team. 4E다 신은빈 오효근 장우솔 최가희

MINIPROJECT

Image-to-Illustration

[참고논문]

GANILLA: Generative adversarial networks for image to illustration translation

목차

01 Introduction

02 Data

03 Model

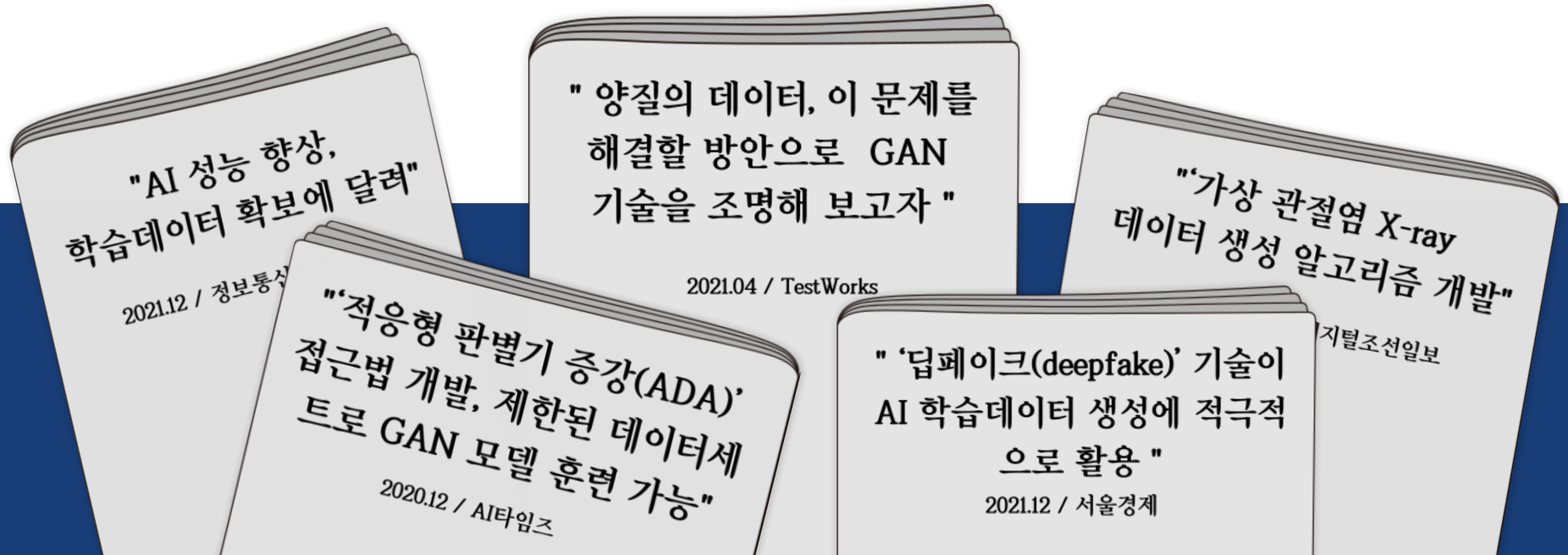
04 Source code

05 Result & Discussion

1.1 주제 선정

GAN을 활용한 Style Transfer

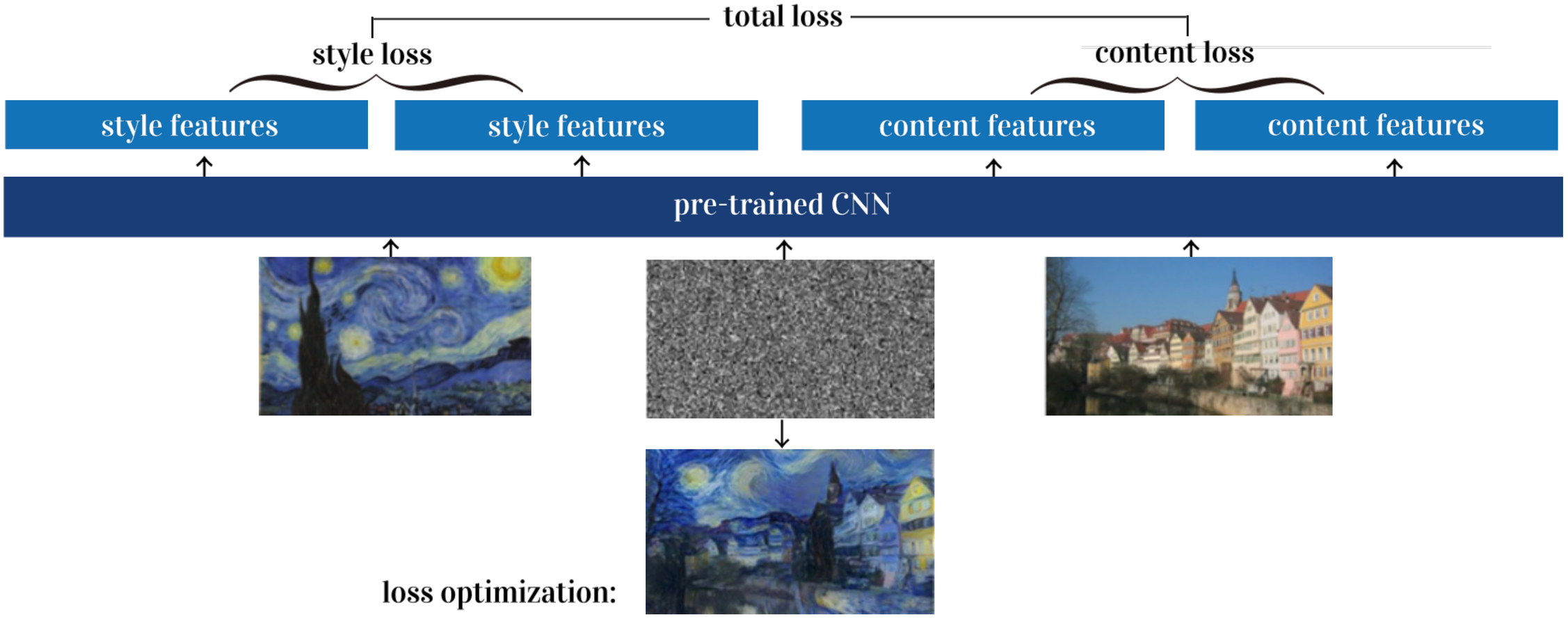
