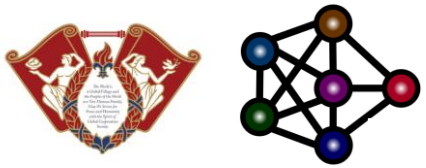
실전기계학습 기말 프로젝트 소개

: Descanning Competition

2023. 05. 19

JungHun Cha



Kyung Hee University
Department of Computer Science and Engineering
Machine Learning and Visual Computing (MLVC) Laboratory

Content

- **대회 일정 및 규칙 안내**
- 이미지 복원 (Image Restoration) 소개
- 디스캐닝 (Descanning) 소개
  - 디스캐닝 Task 및 데이터셋 설명
  - 왜 디스캐닝인가?
- 이미지 복원 모델 소개
  - 초해상화 (Super-Resolution): SRCNN
  - JPEG 압축 열화 제거 (Deblocking): ARCNN
  - 노이즈 제거 (Denoising): DnCNN
- 학습 전략 소개
  - 데이터 관점: 데이터 증강 (Data Augmentation)
  - 모델 관점: 어텐션 구조 (Attention)
  - 손실 함수 관점: 지각 손실 함수 (Perceptual Loss)
- 노이즈 제거 모델 코딩 실습: DnCNN

대회 일정 및 규칙 안내

- 목표 : 주어진 데이터를 훈련 및 최적화하여 **Descanning performance**가 높은 모델 만들기
- 일정 : 총 16일 (모델 훈련 15일 + 보고자료 1일)
  - 모델 훈련 일정 : 2023.05.22 (월) 12:00 ~ 2023.06.07 (수) 23:59
  - 팀 구축 : 학생 자율로 3명으로 구성, 2023.05.21 (일) 23:59 까지 인원 확정 후 조교에게 메일 송부
  - (ex. 2020xxx 김oo, 2021xxx 박oo, 2019xxx 최oo 이 같이 competition 하겠습니다)
  - 메일 송부하지 않은 학생들은 조교가 랜덤하게 팀당 3명이 되도록 인원 배분하여 팀 구성 (경우에 따라 2인 or 4인팀 발생 가능)
  - 05.22 (월) 10:00 ecampus 팀 공지 / 05.22 (월) 12:00 대회 시작 (ecampus에 Kaggle 링크 업로드)
- 발표자료: 2023.06.08 (목) 23:59 까지 조교 이메일로 제출(PPT + 코드 파일)
- 2023.06.09 (금) 수업시간에 하위권 팀부터 발표 → 마지막 시간에 1위 팀 발표

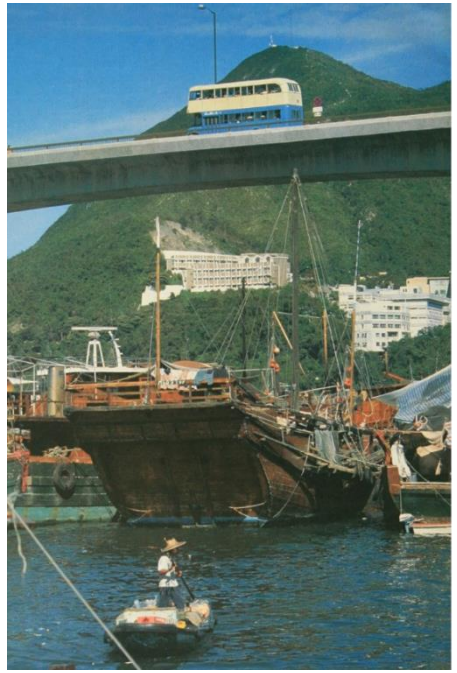
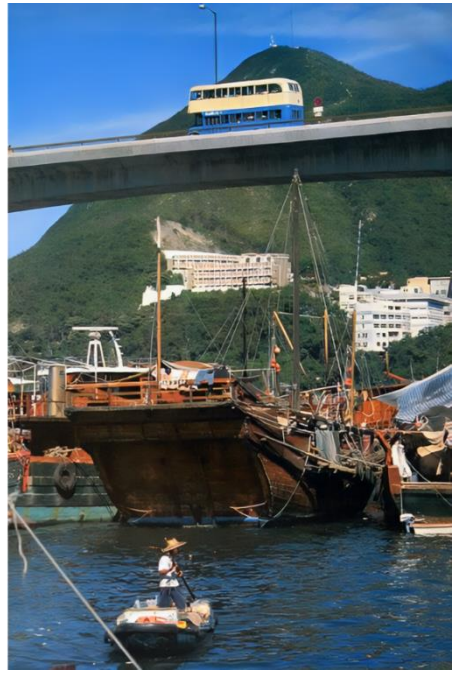
대회 일정 및 규칙 안내

- 평가지표 : 디스캐닝 된 (복원된) 이미지의 Y채널 값의 평균에 대한 MAE (Mean Absolute Error)
- 제출방법 : Kaggle 사이트에 csv 파일 제출 (하루 최대 3번)
- 훈련 조건:
  - 모델 파라미터 개수 10M (1천만) 개 미만으로 제한
  - Pretrained Model 사용 금지
  - Personal Data 추가 금지
  - Test Data 사용 금지

이미지 복원 (Image Restoration) 소개

- 이미지 복원
  - **열화된 이미지**에 있는 노이즈, 색상 바램 등을 제거하여 **원본 이미지**로 복원하는 작업
  - 즉, 이미지 분류와 달리, 이미지 복원에서는 인풋: 이미지 / 아웃풋: 이미지 / 레이블: 이미지
  - 딥러닝 등장 이전부터 전통적인 영상 처리 기법 등을 통해 발전해온 분야 (Low-level vision)

