시착 조건 입력 생성

프로젝트 수행 결과

결과 제시 ① 이미지 전처리

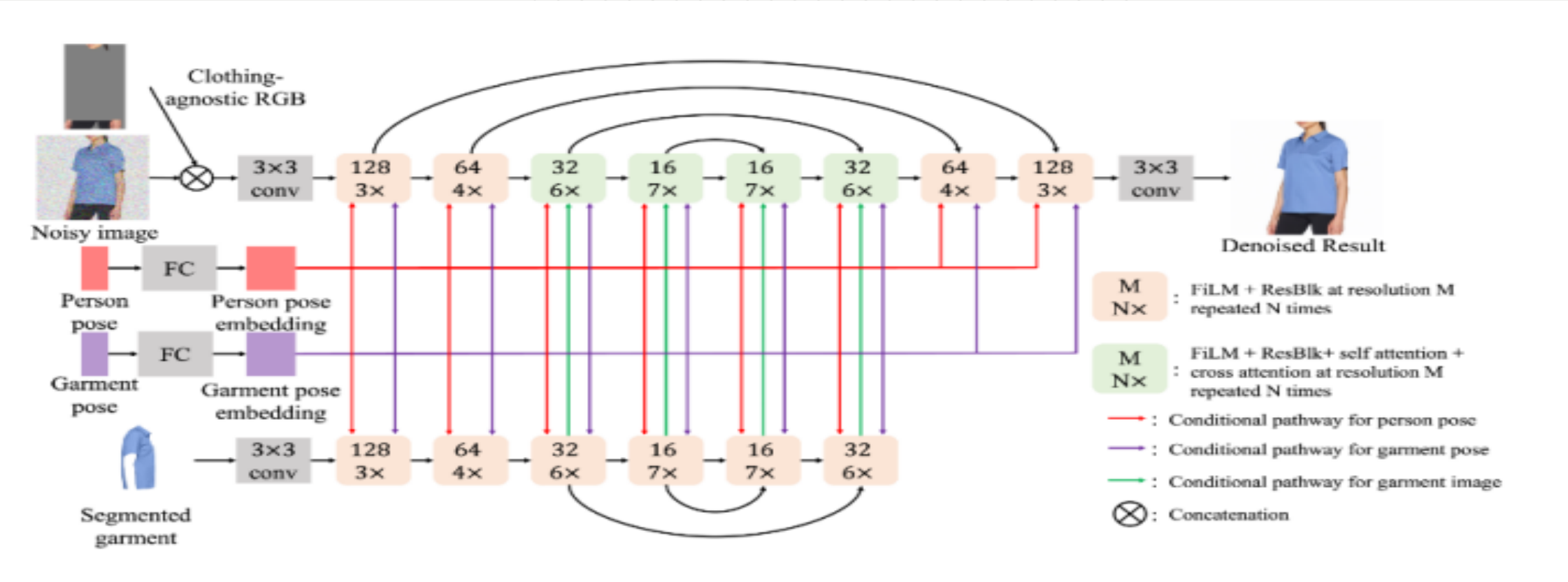
이미지 전처리 프로세스



프로젝트 수행 결과

결과 제시 ② ParallelUNet

ParallelUNet(128 x 128)