- 학습용 데이터의 양과 품질이 AI 성능을 좌우 → 다량의 고품질 데이터 확보에 대한 현실적인 한계점 존재
- 학습데이터 증강(data augmentation)을 위한 여러 기법 발달
- 데이터 '생성'을 위한 '합성' 기법이 많이 활용되고 있어, GAN 기반의 Style Transfer 모델 개발이 활발히 진행 중



1.2 Style Transfer

Style Transfer?

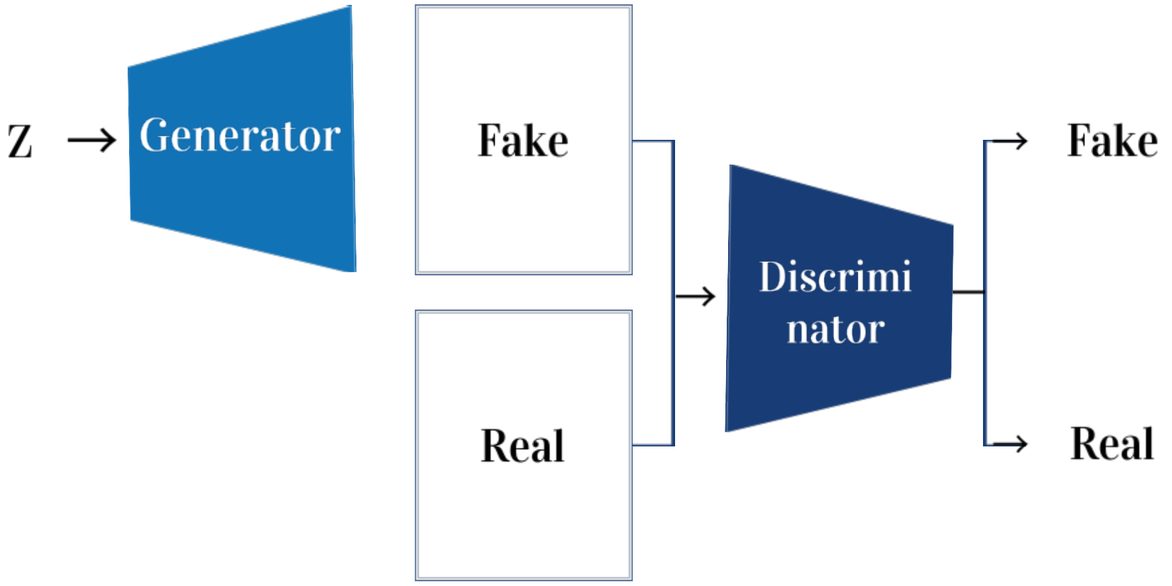
- 두 이미지(content & style)가 주어졌을 때 주된 형태는 content image를 유지하면서 스타일만 style image처럼 바꾸는 것



1.3 GAN

Generative Adversarial Network?