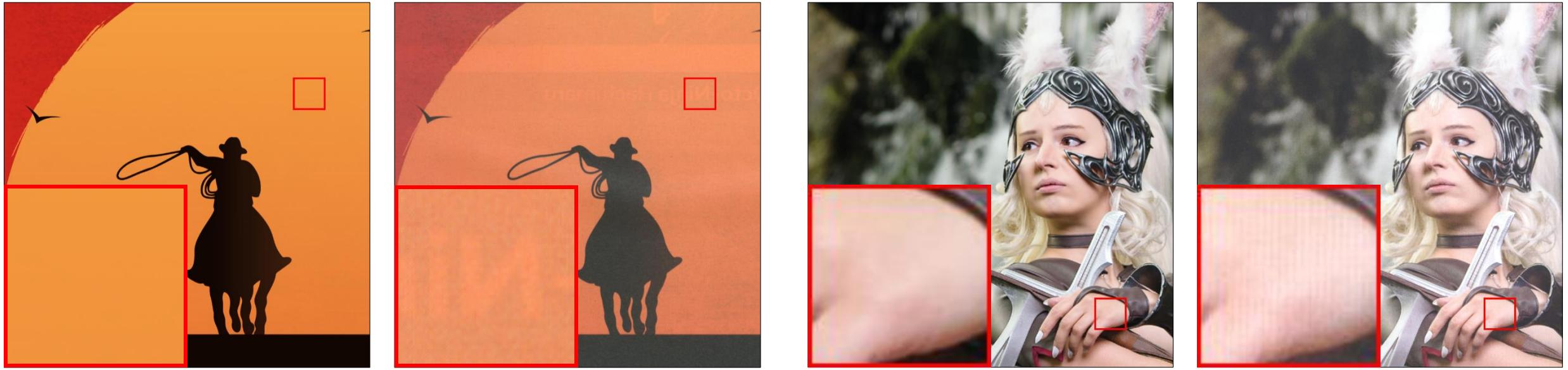


[1] Wan *et al.*, “Bringing Old Photos Back to Life”, CVPR, 2020

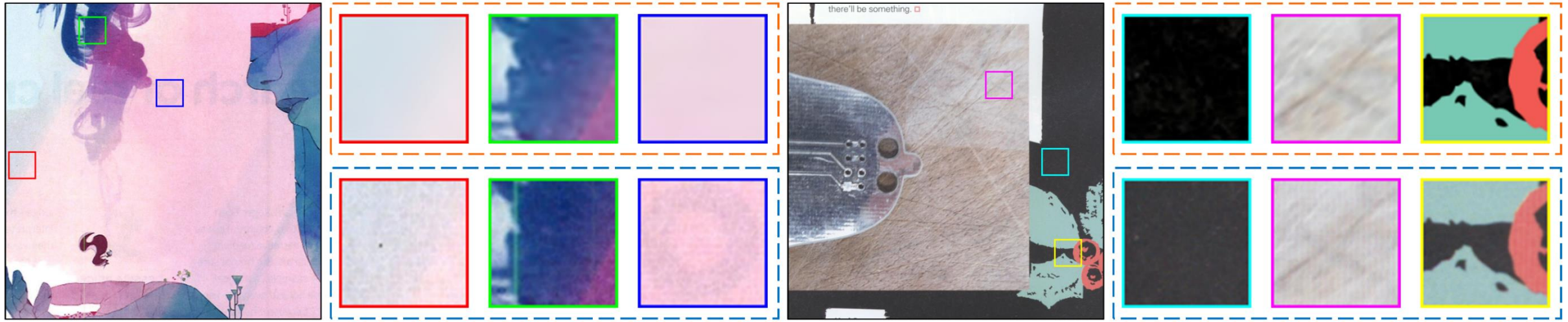
디스캐닝 (Descanning) 소개

- 디스캐닝 Task 및 데이터셋 소개
  - **디스캐닝** : 스캔 이미지에 있는 여러 화질 열화 요소들을 제거하여 원본에 가깝게 복원하는 작업
  - 스캔 이미지에는 하프톤 패턴, 뒷면 비침 효과, 색상 바램 등의 특수&다양한 화질 열화 요소 존재
  - 이들을 효과적으로 제거하여 깨끗한 원본 이미지를 만드는 것이 대회의 목적



디스캐닝 (Descanning) 소개

- 디스캐닝 Task 및 데이터셋 소개
  - 데이터셋은 총 **10,000장**의 훈련용 이미지와 **500장**의 테스트용 이미지로 구성됨
  - 각 이미지들은 최근 수년간 출간된 잡지를 직접 수집 후 총 4대의 스캐너로 수동 스캔한 결과물들
  - 원본 데이터셋은 1024x1024 크기의 총 18,360장의 이미지이며. 이중 일부를 추출 후 512x512 크기로 랜덤 크롭하여 대회 데이터셋으로 제공함.



(a) 1st Scanned Image (b) External Noise (c) Internal Noise (d) Bleed -through effect (e) 2nd Scanned Image (f) Texture Distortion (g) Halftone Pattern (h) Color Transition

