[대학혁신] 2021학년도 1학기 학부생
연구형 인턴쉽 프로그램 연구 결과 보고서

<table>
<tr><td rowspan="3">연구자</td><td>소 속</td><td colspan="3">IT 융합대학 소프트웨어학과(전공) 4학년 2학기</td></tr>
<tr><td>성 명</td><td>윤호운</td><td>학 번</td><td>201735626</td></tr>
<tr><td>휴대폰 번호</td><td>010 - 3353 - 6829</td><td>E-mail</td><td>hannah1258@naver.com</td></tr>
<tr><td rowspan="2">연구개요</td><td>연구제목</td><td colspan="3">Deep learning hint-based image colorization</td></tr>
<tr><td>연구기간</td><td>2021.4- 2021.7.18</td><td>지도교수명</td><td>정용주 교수님</td></tr>
<tr><td>연구내용 및 결과 주요내용</td><td colspan="4">❑ 연구의 배경 및 목적
옛날 사진들은 대부분 흑백이며 화질이 선명하지 못한 경우가 많다. 많은 사람들이 이러한 사진으로 이전 시대의 모습을 추측하기도 하고, 생생하게 보고 싶어 한다. 이를 위해 흑백 이미지의 원래 색상을 추측해서 컬러로 볼 수 있게 해주는 컬러화 작업이 필요하다. 이번 연구에서는 흑백이미지의 컬러화를 위해 다양한 CNN 기반의 딥러닝 모델을 사용하여 컬러 힌트를 기반으로 새로운 컬러이미지를 생성하였다.

❑ 연구내용 및 방법
❑ 연구결과

❑ 기대효과
이번 연구를 통해 다양한 U-Net 구조의 딥러닝 모델을 컬러화 작업에 적용시켜보면서, 기본 Unet 구조를 사용했을 때 보다 자연스럽고 높은 해상도와 고품질의 컬러화 된 이미지 결과를 얻을 수 있었다. 향후에는 더 많은 dataset을 사용하고 다양한 hyperparameter들을 조절하면서 학습을 개선할 것이고, 여기에 다양한 high-level의 feature를 세밀하게 추출하는 발전된 모델을 적용한다면 성능은 더욱 더 향상 될 것으로 기대한다.</td></tr>
</table>

본인은 첨부와 같이 연구 결과 보고서를 제출합니다.

첨 부 : 연구 결과물을 첨부하여 제출

2021년 7월 21일

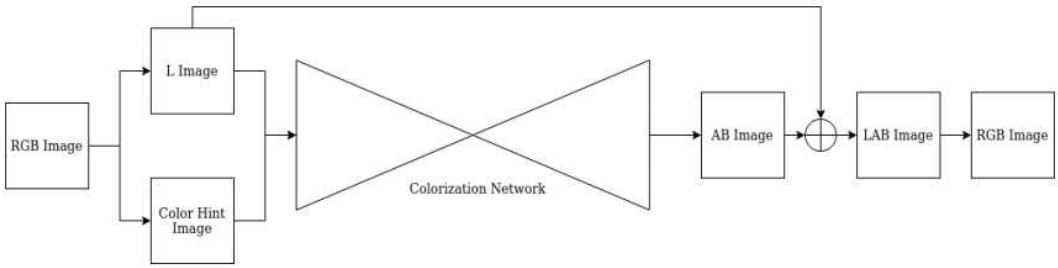
신 청 인 윤호운 (날인/서명)

지도교수 정용주 (날인/서명)

일반대학원장 귀하

❑연구 내용 및 방법

1) Overview of the Task



hint-based colorization은 사용자가 제공하는 컬러 힌트를 기반으로 이미지를 학습하여 전체 컬러이미지를 재구성하는 과정이다. 우선 제공된 RGB 이미지를 LAB 이미지로 변환하는 과정을 거친다. 모델의 입력은 원본 이미지의 'L image' (i.e. grayscale image or intensity image) 와 'color hint image' 이다. 출력은 모델로부터 학습된 'AB image'이다. 입력으로 사용한 'L image'와 해당 'AB image'를 연결하여 최종 컬러 이미지를 생성한다.