● 각각의 역할을 가진 두 모델(생성기 및 판별기)을 통해 적대적 학습을 하면서 '진짜 같은 가짜'를 생성해내는 것



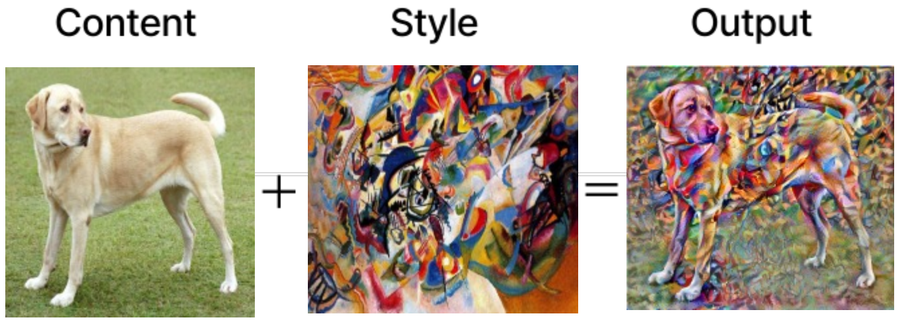
[gan 구조도]

[gan 으로 표현한 로마 황제 초상]

1.4 한계점

Neural Style Transfer

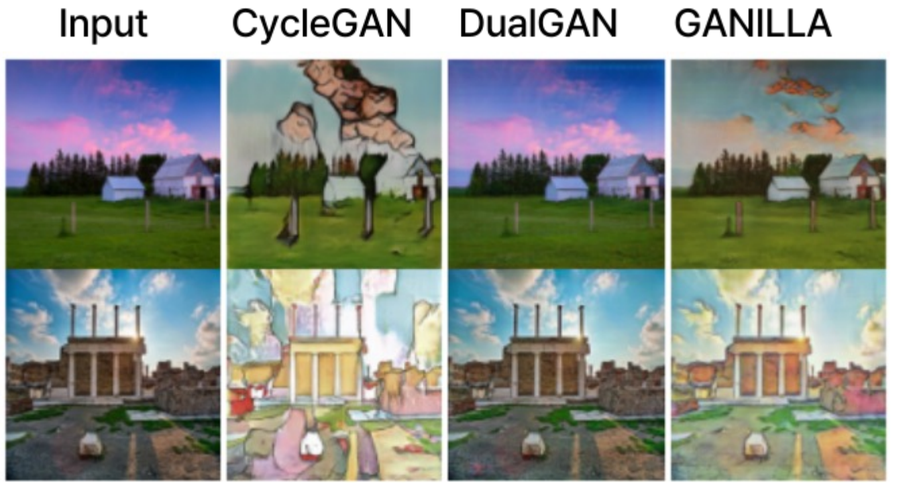
- 이미지마다 gram matrix 연산 필요
- 하나의 소스 데이터만 사용하기 때문에 좋은 결과 X
- Color 정보는 정확 but, ‘style’을 반영한다고 하기에는 어려움



[NST를 통한 스타일 전이]

Conventional GAN

- CycleGAN: 스타일을 잘 전달하지만 기존의 콘텐츠 변형 발생
- CartoonGAN & DualGAN : 콘텐츠를 잘 보존하지만 스타일 전달 저조



[gan 모델 별 스타일 전이]