디스캐닝 (Descanning) 소개

Original

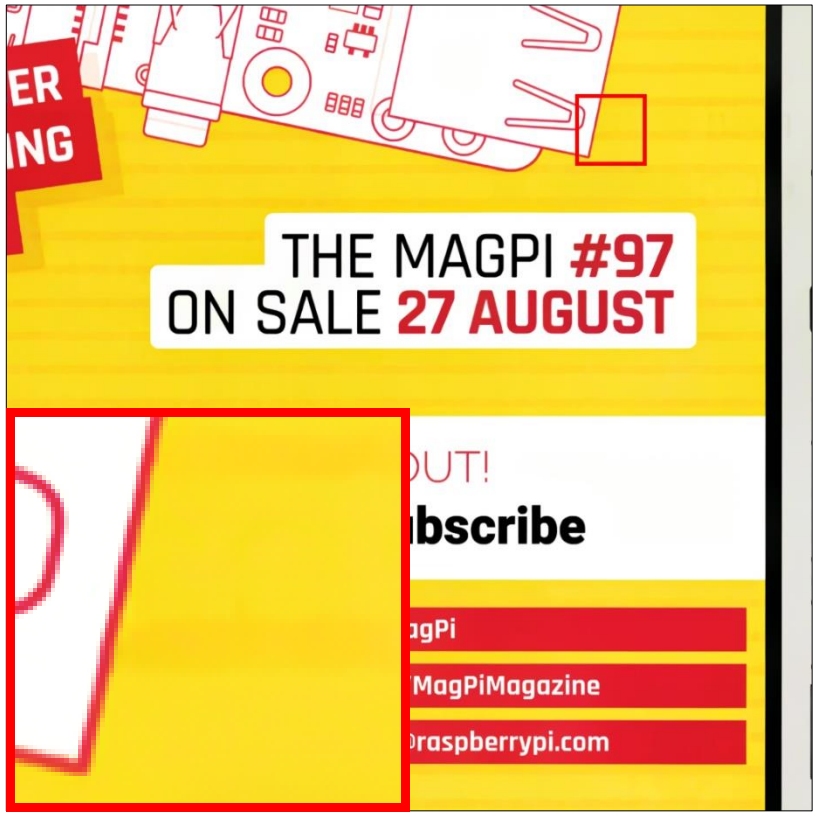
Scanned

Descanned

디스캐닝 (Descanning) 소개

Original

Scanned



Descanned

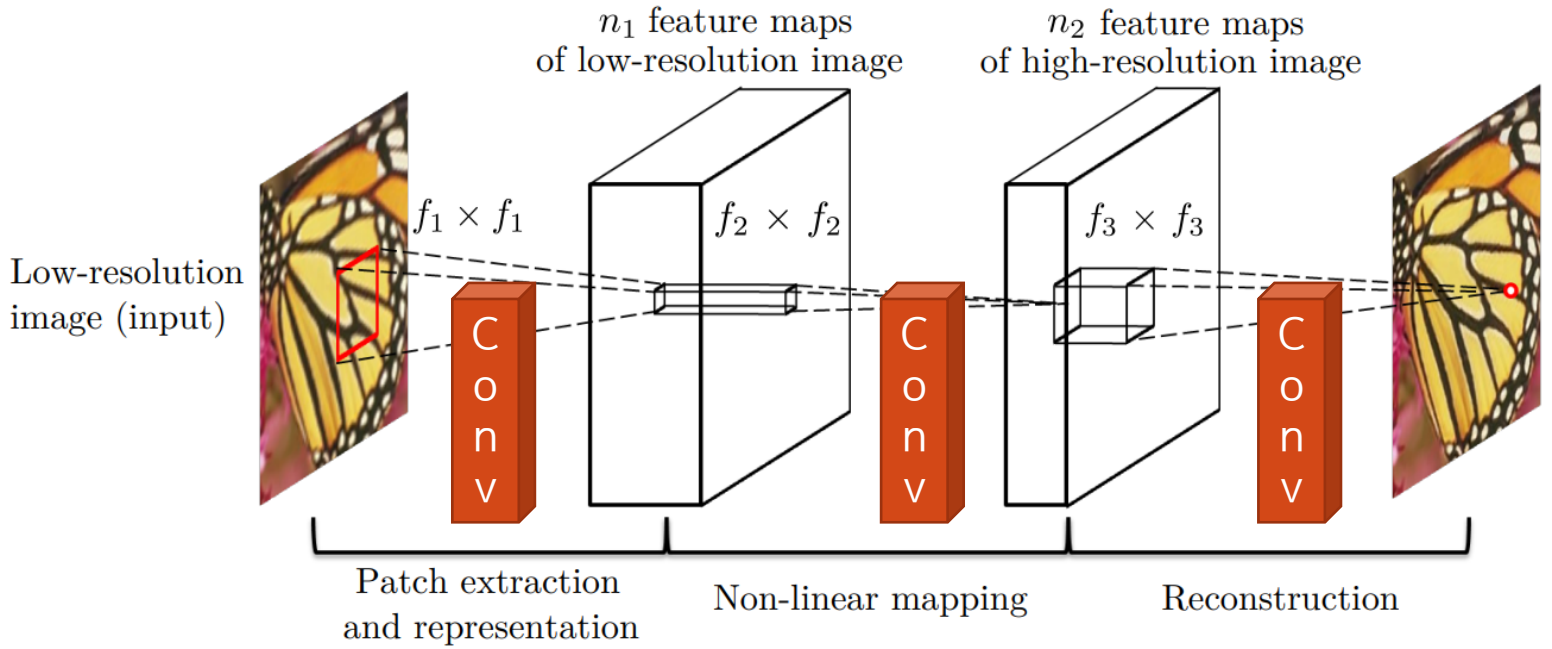
MLVC

디스캐닝 (Descanning) 소개

- 왜 디스캐닝인가?
  - ICT 기술의 발달과 보급 → 아날로그 매체를 디지털 형태로 변환 및 보존하는 작업이 일반화 됨.
  - 그러나 아날로그 매체의 정보는 인쇄, 보존, 스캔 과정에서 지속적으로 **열화됨.**
  - ∴ 현대에선 스캔 이미지에 존재하는 열화들을 효과적으로 제거하여 원본에 가깝게 복원하는 작업이 필수적이며, 이를 **디스캐닝(Descanning)**으로 명명함.
  - 스캔된 문자 정보 → OCR 등을 통해 원본의 손실을 최소화 가능.
  - But 스캔된 이미지 정보 → 이미지로부터 원본을 온전히 복원하는 방법이 현재로선 전무.
  - **이번 Competition을 통해 제안 문제의 중요성을 깨닫고, 효과적인 방법론을 탐색하는 것이 목표**

이미지 복원 모델 소개

- 초해상화 (Super-Resolution): SRCNN
  - 초해상화를 위한 최초의 CNN 기반 모델 → 컨볼루션 (Convolution), ReLU 활성화 함수로 구성
  - 전통 영상처리 기법들을 활용한 초해상화 방법이 CNN으로 대체될 수 있다는 것을 증명함
  - 훈련: LR 이미지를 전통 영상처리 기법으로 HR 이미지의 크기와 같게 한 후, 모델 통과