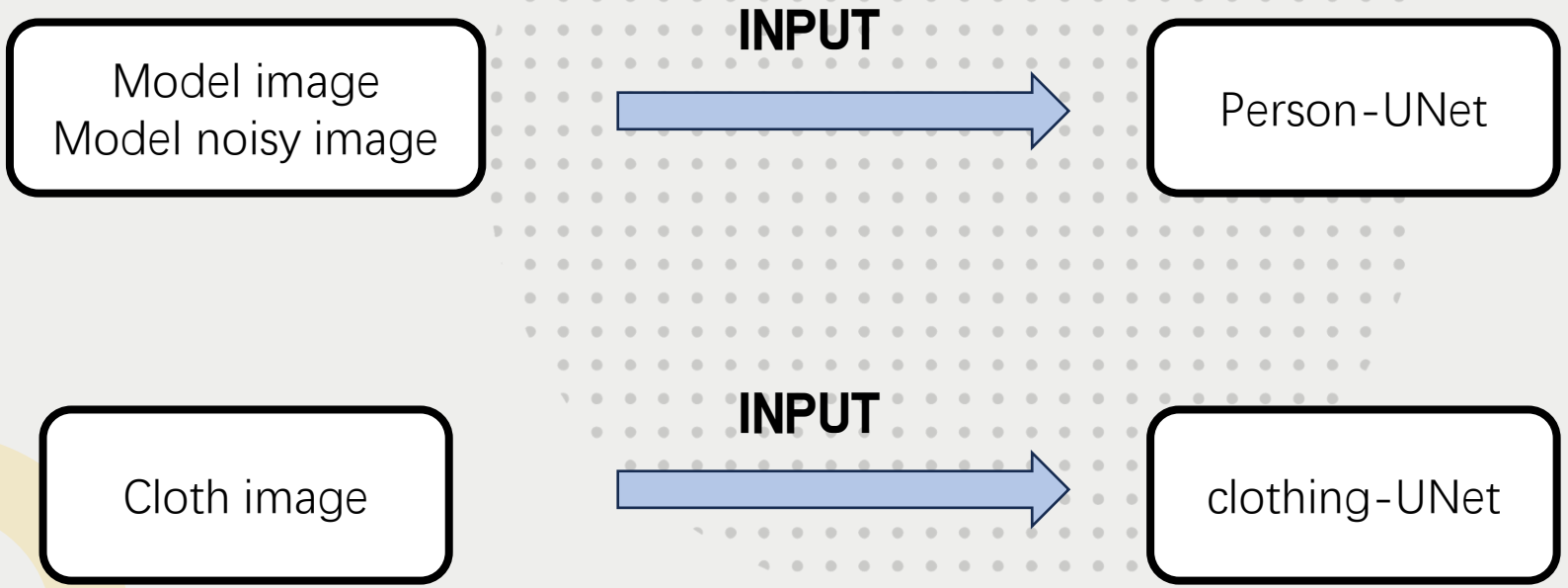


자료 출처 : https://tryondiffusion.github.io/

프로젝트 수행 결과

결과 제시 ② ParallelUNet

ParallelUNet



프로젝트 수행 결과

결과 제시 ② ParallelUNet

Cross Attention & Pose Embedding

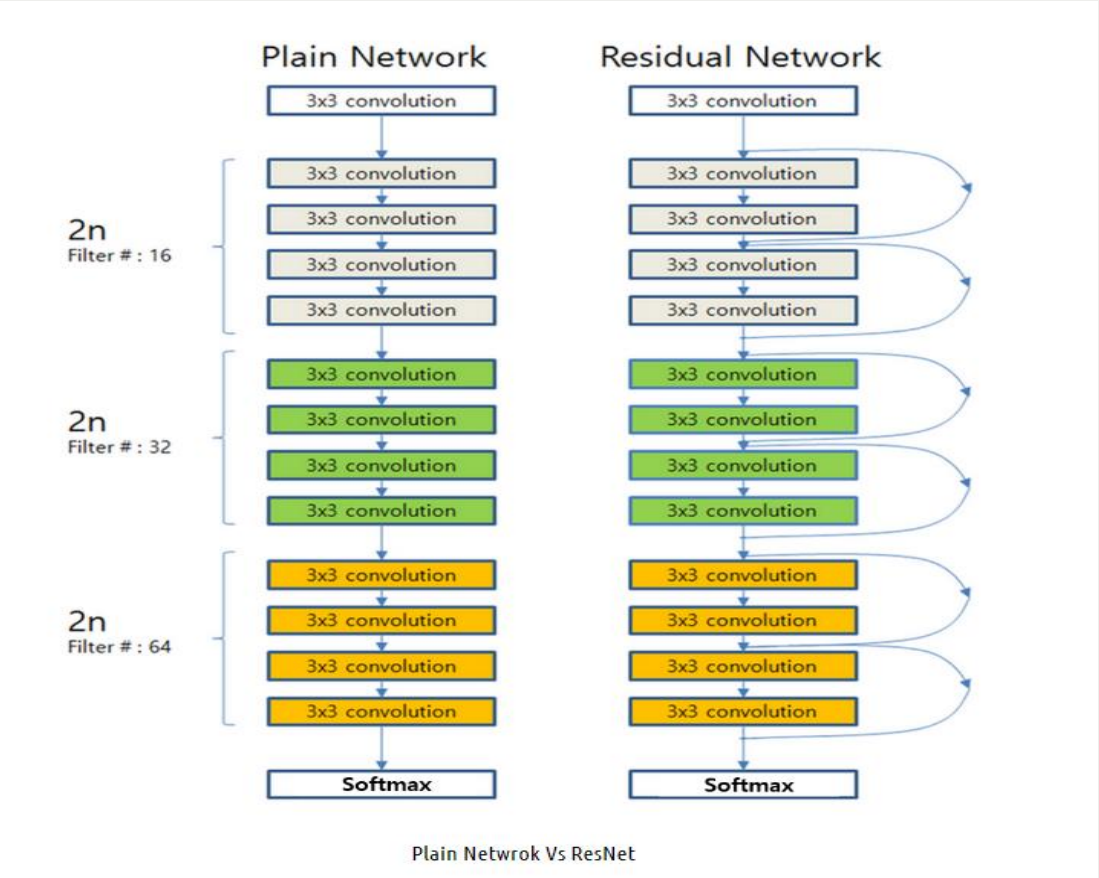
```python
class CrossAttention(nn.Module):
    def __init__(self, channels, num_heads, mlp_ratio=4) -> None:
        super().__init__()

        group = min(channels // 4, 32)
        self.norm1 = nn.GroupNorm(group, channels)

        self.channels = channels
        self.num_heads = num_heads

        self.attn = nn.MultiheadAttention(channels, num_heads)
        self.norm2 = nn.GroupNorm(group, channels)

        self.ffn = nn.Sequential(
            nn.Conv2d(channels, mlp_ratio * channels, kernel_size=1),
            nn.SiLU(),
            nn.Conv2d(mlp_ratio * channels, channels, kernel_size=1)
        )
```

```python
class ParallelUNet(nn.Module):
    def __init__(self, emb_dim, config) -> None:
        super().__init__()
        self.garment_unet = GarmentUNet(emb_dim, config['garment_unet'])
        self.person_unet = PersonUNet(emb_dim, config['person_unet'])


    def forward(self, x, gar_emb, pose_emb, seg_garment):
        g_feature = self.garment_unet(seg_garment, gar_emb)
        deno_x = self.person_unet(x, g_feature, pose_emb)


        return deno_x
```