2. Data

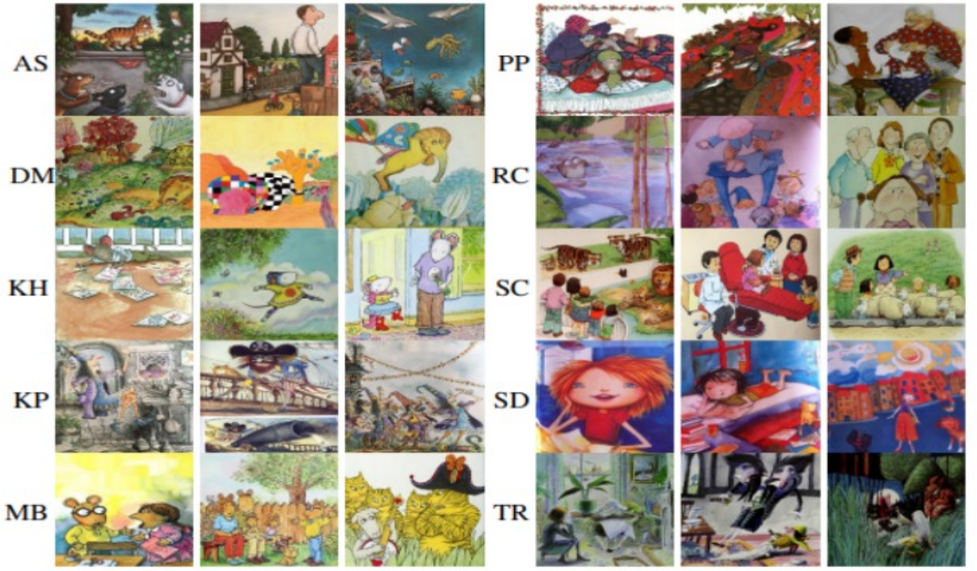
Datasets

- 24명의 아티스트에 대한 약 9500개의 illustration으로 구성
- Unpairde data
  - source domain: content 정보를 가져올 이미지
  - target domain: illustration, style 정보를 가져올 이미지

[source domain]

Id	Illustrator	Book Cnt	Image Cnt	Id	Illustrator	Book Cnt	Image Cnt
AS	Axel Scheffler	15	571	PP	Patricia Polacco	20	756
DM	David McKee	20	415	RC	Rosa Curto	10	336
KH	Kevin Henkes	11	278	SC	Stephen Cartwright	25	405
KP	Korky Paul	13	362	SD	Serap Deliorman	5	158
MB	Marc Brown	18	461	TR	Tony Ross	21	521

[target domain data 구성]

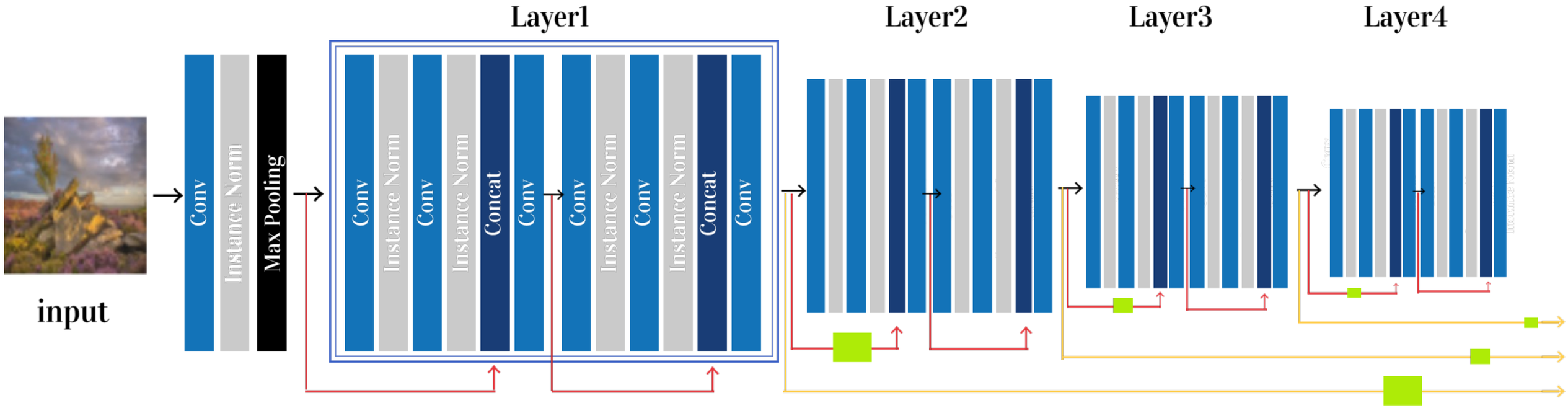


[target domain]