Example of hint-based colorization

2) Task Performance Process

(1) Dataset

Training dataset과 Validation dataset은 Place365 와 ImagNet training dataset에서 무작위로 샘플링하였다. 256*256 사이즈의 모든 이미지를 128*128 사이즈로 변경한 뒤 학습을 진행하였고, test 또한 128*128 사이즈의 이미지가 적용되었다. 모든 dataset의 color-hint image는 원본이미지의 1%,3%,5% color를 랜덤으로 추출하여 생성하였다.

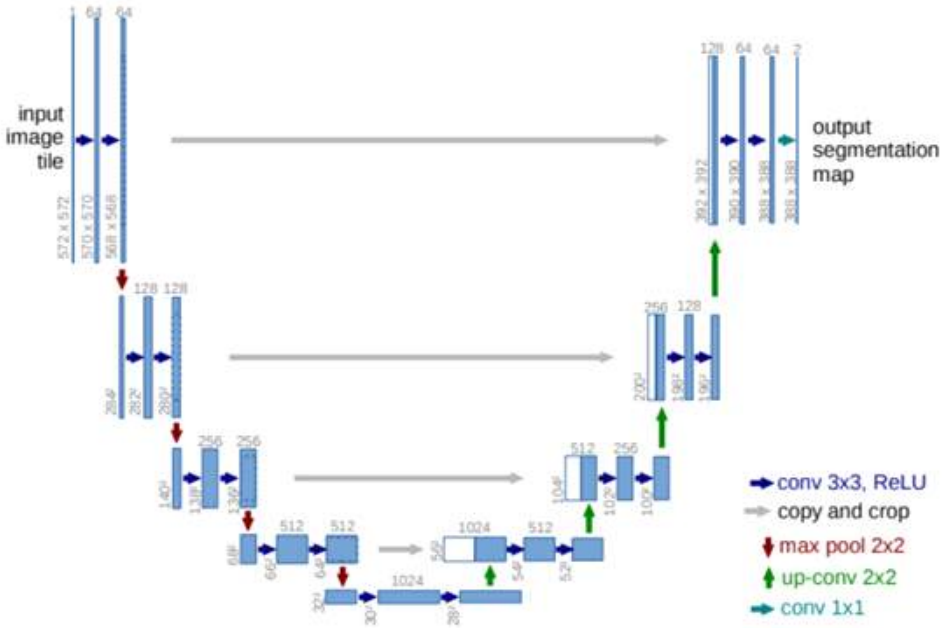
Training Dataset
- Size : 4500
- Batch size : 4
- Input data size : 128*128
- Color hint : [1% , 3% , 5%] randomly from dataloader

Validation Dataset
- Size : 500
- Input data size : 128*128
- Color hint : [1% , 3% , 5%] randomly from dataloader

Test Dataset
- Size : 1000
- Input data size : 128*128
- Color hint : [1% , 3% , 5%] randomly from dataloader

(2) Training and Evaluation

- 1)Basic UNet structure



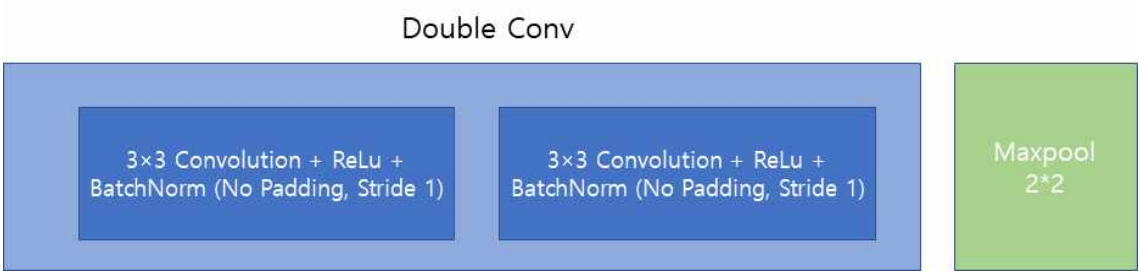
처음엔 기본적인 Unet 구조를 그대로 구현하여 모델을 생성하였다. Unet은 점진적으로 넓은 범위의 이미지 픽셀을 보며 이미지의 전반적인 Context 정보를 얻기 위한 Contracting Path와 의미정보를 픽셀 위치 정보와 결합하여 각 픽셀마다 어떤 객체에 속하는지를 구분하는 정확한 Localization를 위한