Y: LR 이미지

X: HR 이미지

F(Y): 모델에 의해 예측된 HR 이미지

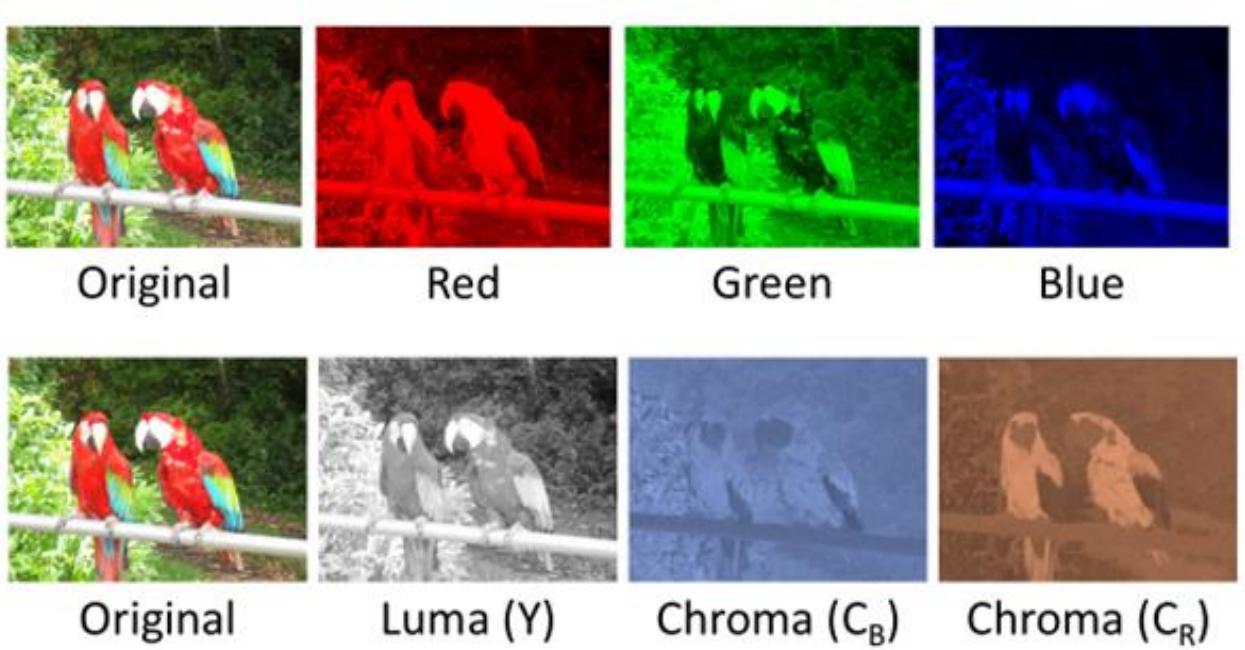
$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i;\Theta) - \mathbf{X}_i||^2$$

[1] Dong *et al.*, "Image Super-Resolution Using Deep Convolutional Networks", ECCV, 2014

이미지 복원 모델 소개

- 초해상화 (Super-Resolution): SRCNN - **Code**
  - 이미지를 YCbCr로 변환 → Y 채널 이미지 인풋 → Y 채널 이미지 아웃풋
  - 기존 Cb, Cr 채널은 전통 영상처리 기법으로 초해상화한 후 Y 채널과 병합

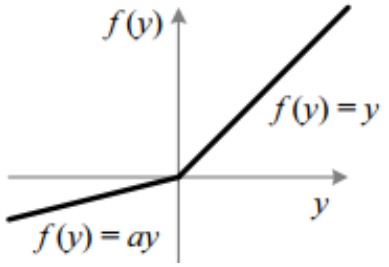
```
class SRCNN(nn.Module):
    def __init__(self):
        super(SRCNN, self).__init__()

        self.layer1 = nn.Sequential(
                nn.Conv2d(1, 64, kernel_size = 9, stride = 1, padding = 4),
                nn.ReLU())

        self.layer2 = nn.Sequential(
                nn.Conv2d(64, 32, kernel_size = 1, stride = 1, padding = 0),
                nn.ReLU())

        self.layer3 = nn.Sequential(
                nn.Conv2d(32, 1, kernel_size = 5, stride = 1, padding = 2))

    def forward(self, x):

        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)

        return x
```



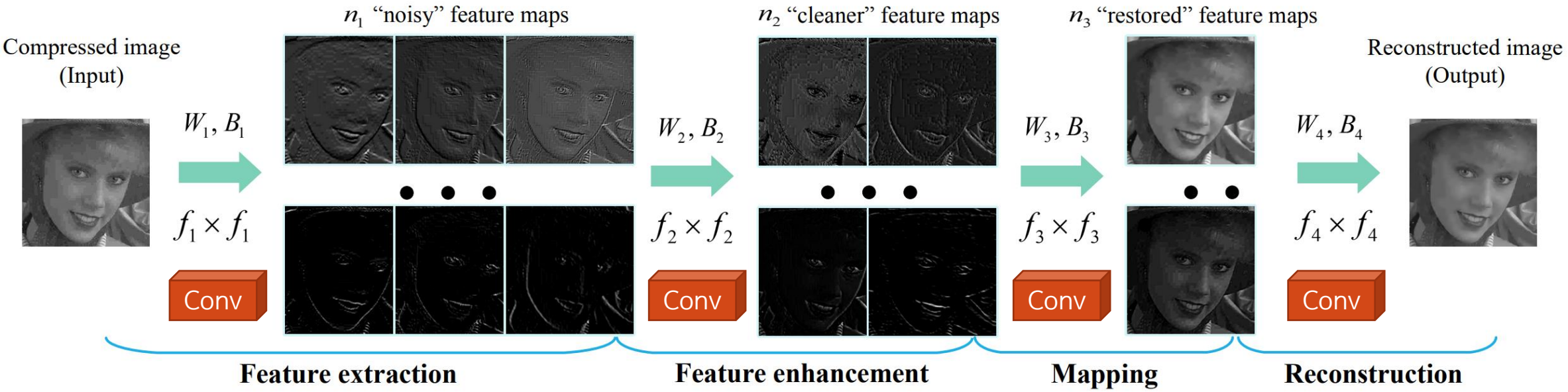
밝기 성분

[1] Dong *et al.*, "Image Super-Resolution Using Deep Convolutional Networks", ECCV, 2014

이미지 복원 모델 소개



- JPEG 압축 열화 제거 (Deblocking): ARCNN
  - JPEG 압축 열화 제거를 위한 최초의 CNN 기반 모델 → 컨볼루션, PReLU 활성화 함수로 구성
  - SRCNN과 마찬가지로, 전통 영상처리 기법들을 활용한 JPEG 압축 열화 제거 방법이 CNN으로 대체될 수 있다는 것을 증명함



[1] Yu *et al.*, "Deep Convolution Networks for Compression Artifacts Reduction", ICCV, 2015

이미지 복원 모델 소개

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i;\Theta) - \mathbf{X}_i||^2$$