3.1 Model - Generator

Downsampling

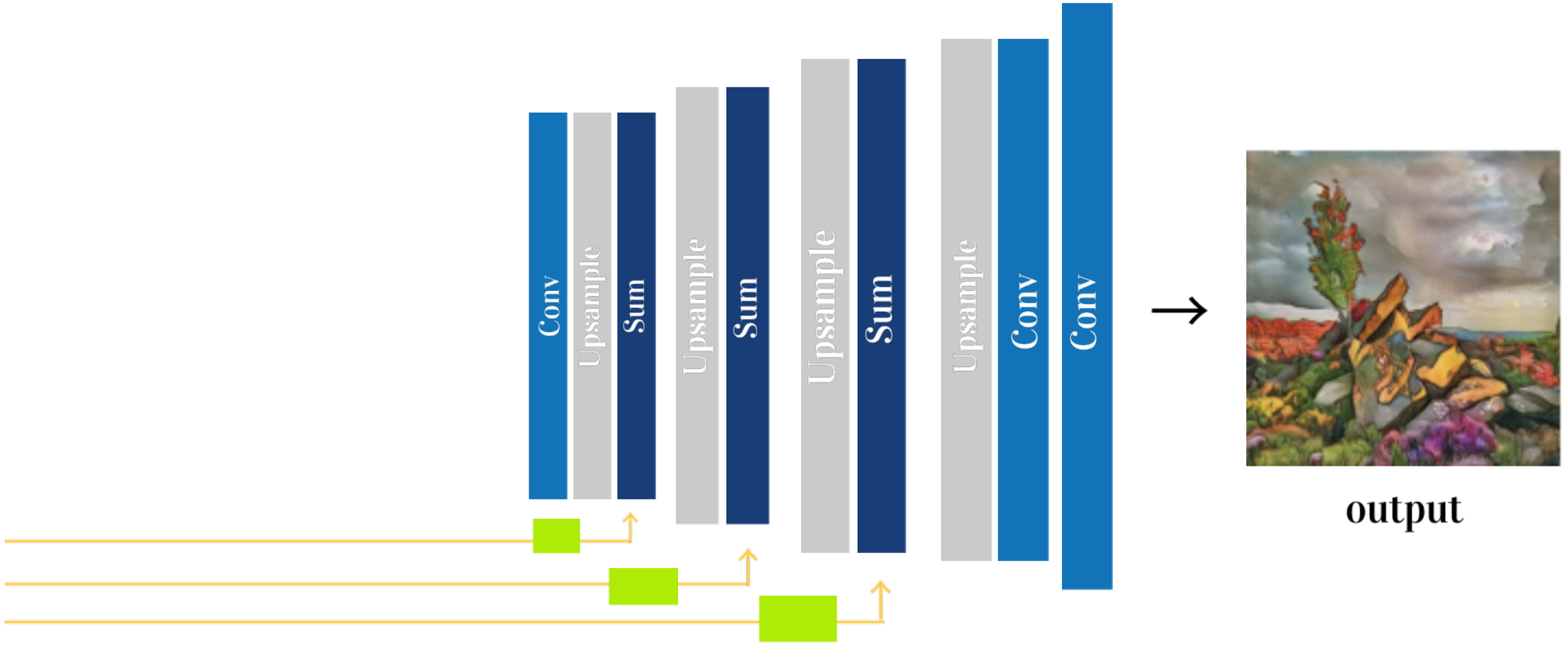
- 수정된 ResNet-18
- 저수준 feature를 통합하기 위해 다운 샘플링의 각 레이어에서 이전 레이어의 feature 연결
- 저수준 레이어는 형태적 특징, 가장자리 및 모양과 같은 정보 통합