프로젝트 수행 결과

결과 제시 ② ParallelUNet

Residual Block



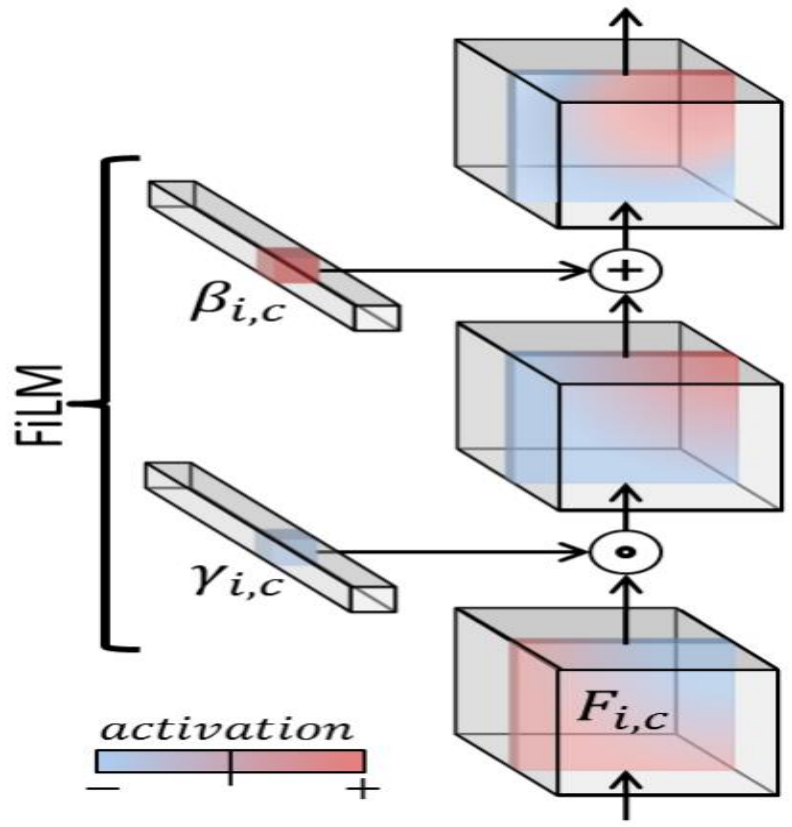
Plain Netwrok Vs ResNet

```python
class ResBlock(nn.Module):
    def __init__(self, channels, kernel_size):
        super().__init__()

        group = min(channels // 4, 32)
        self.nomr1 = nn.GroupNorm(group, channels)
        self.swish1 = nn.SiLU()
        self.conv2d1 = nn.Conv2d(channels, channels, kernel_size, padding=kernel_size // 2)

        self.nomr2 = nn.GroupNorm(group, channels)
        self.swish2 = nn.SiLU()
        self.conv2d2 = nn.Conv2d(channels, channels, kernel_size, padding=kernel_size // 2)

    def forward(self, x):
        skip = x
        x = self.nomr1(x)
        x = self.swish1(x)
        x = self.conv2d1(x)

        x = self.nomr2(x)
        x = self.swish2(x)
        x = self.conv2d2(x)

        return x + skip
```

프로젝트 수행 결과

결과 제시 ② ParallelUNet

FiLM




```python
class FiLM(nn.Module):
    def __init__(self, emb_dim, channels) -> None:
        super().__init__()

        self.fc_scale = nn.Linear(emb_dim, channels)
        self.fc_bias = nn.Linear(emb_dim, channels)

    def forward(self, x, emb):
        batch_size, _, _, _ = x.size()
        emb_flat = emb.view(batch_size, -1)  # Flatten the spatial dimensions
        scale = self.fc_scale(emb_flat).unsqueeze(-1).unsqueeze(-1)
        bias = self.fc_bias(emb_flat).unsqueeze(-1).unsqueeze(-1)

        return x * scale + bias
```