Expanding Path가 대칭 형태로 구성되어 있다.

Contracting path

Contracting path에서 아래와 같은 Downsampling 과정을 4번 반복하여 feature map을 생성하였다. Downsampling 할 때 마다 Channel의 수를 2배씩 증가시켰다. 즉 처음 Input Channel 3개를 64개로 증가시키는 부분을 제외하면 채널은 3->64->128->256->512->1024 개로 Downsampling 진행할 때마다 2배씩 증가한다.

1. 3×3 Convolution Layer + ReLu + BatchNorm (Padding 1, Stride 1)
2. 3×3 Convolution Layer + ReLu + BatchNorm (Padding 1, Stride 1)
3. 2×2 Max-polling Layer (Stride 2)

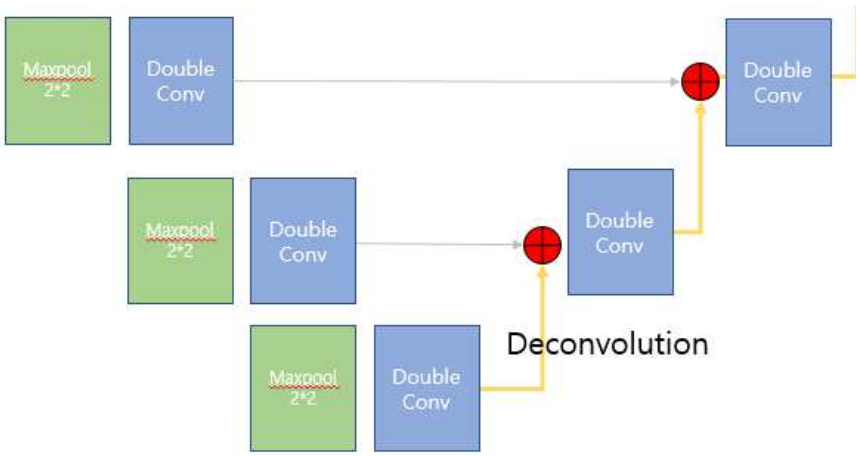


< Downsampling process >

Expanding path

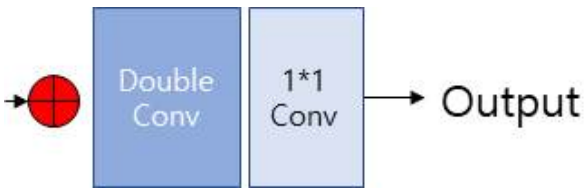
Expanding path에서 아래와 같은 Upsampling 과정을 반복하여 Feature Map을 생성했다.

1. 2×2 Deconvolution layer (Stride 2) #nn.ConvTranspose2d
2. Contracting path에서 동일한 Level의 Feature Map을 추출하여,deconvolution한 Feature Map과 연결. (padding을 적용하여 crop 과정 생략)
3. 3×3 Convolution Layer + ReLu + BatchNorm (Padding 1, Stride 1)
4. 3×3 Convolution Layer + ReLu + BatchNorm (Padding 1, Stride 1)