3.1 Model - Generator

Upsampling

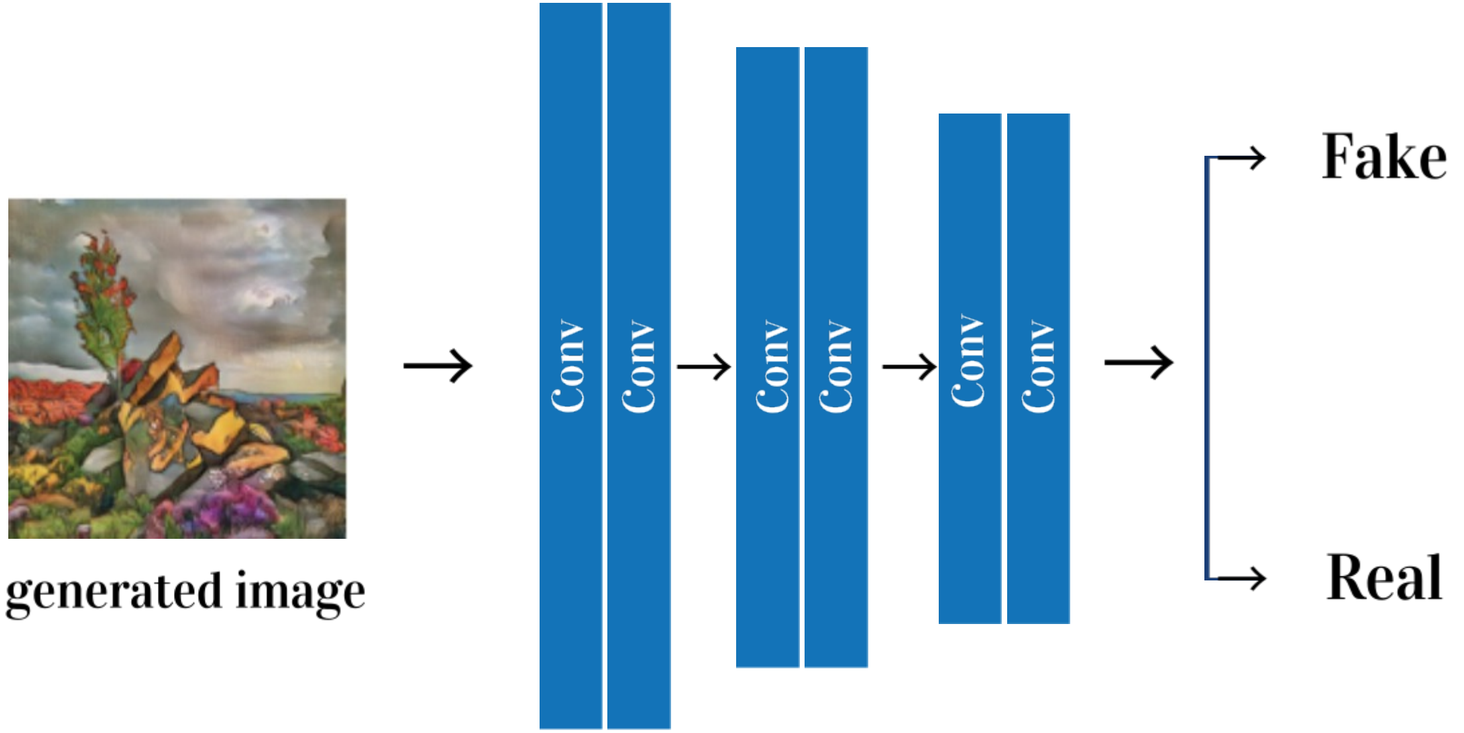
- summation layers에 skip-connection을 통해 low-level feature 생성
- 콘텐츠를 보존에 효과적



3.2 Model - Discriminator

Discriminator Network

- 70 × 70 PatchGAN
- PatchGAN: image-to-image translation에 성공적으로 사용된 모델



3.3 Evaluation

Style-CNN

- style 전달 측면의 성능 측정
- 스타일을 유지하며 시각적 콘텐츠에서 훈련 이미지 분리
- style 이미지에서 무작위로 자른 작은 패치(100 × 100)를 사용하여 스타일 분류기 훈련
- 11개 class: 10개 - illustration ,1개-natural image

Content-CNN

- contents 보존 측면의 성능 측정
- 특정 장면 범주 (숲, 거리 등)을 콘텐츠로 정의
- ex) 특정 스타일로 산 이미지를 생성한다면 생성 이미지 또한 산 이미지로 분류
- data: 4150개의 훈련 이미지, 500개의 테스트 이미지 사용

4.1 Source code - Model

Generator - Downsampling

- **Pretrained Model : ResNet-18**
- **' Class AblationModel1' 내 외부 url 호출하여 ResNet-18 import 하는 함수 정의**

```
def resnet18 (input_nc, output_nc, ngf, fpn_weights, use_dropout, pretrained=False, **kwargs):
    """ Constructs a ResNet-18 model.
    Args:
        pretrained (bool): If True, returns a model pre-trained on ImageNet
    """
    model = ResNet(input_nc, output_nc, ngf, fpn_weights, BasicBlock_Ganilla, [2, 2, 2, 2], use_dropout, **kwargs)
    if pretrained:
        model.load_state_dict(model_zoo.load_url(model_urls['resnet18'], model_dir='.'), strict=False)
    return model
```

4.1 Source code - Model

Generator - Downsampling

- **class ResidualBlock 정의 : 실질 downsampling layer**

```
class ResidualBlock(nn.Module):
    def __init__(self, in_features):
        super(ResidualBlock, self).__init__()
        conv_block = [ nn.ReflectionPad2d(1),
                       nn.Conv2d(in_features, in_features, 3),
                       nn.InstanceNorm2d(in_features),
                       nn.ReLU(inplace=True),
                       nn.ReflectionPad2d(1),
                       nn.Conv2d(in_features, in_features, 3),
                       nn.InstanceNorm2d(in_features) ]
        self.conv_block = nn.Sequential(*conv_block)
    def forward(self, x):
        return x + self.conv_block(x)
```