프로젝트 수행 결과

결과 제시 ③ EfficientUNet Architecture 구현

TryOnDiffusion: A Tale of Two Unets → Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

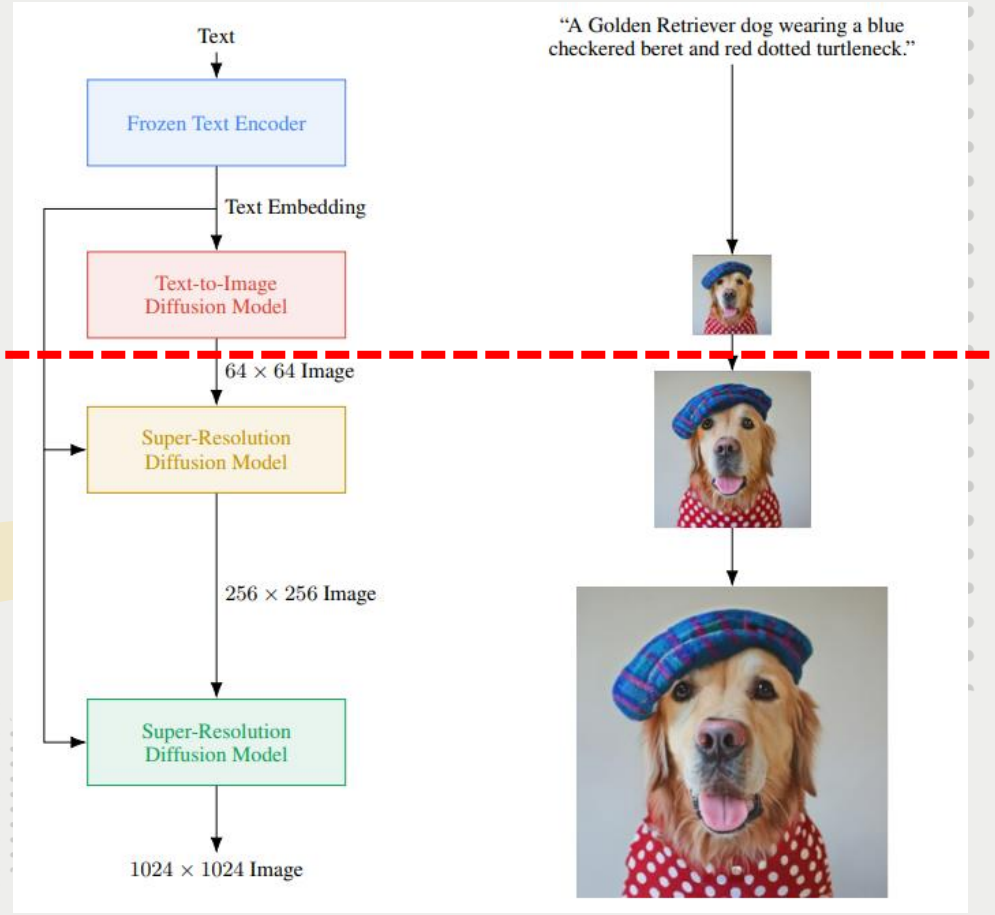
The 256×256→1024×1024 SR diffusion model is parameterized as Efficient-UNet introduced by Imagen [37]. This stage is a pure super-resolution model, with no try-on conditioning. For training, random 256×256 crops, from

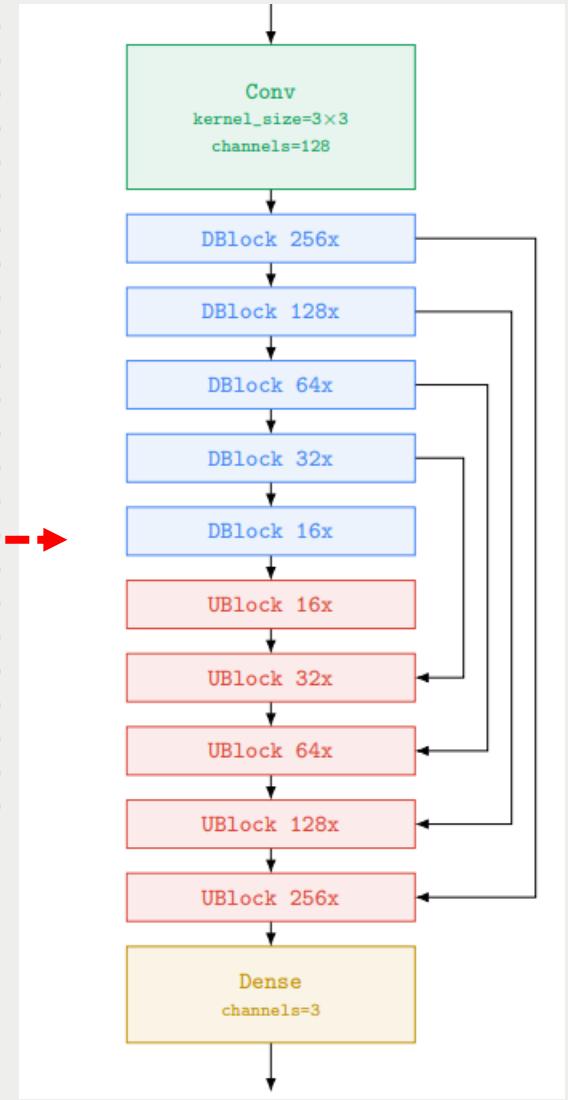
[37] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 2022. 2, 3, 4, 5