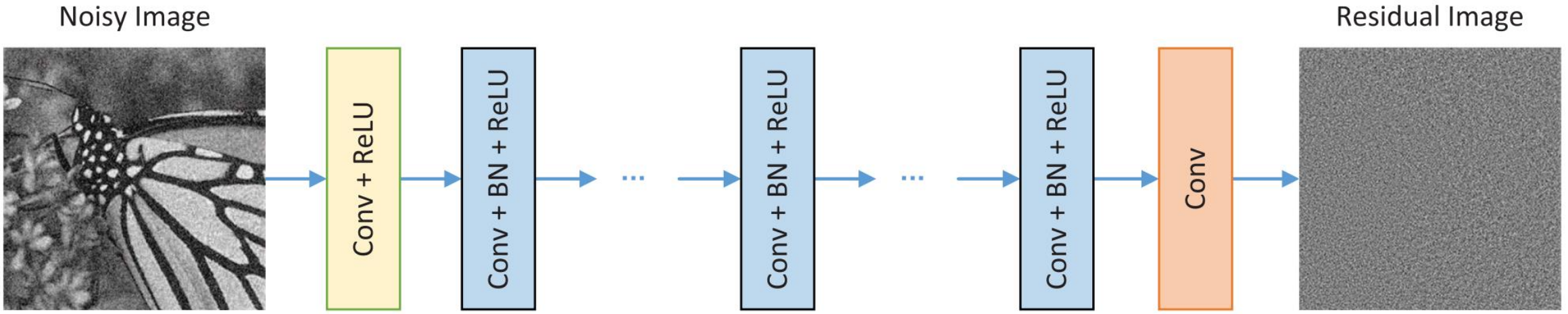
- JPEG 압축 열화 제거 (Deblocking): ARCNN – **Code**
  - SRCNN과 조금 다르게, 컨볼루션을 하나 더 추가하고, YCbCr 3채널 이미지가 인풋 및 아웃풋
  - 원본 이미지에 Image Quality Factor을 이용해 JPEG 압축 이미지 생성 후 훈련에 활용

```
class ARCNN(nn.Module):
    def __init__(self):
        super(ARCNN, self).__init__()
        self.base = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=9, padding=4),
            nn.PReLU(),
            nn.Conv2d(64, 32, kernel_size=7, padding=3),
            nn.PReLU(),
            nn.Conv2d(32, 16, kernel_size=1),
            nn.PReLU()
        )
        self.last = nn.Conv2d(16, 3, kernel_size=5, padding=2)

    def forward(self, x):
        x = self.base(x)
        x = self.last(x)
        return x
```

[1] Yu *et al.*, “Deep Convolution Networks for Compression Artifacts Reduction”, ICCV, 2015

이미지 복원 모델 소개

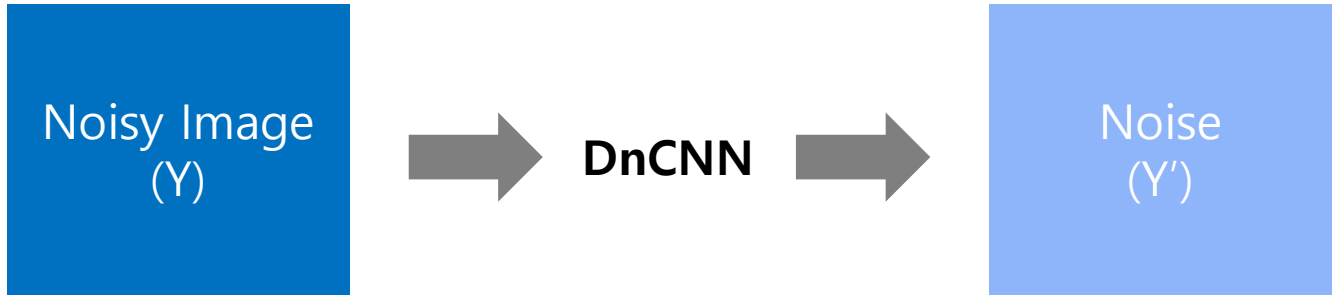
- 노이즈 제거 (Denoising): DnCNN
  - 노이즈 제거를 위한 최초의 CNN 기반 모델
  - 컨볼루션, 배치 정규화 (Batch Normalization), ReLU 활성화 함수로 구성
  - 훈련: 원본 이미지에 노이즈를 주입하여 노이즈 이미지 생성 후, CNN을 통과시켜 노이즈를 예측
  - 추론: 모델을 통해 예측된 노이즈를 실제 노이즈 이미지에서 빼서 원본 이미지를 예측



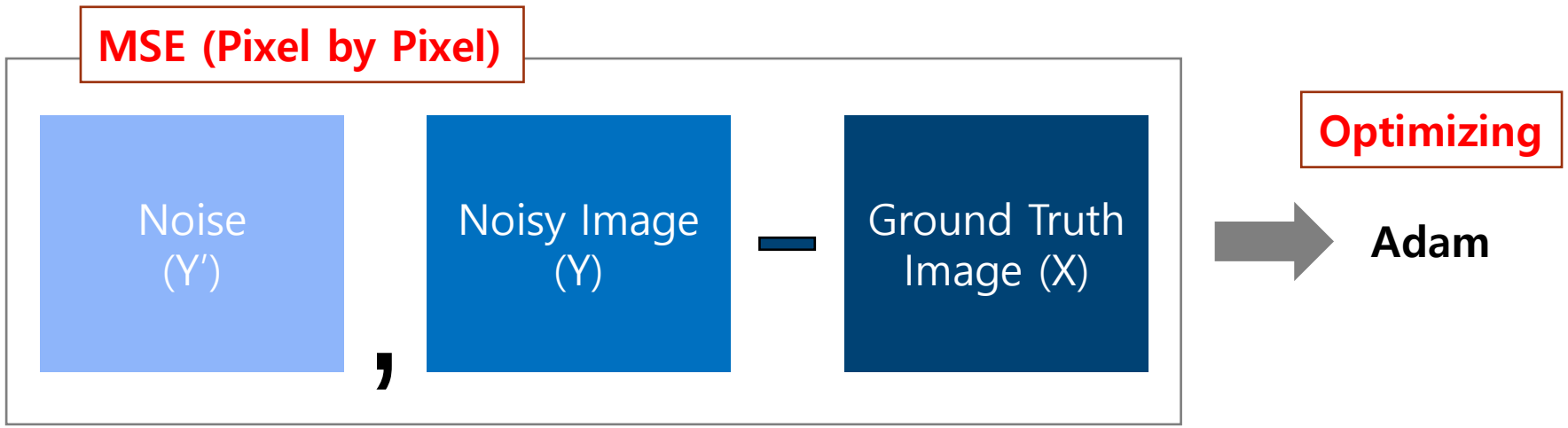
[1] Zhang *et al.*, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising", TIP, 2017