- **class Generator 내 downsampling 정의**

```
# Downsampling
        in_features = 64
        out_features = in_features*2
        for _ in range(2):
            model +=
            [ nn.Conv2d(in_features, out_features, 3,
              stride=2, padding=1),
               nn.InstanceNorm2d(out_features),
               nn.ReLU(inplace=True) ]
            in_features = out_features
            out_features = in_features*2
        # Residual blocks
        for _ in range(n_residual_blocks):
            model += [ResidualBlock(in_features)]
```

× ④

4.1 Source code - Model

Generator - Upsampling

- **class Generator 내 upsampling 정의**

  : downsampling 각 layer 기반 low-level feature 제작

```
        out_features = in_features//2
        for _ in range(2):
            model +=
            [ nn.ConvTranspose2d (in_features,
             out_features, 3, stride=2, padding=1,
             output_padding=1),
             nn.InstanceNorm2d(out_features),
             nn.ReLU(inplace=True) ]
             in_features = out_features
             out_features = in_features//2
```

- **Output layer**

```
# Output layer
        model += [ nn.ReflectionPad2d(3),
                   nn.Conv2d(64, output_nc, 7),
                   nn.Tanh() ]
        self.model = nn.Sequential(*model)
```

4.1 Source code - Model

Discriminator

- class Discriminator 로 정의
- 이미지를 변환에 성공한 모델에서 가져온

  70x70 PatchGAN : 네개의 convolution block 존재
- 각 convolution block = 2개의 convolution layer
- filter size: 64x64 로 시작하여 두 배 씩 증가

```
class Discriminator(nn.Module):
    def __init__(self, input_nc):
        super(Discriminator, self).__init__()
        # A bunch of convolutions one after another
        model = [ nn.Conv2d(input_nc, 64, 4, stride=2, padding=1),
                  nn.LeakyReLU(0.2, inplace=True) ]
        model += [ nn.Conv2d(64, 128, 4, stride=2, padding=1),
                   nn.InstanceNorm2d(128),
                   nn.LeakyReLU(0.2, inplace=True) ]
        model += [ nn.Conv2d(128, 256, 4, stride=2, padding=1),
                   nn.InstanceNorm2d(256),
                   nn.LeakyReLU(0.2, inplace=True) ]
        model += [ nn.Conv2d(256, 512, 4, padding=1),
                   nn.InstanceNorm2d(512),
                   nn.LeakyReLU(0.2, inplace=True) ]
```

4.2 Source code - Evaluation

Style CNN

- **Pretrained model: ResNet-50**
  - **[3, 4, 6, 4] layer 구성**
- **외부 url 에서 import 하는 함수 정의**

```
def resnet50(pretrained=False, **kwargs):
    """Constructs a ResNet-50 model.
     Args:
          pretrained (bool): If True, returns a model
          pre-trained on ImageNet
    """
    model = ResNet(Bottleneck, [3, 4, 6, 3], **kwargs)
    if pretrained:
       model.load_state_dict(model_zoo.load_url
      (model_urls['resnet50']), strict=False)
    return
```

- **Image pre-processing**
  - **illustration imgae 100x100 무작위 crop**
  - **padding_mode="reflect" : 가장자리에서 반복하지 않고 반사**

```
transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.RandomCrop(cfg.PATCH_SIZE),
    # n_random_crops.NRandomCrop(cfg.PATCH_SIZE, 10),
    # transforms.Resize((cfg.RESIZE, cfg.RESIZE)),
    # avgpool u 3x3 yapinca 96 olmali.
    transforms.Pad((224-cfg.PATCH_SIZE)/2,
                   padding_mode="reflect"),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))])
```

4.2 Source code - Evaluation

Style CNN

● **class ArtModel 내 신경망 정의**

- **앞서 정의한 함수를 통해 ResNet-50 import**

```
class ArtModel(nn.Module):
    def __init__(self):
        """Load the pretrained ResNet-152 and replace top fc layer."""
        super(ArtModel, self).__init__()
        model_urls['resnet50'] =
        model_urls['resnet50'].replace('https://', 'http://')
        self.resnet = models.resnet50(pretrained=True)
```

- **11개 CLASS_LIST 정의 : 1개 photo, 10개 illustration**

```
        # self.__freeze_network__()
        # Newly created modules have require_grad=True by default
        num_features = self.resnet.fc.in_features
        self.resnet.fc = nn.Linear(num_features, cfg.CLASS_COUNT)
        print self.resnet # display model
    def __freeze_network__(self):
        # Freeze training for all layers
        for param in self.resnet.features.parameters():
            param.require_grad = False
    def forward(self, images):
        vgg_out = self.resnet(images)
        return vgg_out
```