프로젝트 수행 결과

결과 제시 ③ EfficientUNet Architecture 구현

EfficientUNet Pipeline

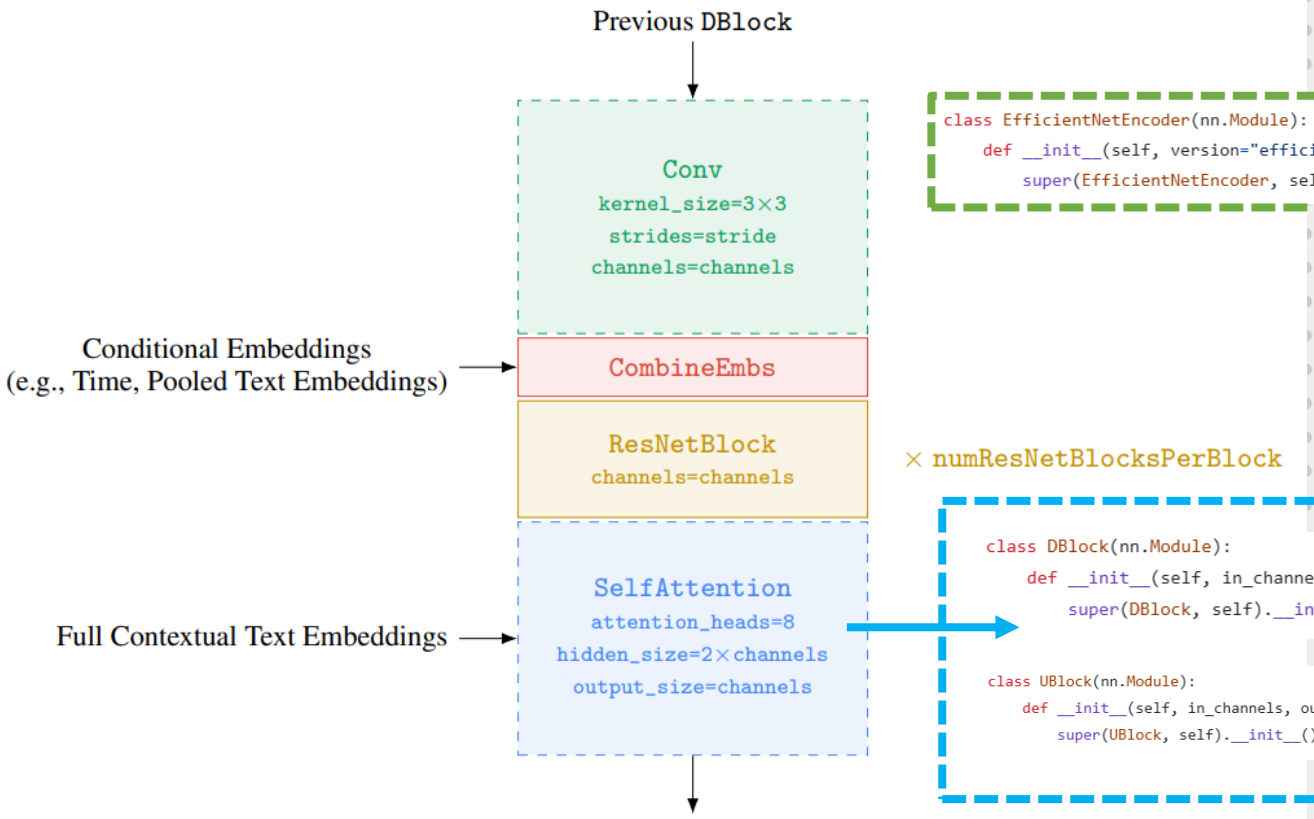




프로젝트 수행 결과

결과 제시 ③ EfficientUNet Architecture 구현

Dblock (Down Sampling)




```python
class EfficientNetEncoder(nn.Module):
    def __init__(self, version="efficientnet_b0", pretrained=True):
        super(EfficientNetEncoder, self).__init__()
```