이미지 복원 모델 소개

- 노이즈 제거 (Denoising): DnCNN – **Residual Learning**



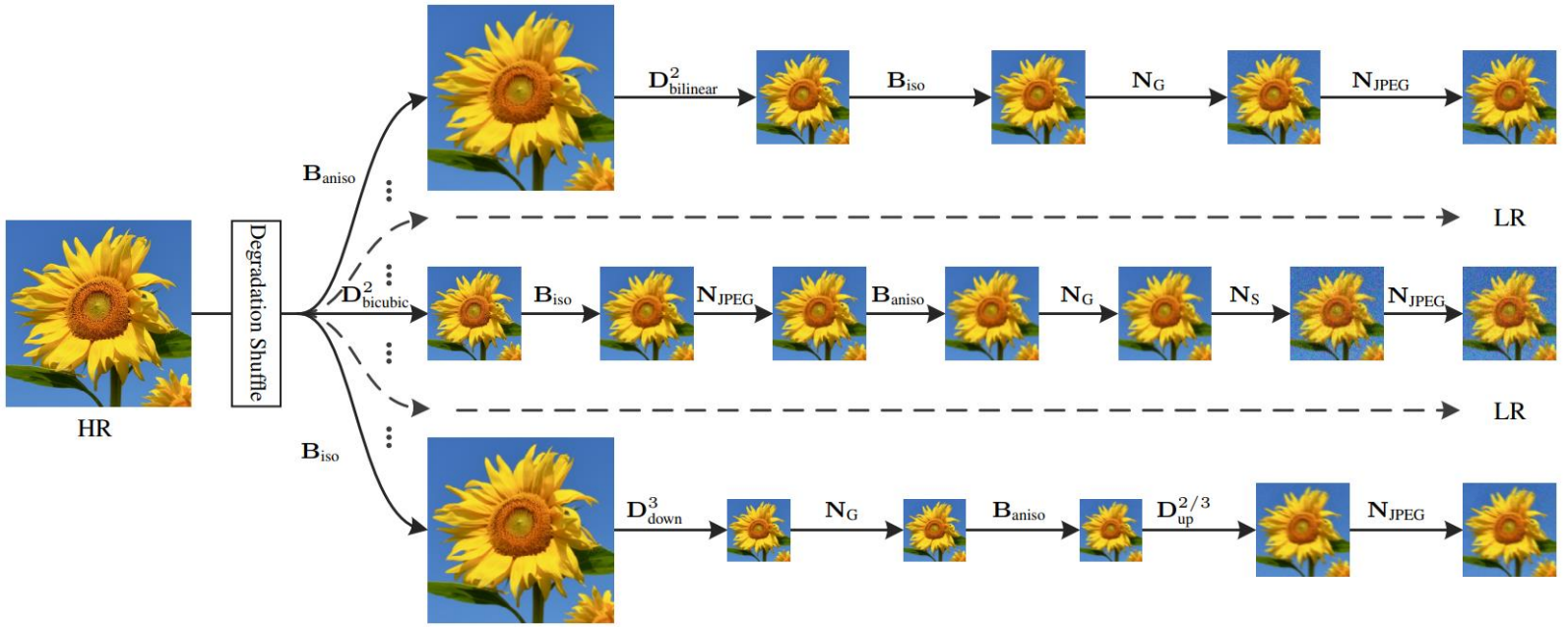
$$\ell(\Theta) = \frac{1}{2N}\sum_{i=1}^{N} \|\mathcal{R}(\mathbf{y}_i;\Theta) - (\mathbf{y}_i - \mathbf{x}_i)\|_F^2$$



[1] Zhang *et al.*, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising", TIP, 2017

학습 전략 소개

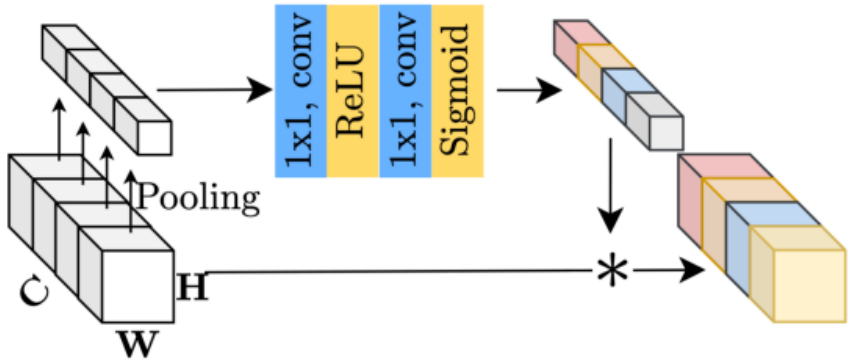
- 데이터 관점: 데이터 증강 (Data Augmentation)
  - 이미지 복원에서도 이미지 분류와 마찬가지로, 기본적인 데이터 증강 기법이 사용됨
  - 랜덤 크롭, 90도 단위 회전, 좌우 반전 등등
  - 최신 논문에선 이미지에 다양한 열화 요소들을 추가하는 데이터 증강 기법이 소개됨

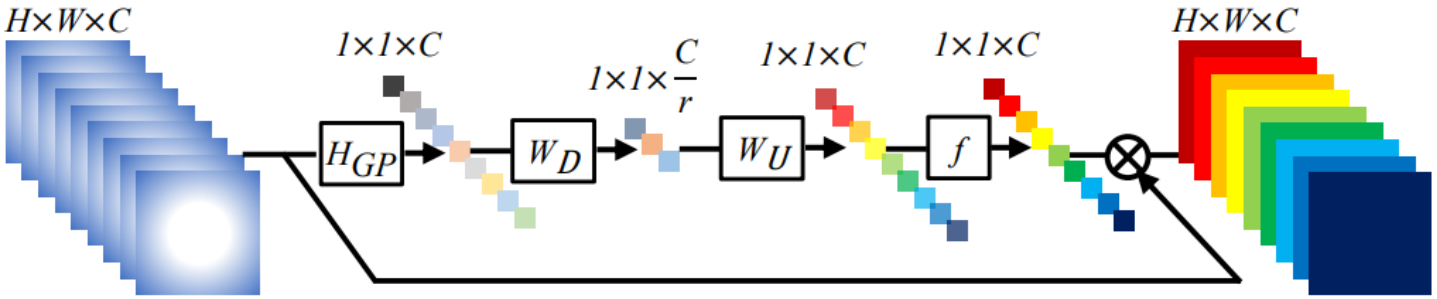


[1] Zhang *et al.*, "Designing a Practical Degradation Model for Deep Blind Image Super-Resolution", ICCV, 2021

학습 전략 소개

- 모델 관점: 어텐션 구조 (Attention)
  - 이미지의 채널 정보와 공간 정보를 한데 끌어 모아 다음 레이어에 주입시키는 방법
  - 각 채널을 나타내는 feature map이 공간적 관점에서 얼마나 중요한지를 알려줌
  - 입력 이미지의 정보를 최대한으로 활용해야하는 이미지 복원 문제에 적합하여 널리 사용됨