4.2 Source code - Evaluation

Content CNN

- **class ContentModel 내 신경망 정의 :Style CNN과 동일한 구조**
  - **CLASS_LIST 수정 : ["all_ills", "beach", "building_facade" ,,,] 등 콘텐츠 클래스 12개**

```
class ArtModel(nn.Module):
    def __init__(self):
        """Load the pretrained ResNet-152 and replace top fc layer."""
        super(ArtModel, self).__init__()
        model_urls['resnet50'] =
        model_urls['resnet50'].replace('https://', 'http://')
        self.resnet = models.resnet50(pretrained=True)
```

```
        # self.__freeze_network__()
        # Newly created modules have require_grad=True by default
        num_features = self.resnet.fc.in_features
        self.resnet.fc = nn.Linear(num_features, cfg.CLASS_COUNT)
        print self.resnet # display model
    def __freeze_network__(self):
        # Freeze training for all layers
        for param in self.resnet.features.parameters():
            param.require_grad = False
    def forward(self, images):
        vgg_out = self.resnet(images)
        return vgg_out
```

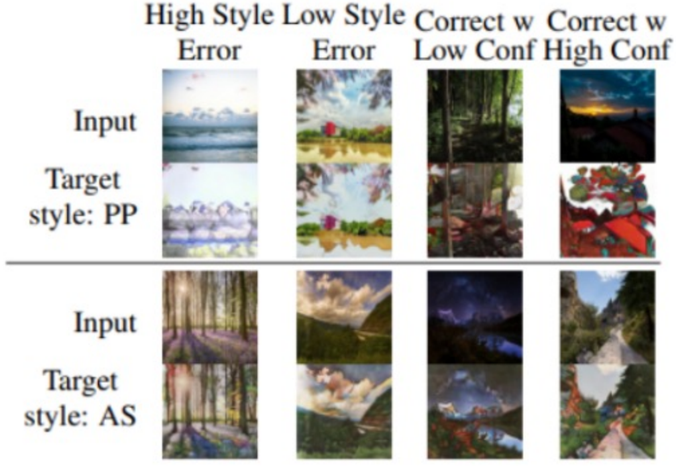
5.1 Result - Evaluation

Qualitative Analysis : 제안한 Evaluation model 로 측정

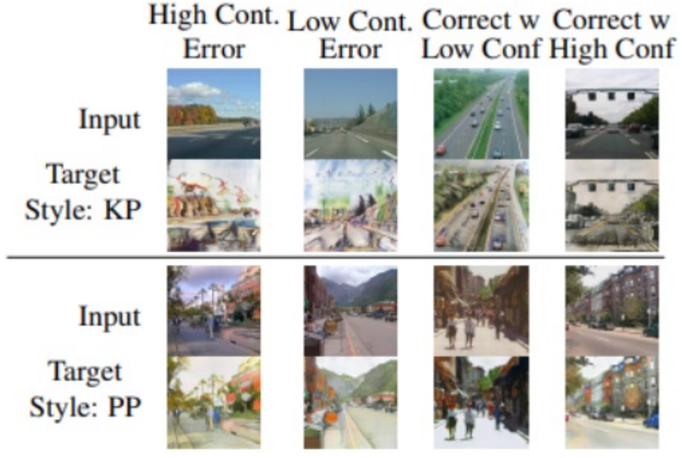
- CycleGAN, DualGAN, CartoonGAN 보다 월등한 성능

	CartoonGAN	CycleGAN	DualGAN	GANILLA
Style	40.9	44.4	54.5	**60.0**
Content	67.0	69.4	68.1	**70.0**
Content & Style	54.0	56.9	61.3	**65.0**
Avg. Ranking	2.68	2.89	**2.18**	2.21

Quantitative Analysis



[Style]



[Content]

5.2 Result - Conclusion

의의

[멘토 에러]

● Model 측면

- 기존: content와 style을 균형있게 전달 및 합성하지 못함.
- GANILLA: Downsampling 및 Upsampling 부분에서 low-level feature 에 주목.

● Evaluation 측면

- 기존: Image-to-image translation domain에서 수치적 평가의 부재
- 평가 프레임워크 제시: 스타일과 콘텐츠 측면을 별도로 측정하는 두 개의 CNN

한계점

● 스타일화 측면에서 일부 삽화에서 실패

● dataset의 단순한 채색 및 illustration에 사람, 동물 주로 포함.