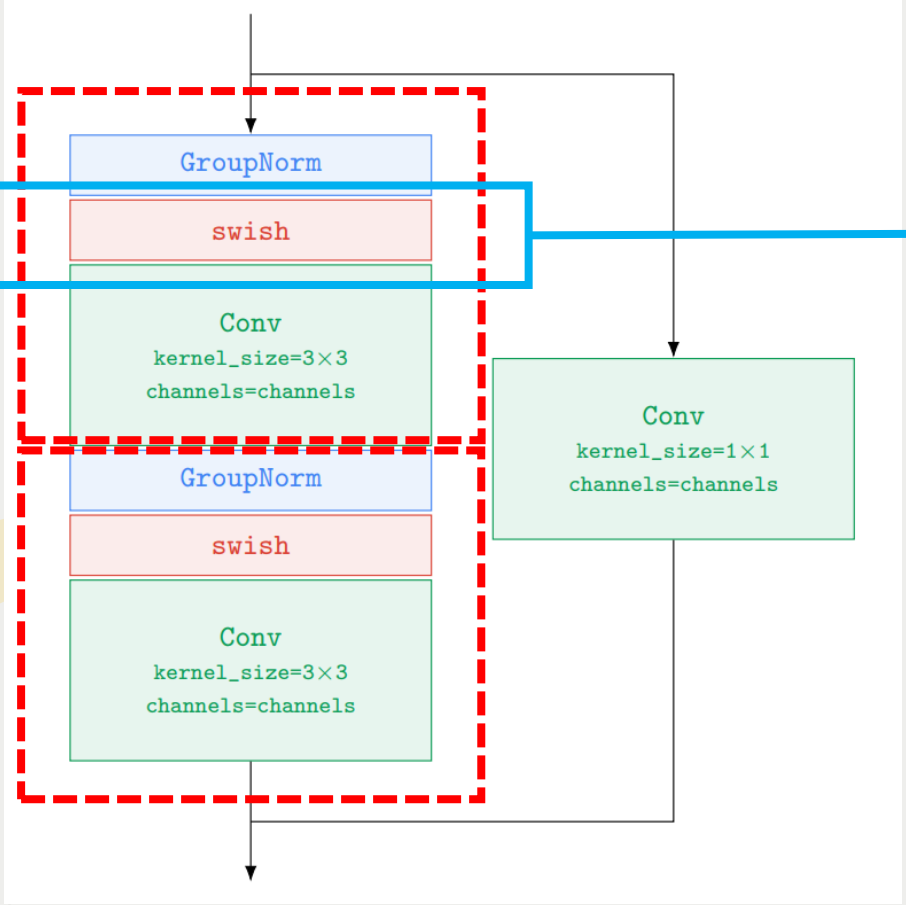
```python
# 각 버전별 U-Net 모델을 반환하는 함수들
def get_efficientunet_b0():
    return EfficientUNet(version="efficientnet_b0")

def get_efficientunet_b1():
    return EfficientUNet(version="efficientnet_b1")

def get_efficientunet_b2():
    return EfficientUNet(version="efficientnet_b2")

def get_efficientunet_b3():
    return EfficientUNet(version="efficientnet_b3")

def get_efficientunet_b4():
    return EfficientUNet(version="efficientnet_b4")

def get_efficientunet_b5():
    return EfficientUNet(version="efficientnet_b5")

def get_efficientunet_b6():
    return EfficientUNet(version="efficientnet_b6")

def get_efficientunet_b7():
    return EfficientUNet(version="efficientnet_b7")
```

```python
class DBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride=(1, 1), numResNetBlocksPerBlock=1, use_self_attention=False):
        super(DBlock, self).__init__()

class UBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride=(1, 1), numResNetBlocksPerBlock=1, use_self_attention=False, use_upsampling=True):
        super(UBlock, self).__init__()
```