<Upsampling process>

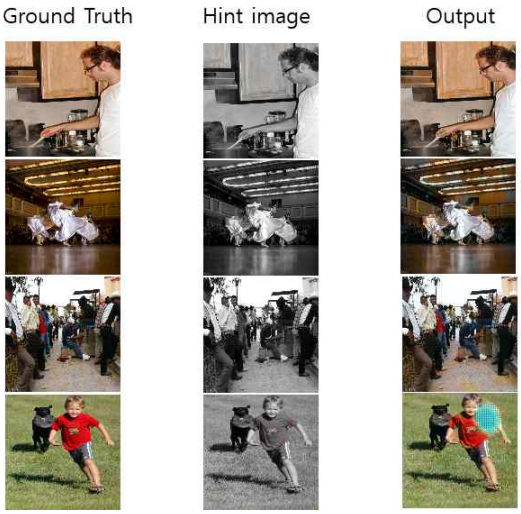
Expanding path는 Skip Connection을 통해 Contracting path 경로에서 생성된 Contextual 정보와 위치정보를 결합하는 역할을 한다. Expanding path의 마지막에 3 크기의 만큼 필터를 갖고 있는 1×1 Convolution Layer를 두어 최종 이미지를 생성하였다.



마지막으로 Hyperparameter는 아래와 같이 적용하여 학습을 진행하였다.

<Hyperparameter>
Learning rate : 1e-3
Epoch : 80
Optimizer : Adam
Loss function : L1Loss
Batch size : 4

Basic Unet Result



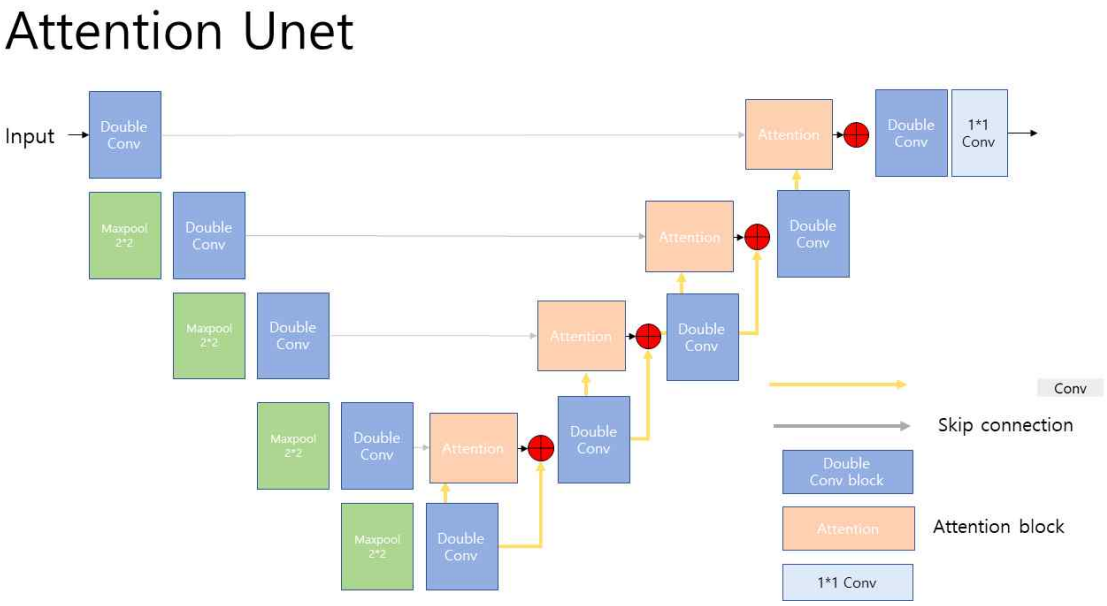
Epoch 40
psnr : 31.221867
ssim : 0.907829