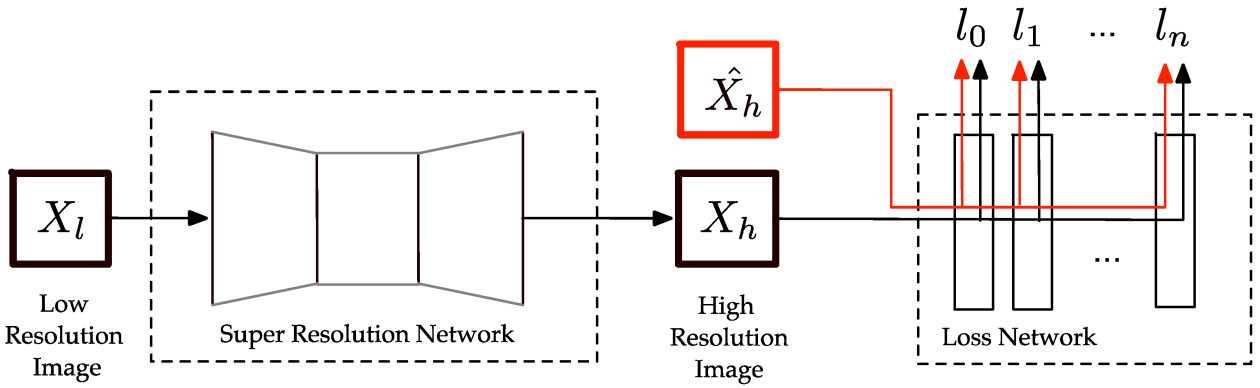




[1] Zhang *et al.*, "Image Super-Resolution Using Very Deep Residual Channel Attention Networks", ECCV, 2018
[2] Chen *et al.*, "Simple Baselines for Image Restoration", ECCV, 2022

학습 전략 소개

- 손실 함수 관점: 지각 손실 함수 (Perceptual Loss)
  - MSE : 픽셀들 각각을 정교하게 비교하려 함 → 픽셀들 평균이 가장 최적값 됨 → Blur 현상 발생
  - Pre-trained된 VGGNet을 이용해서 feature map에서의 L2 loss를 계산하는 방법
    - VGGNet 내부 i번째 max pooling layer 전의 j번째 convolution에 의해 얻어진 feature map
  - 픽셀 각각의 값이 아닌 Perceptual Similarity에 집중함 → 디테일을 더 잘 살릴 수 있게 됨
  - L1 loss와 함께 많이 활용함 (ex. $L_{total} = L_1 + 0.1 * L_{VGG}$)



$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

[1] Ledig *et al.*, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", CVPR, 2017

노이즈 제거 모델 코딩 실습: DnCNN

- 구글 코랩 실행 (런타임 GPU로 설정)
- ecampus 오픈

노이즈 제거 모델 코딩 실습: DnCNN

- 필요 패키지 불러오기

```
import random
import numpy as np
import cv2
import os
import torch
import torch.utils.data as data
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision.transforms import ToTensor, Normalize, Compose
from os.path import join
from os import listdir
from torchsummary import summary
import time
import zipfile
```

MLVC

노이즈 제거 모델 코딩 실습: DnCNN

- 데이터셋 다운로드 (ecampus 코드 복사+붙여넣기)

```
!gdown https://drive.google.com/uc?id=16zAeGDmqbAvn7Iy8V-mBylKx6rG-wgLD
!gdown https://drive.google.com/uc?id=1cqxSVFxfonx5qKVIdfBy0q7uYdNZY8ea
!gdown https://drive.google.com/uc?id=1zmfrXzT9InLg7NIQ-hXekZIyX9aGNNqj

os.makedirs('/content/dataset/train/clean')
os.makedirs('/content/dataset/train/scan')
os.makedirs('/content/dataset/test/scan')

train_clean_zip = zipfile.ZipFile('/content/train_clean.zip')
train_clean_zip.extractall('/content/dataset/train/clean')
train_clean_zip.close()

train_scan_zip = zipfile.ZipFile('/content/train_scan.zip')
train_scan_zip.extractall('/content/dataset/train/scan')
train_scan_zip.close()

test_scan_zip = zipfile.ZipFile('/content/test_scan.zip')
test_scan_zip.extractall('/content/dataset/test/scan')
test_scan_zip.close()
```

MLVC

노이즈 제거 모델 코딩 실습: DnCNN

- 필요한 기초 작업하기

```python
# 랜덤 시드 고정
np.random.seed(42)

# 시작 시간 기록
start_time = time.time()

# 이미지 로드 함수 정의
def load_img(filepath):
    img = cv2.imread(filepath)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    return img
```

노이즈 제거 모델 코딩 실습: DnCNN

- DnCNN 모델 정의

```python
# DnCNN 모델 정의
class DnCNN(nn.Module):
    def __init__(self, num_layers=17, num_channels=64):
        super(DnCNN, self).__init__()
        layers = [nn.Conv2d(3, num_channels, kernel_size=3, padding=1), nn.ReLU(inplace=True)]
        for _ in range(num_layers - 2):
            layers.append(nn.Conv2d(num_channels, num_channels, kernel_size=3, padding=1))
            layers.append(nn.BatchNorm2d(num_channels))
            layers.append(nn.ReLU(inplace=True))
        layers.append(nn.Conv2d(num_channels, 3, kernel_size=3, padding=1))
        self.dncnn = nn.Sequential(*layers)

    def forward(self, x):
        out = self.dncnn(x)
        return out
```

MLVC

노이즈 제거 모델 코딩 실습: DnCNN

- 커스텀 데이터셋 정의

```python
# 커스텀 데이터셋 클래스 정의
class CustomDataset(data.Dataset):
    def __init__(self, noisy_image_paths, clean_image_paths, patch_size = 128, transform=None):
        self.clean_image_paths = [join(clean_image_paths, x) for x in listdir(clean_image_paths)]
        self.noisy_image_paths = [join(noisy_image_paths, x) for x in listdir(noisy_image_paths)]
        self.transform = transform
        self.patch_size = patch_size

    def __len__(self):
        return len(self.noisy_image_paths)
```