프로젝트 수행 결과

결과 제시 ③ EfficientUNet Architecture 구현

ResNetBlock



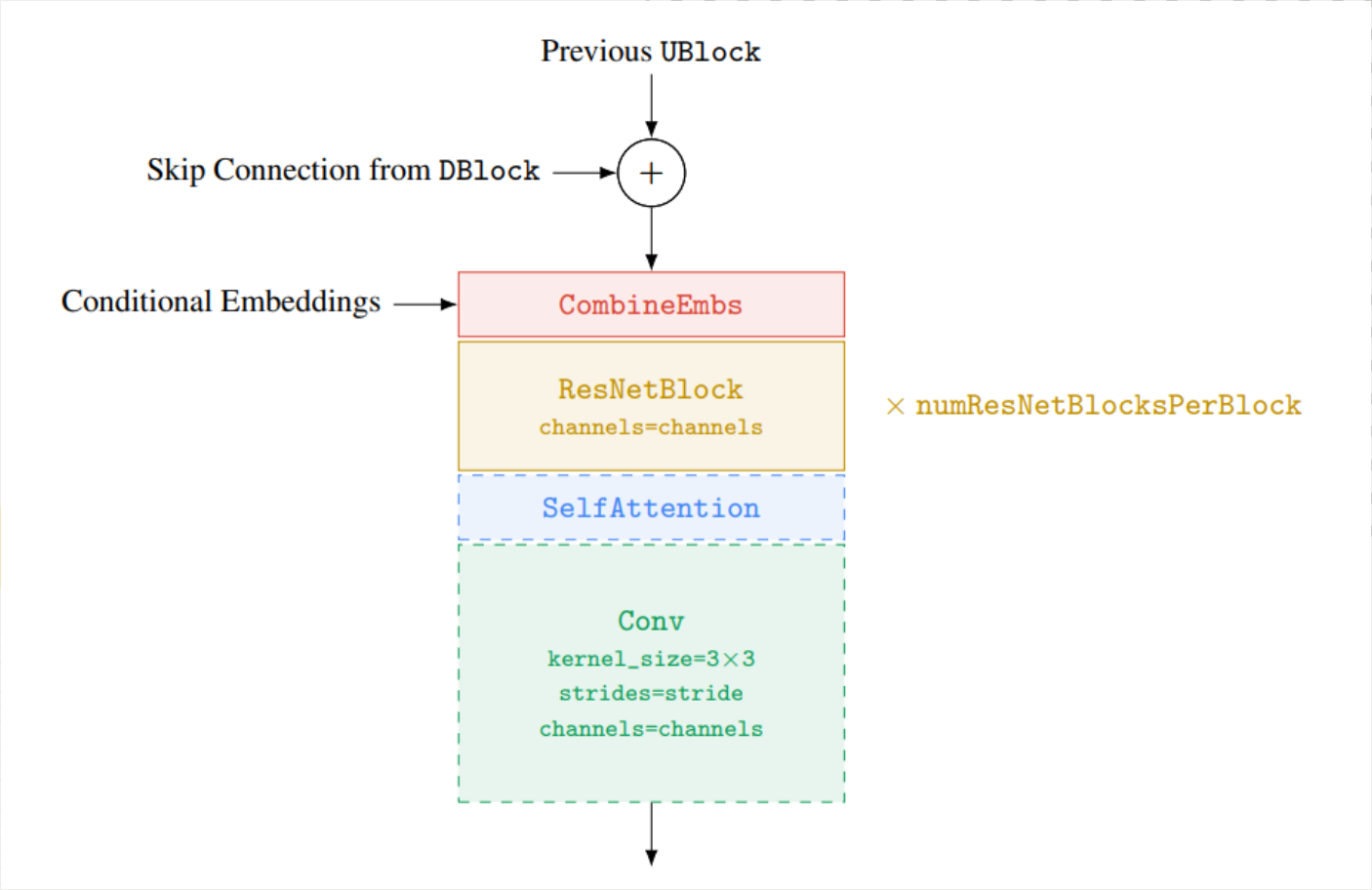

```
class Swish(nn.Module):
    def forward(self, x):
        return x * torch.sigmoid(x)
```

Swish(x)=$x \cdot \sigma(x)$
ReLU(Rectified Linear Unit)의 장점을 유지

프로젝트 수행 결과

결과 제시 ③ EfficientUNet Architecture 구현

UBlock



프로젝트 수행 결과

결과 제시 ③ EfficientUNet Architecture 구현

EfficientUNet

Optimizer: We use the Adafactor optimizer for training the base model. We use the default optax.adafactor parameters. We use a learning rate of 1e-4 with 10000 linear warmup steps.

Diffusion: We use the cosine noise schedule similar to [40]. We train using continuous time steps $t \sim \mathcal{U}(0, 1)$.

↑ 학습률 관련 스케줄러

```
# Convolution layer without downsampling
self.conv = nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=(1, 1), padding=1, bias=False)

# Skip connection for downsampling
self.skip = nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=(1, 1), bias=False)
```

```
# Multiple ResNetBlocks
self.resblocks = nn.Sequential(*[ResNetBlock(in_channels, out_channels) for _ in range(numResNetBlocksPerBlock)])
```

```
"dropout": 0.0,
"feature_pooling_type": "attention",
"use_scale_shift_norm": true,
"blocks"=[
    {
      "channels": 128,
      "strides": (2, 2),
      "kernel_size": (3, 3),
      "num_res_blocks": 2,
    },
    {
      "channels": 256,
      "strides": (2, 2),
      "kernel_size": (3, 3),
      "num_res_blocks": 4,
    },
    {
      "channels": 512,
      "strides": (2, 2),
      "kernel_size": (3, 3),
      "num_res_blocks": 8,
    },
    {
      "channels": 1024,
      "strides": (2, 2),
      "kernel_size": (3, 3),
      "num_res_blocks": 8,
      "text_cross_attention": true,
      "num_attention_heads": 8
    }
]
```