Epoch 53
psnr : 30.674506
ssim : 0.908905

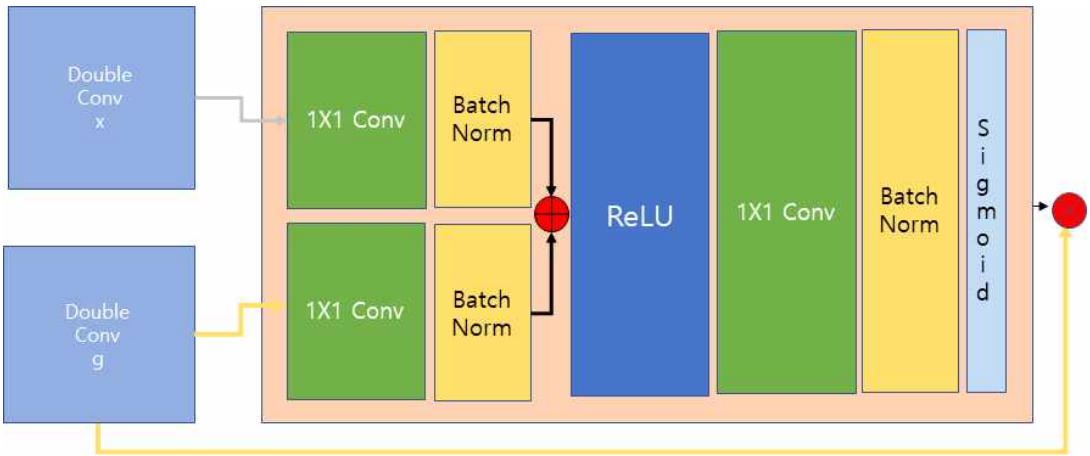
Epoch 49
psnr : 30.031631
ssim : 0.910166

기본 Unet 구조의 test결과는 epoch 40에서 제일 높았지만 PSNR 31db, SSIM 0.90 정도로 좋은 성능은 보이지 못했다.

- 2) Attention UNet



두 번째로 기본 Unet 구조에 attention block을 더해 중요한 부분을 더 집중해서 학습할 수 있는 attention unet 모델을 적용하였다. Attention block을 통해서 앞부분에서 low level의 feature들이 중복적으로 추출되는 것을 막고 관련성이 낮은 영역의 활성화를 억제하도록 했다.



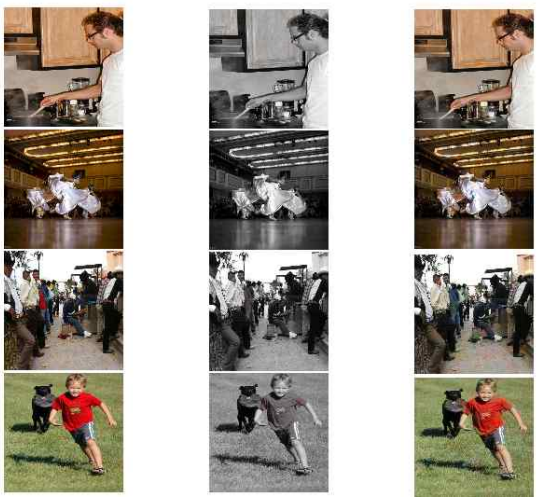
<Attention block>

Hyperparameter는 앞 모델과 동일하게 사용하였다.

<Hyperparameter>
Learning rate : 1e-3
Epoch : 80
Optimizer : Adam
Loss function : L1Loss
Batch size : 4