MLVC

노이즈 제거 모델 코딩 실습: DnCNN

- 커스텀 데이터셋 정의

```python
def __getitem__(self, index):
    # 이미지 불러오기
    noisy_image = load_img(self.noisy_image_paths[index])
    clean_image = load_img(self.clean_image_paths[index])

    H, W, _ = clean_image.shape

    # 이미지 랜덤 크롭
    rnd_h = random.randint(0, max(0, H - self.patch_size))
    rnd_w = random.randint(0, max(0, W - self.patch_size))
    noisy_image = noisy_image[rnd_h:rnd_h + self.patch_size, rnd_w:rnd_w + self.patch_size, :]
    clean_image = clean_image[rnd_h:rnd_h + self.patch_size, rnd_w:rnd_w + self.patch_size, :]

    # transform 적용
    if self.transform:
        noisy_image = self.transform(noisy_image)
        clean_image = self.transform(clean_image)

    return noisy_image, clean_image
```

노이즈 제거 모델 코딩 실습: DnCNN

- 하이퍼 파라미터 및 데이터셋 관련 정의

```
# 하이퍼파라미터 설정
num_epochs = 10
batch_size = 64
learning_rate = 0.001

# 데이터셋 경로
noisy_image_paths = '/home/hallasan/nvme0n1/MLIP/dataset/train/scan'
clean_image_paths = '/home/hallasan/nvme0n1/MLIP/dataset/train/clean'

# 데이터셋 로드 및 전처리
train_transform = Compose([
    ToTensor(),
    Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])

# 커스텀 데이터셋 인스턴스 생성
train_dataset = CustomDataset(noisy_image_paths, clean_image_paths, transform=train_transform)

# 데이터 로더 설정
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
```

MLVC

노이즈 제거 모델 코딩 실습: DnCNN

- GPU 관련 설정 및 손실함수, 옵티마이저 정의

```
# GPU 사용 여부 확인
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# DnCNN 모델 인스턴스 생성 및 GPU로 이동
model = DnCNN().to(device)
print(summary(model, (3, 128, 128)))

# 손실 함수와 최적화 알고리즘 설정
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)
```

노이즈 제거 모델 코딩 실습: DnCNN

- 모델 학습

```python
# 모델 학습
model.train()
best_loss = 9999.0
for epoch in range(num_epochs):
    running_loss = 0.0
    for noisy_images, clean_images in train_loader:
        noisy_images = noisy_images.to(device)
        clean_images = clean_images.to(device)
        optimizer.zero_grad()
        outputs = model(noisy_images)
        loss = criterion(outputs, noisy_images-clean_images)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * noisy_images.size(0)
    epoch_loss = running_loss / len(train_dataset)
    print(f'Epoch {epoch+1}/{num_epochs}, Loss: {epoch_loss:.4f}')

# 현재 epoch의 loss가 최소 loss보다 작으면 모델 갱신
    if epoch_loss < best_loss:
        best_loss = epoch_loss
        torch.save(model.state_dict(), 'best_dncnn_model.pth')
        print(f"{epoch+1}epoch 모델 저장 완료")
```

MLVC

노이즈 제거 모델 코딩 실습: DnCNN

- 학습 종료 시간 기록

```python
# 종료 시간 기록
end_time = time.time()

# 소요 시간 계산
training_time = end_time - start_time

# 시, 분, 초로 변환
minutes = int(training_time // 60)
seconds = int(training_time % 60)
hours = int(minutes // 60)
minutes = int(minutes % 60)

# 결과 출력
print(f"훈련 소요 시간: {hours}시간 {minutes}분 {seconds}초")
```

MLVC

노이즈 제거 모델 코딩 실습: DnCNN

- 파라미터 수 측정 (학습 패치사이즈로 넣었을 때 1000만개 미만 필수)
- 모델 훈련 과정 출력

```
        Conv2d-39         [-1, 64, 128, 128]          36,928
   BatchNorm2d-40         [-1, 64, 128, 128]             128
          ReLU-41         [-1, 64, 128, 128]               0
        Conv2d-42         [-1, 64, 128, 128]          36,928
   BatchNorm2d-43         [-1, 64, 128, 128]             128
          ReLU-44         [-1, 64, 128, 128]               0
        Conv2d-45         [-1, 64, 128, 128]          36,928
   BatchNorm2d-46         [-1, 64, 128, 128]             128
          ReLU-47         [-1, 64, 128, 128]               0
        Conv2d-48          [-1, 3, 128, 128]           1,731
================================================================
Total params: 559,363
Trainable params: 559,363
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 376.38
Params size (MB): 2.13
Estimated Total Size (MB): 378.70
----------------------------------------------------------------
```

```
Epoch 1/10, Loss: 0.0417
1epoch 모델 저장 완료
Epoch 2/10, Loss: 0.0334
2epoch 모델 저장 완료
Epoch 3/10, Loss: 0.0313
3epoch 모델 저장 완료
Epoch 4/10, Loss: 0.0293
4epoch 모델 저장 완료
Epoch 5/10, Loss: 0.0298
Epoch 6/10, Loss: 0.0287
6epoch 모델 저장 완료
Epoch 7/10, Loss: 0.0282
7epoch 모델 저장 완료
Epoch 8/10, Loss: 0.0263
8epoch 모델 저장 완료
Epoch 9/10, Loss: 0.0270
Epoch 10/10, Loss: 0.0262
10epoch 모델 저장 완료
훈련 소요 시간: 0시간 46분 11초
```

MLVC

노이즈 제거 모델 코딩 실습: DnCNN

- 모델 테스트 (추론) 코드 및 제출용 csv 파일 생성 코드: ecampus 참고

```
# 이미지 denoising 및 저장
for noisy_image, noisy_image_path in noisy_loader:
    noisy_image = noisy_image.to(device)
    noise = model(noisy_image)

    denoised_image = noisy_image - noise

    # denoised_image를 CPU로 이동하여 이미지 저장
    denoised_image = denoised_image.cpu().squeeze(0)
    denoised_image = torch.clamp(denoised_image, 0, 1)  # 이미지 값을 0과 1 사이로 클램핑
    denoised_image = transforms.ToPILImage()(denoised_image)

    # Save denoised image
    output_filename = noisy_image_path[0]
    denoised_filename = output_path + '/' + output_filename.split('/')[-1][:-4] + '.png'
    denoised_image.save(denoised_filename)

    print(f'Saved denoised image: {denoised_filename}')
```

Q&A

- ecampus 이메일 작성해주세요
- https://docs.google.com/spreadsheets/d/1Voar9aLIgSrV8XBP8Mw8HnucUHb06-iLRDALDhH4sCQ/edit#gid=0

Q&A

- Don’t cheat! Have fun! Enjoy yourself!

Thank you