프로젝트 수행 결과

결과 제시 ④ 훈련

Train

```
# Initialize both models and their optimizers
model1 = ParallelUNet(EMB_DIM, parallel_config_128)
model2 = ParallelUNet(EMB_DIM, parallel_config_256)  # Assuming you have a ParallelUNet class
model3 = EfficientUNet()

optimizer1 = optim.AdamW(model1.parameters(), lr=0.0001)
optimizer2 = optim.AdamW(model2.parameters(), lr=0.0001)
optimizer3 = optim.AdamW(model3.parameters(), lr=0.0001, betas=(0.9, 0.999), eps=1e-8, weight_decay=0)

# EfficientUNet에 대한 CosineAnnealing 스케줄러
T_max = 2500000  # 학습 에포크 수에 따라 변하는 하이퍼 파라미터 값입니다.
scheduler3 = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer3, T_max=T_max)

# EfficientUNet에 대한 Linear warmup 스케줄러
warmup_steps = 10000
lr_lambda = lambda epoch: min(1.0, epoch / warmup_steps)
warmup_scheduler3 = torch.optim.lr_scheduler.LambdaLR(optimizer3, lr_lambda)

criterion = torch.nn.MSELoss()  # Mean Squared Error Loss
```

- 각각 만든 모델 연결
- Optimizer : AdamW 사용
- Criterion : MSELoss 사용

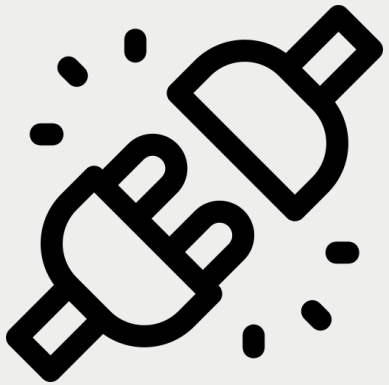
자체 평가 의견

컴퓨터 자원의 한계로
Output을 내지는 못함

Output을 내기 위한 과정 중 3가지의 장애물 존재

자체 평가 의견

1. 모델 연결 문제



```
Shape of x: torch.Size([4, 1280, 4, 4])
Shape of skip_x: torch.Size([4, 512, 2, 2])
```

```
RuntimeError: Expected weight to be a vector of size equal to the number of channels in input, but got weight of shape [32]
and input of shape [4, 16, 4, 4]
```

ParallelUNet Model과 Efficient Model 연결 과정에서
각 Convolution Layer와 Skip Layer의 channel 불일치 발생

자체 평가 의견

2. 데이터셋 문제

RuntimeError: Given input size: (256x1x1). Calculated output size: (256x0x0). Output size is too small

- 우리 모델을 구현하려면 ResNet Model을 통과할 때 Maxpooling Layer를 거쳐야 하는데 이 때 입력 이미지가 사라지는 문제 발생

- 이 문제를 해결하기 위해 데이터셋을 바꿀지 Maxpooling Layer를 없앨 지에서 갈등

-> 시간 문제로 인해 Maxpooling Layer를 없애기로 결정

자체 평가 의견

3. 자원의 한계

```
(AI) C:\Users\labadmin\Downloads\Tryondiffusion-main_2\Tryondiffusion-main> c: && cd c:\Users\labadmin\Downloads\Tryondiffusion-main_2\Tryondiffusio
n-main && cmd /C "C:\Miniconda\envs\AI\python.exe c:\Users\labadmin\.vscode\extensions\ms-python.python-2023.4.1\pythonFiles\lib\python\debugpy\adap
ter/../..\debugpy\launcher 54898 -- c:\Users\labadmin\Downloads\Tryondiffusion-main_2\Tryondiffusion-main\train.py "
Using device: cuda:0
1695003258.3609002
```

```
  File "C:\Users\labadmin\PycharmProjects\pythonProject\parallelUNet.py", line 228, in forward
    x = self.nomr1(x)
  File "C:\Miniconda\envs\my_onnx_env\lib\site-packages\torch\nn\modules\module.py", line 1501, in _call_impl
    return forward_call(*args, **kwargs)
  File "C:\Miniconda\envs\my_onnx_env\lib\site-packages\torch\nn\modules\normalization.py", line 273, in forward
    return F.group_norm(
  File "C:\Miniconda\envs\my_onnx_env\lib\site-packages\torch\nn\functional.py", line 2530, in group_norm
    return torch.group_norm(input, num_groups, weight, bias, eps, torch.backends.cudnn.enabled)
torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 2.00 MiB (GPU 0; 15.85 GiB total capacity; 14.86 GiB already allocated; 2.06 MiB free; 14.90 GiB reserved in

Process finished with exit code 1
```

Future Work

더 좋은 GPU 성능을
가진 컴퓨터를 마련하여
모델 구현

논문에서 사용했던
HD-VITON 데이터셋을
사용하여 모델 구현


BEAUTY
THANK YOU