Attention-UNet Result

Ground Truth | Hint image | Output



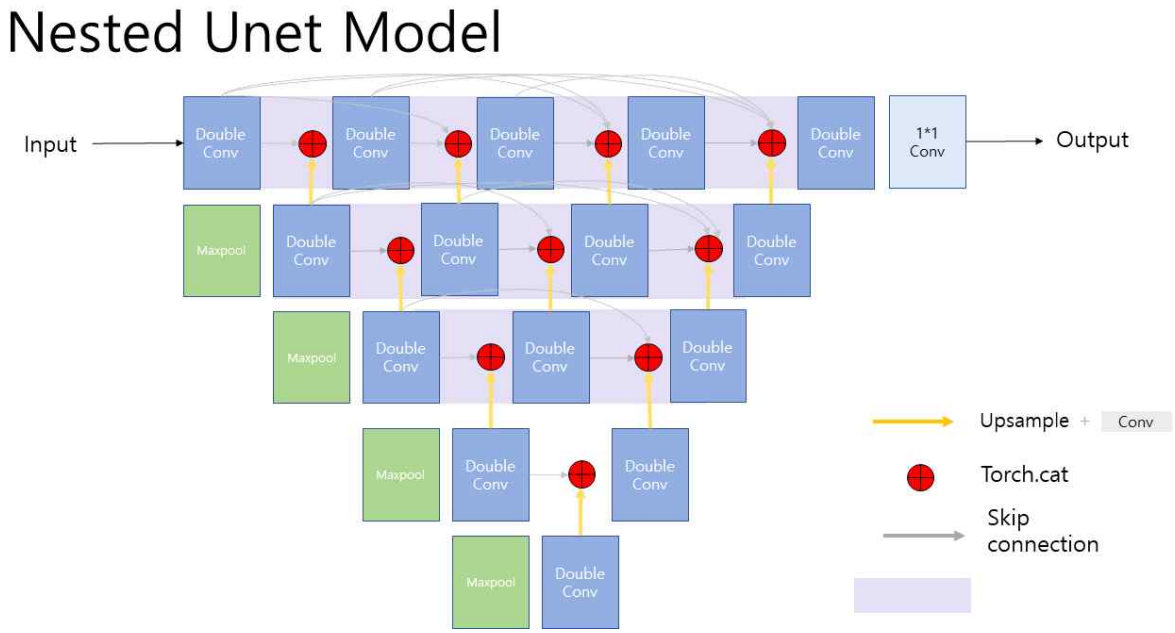
Epoch 74
psnr : 32.706043
ssim : 0.929230

Epoch 71
psnr : 32.820084
ssim : 0.922077

Epoch 58
psnr : 32.699337
ssim : 0.925426

Attention Unet 모델의 test결과는 epoch 74에서 제일 높았고 PSNR 32.7db, SSIM 0.92정도로 기본 Unet 모델보다 2db 정도 높은 정확도의 성능을 보였다.

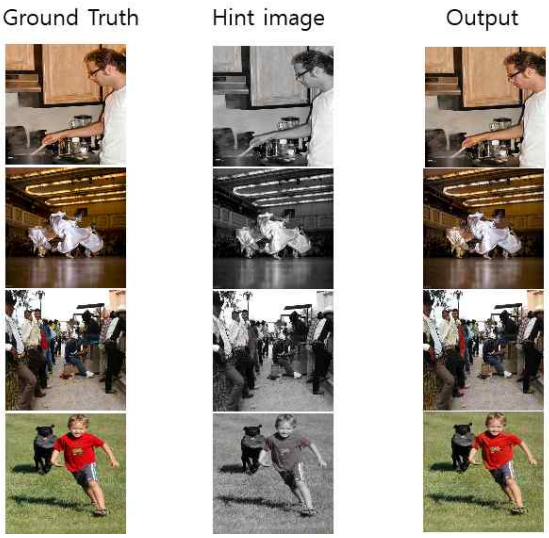
- 3)Nested UNet



<Hyperparameter>
Learning rate : 1e-3
Epoch : 80
Optimizer : Adam
Loss function : L1Loss
Batch size : 4

세번째로 Nested Unet 모델을 적용해보았다. Nested Unet은 DenseNet의 Dense block 아이디어를 사용하여 Unet을 향상시킨 모델인데, skip pathways에 convolution 층을 두면서 기울기를 완화하고 모든 feature map을 연결해 encoder과 decoder 사이의 semantic gap을 연결하도록 했다. Hyperparameter는 이전과 동일하게 두어 학습해보았고 성능은 63 epoch일 때 psnr 34db 정도로 세 모델 중 가장 우수하였다.

Nested UNet Result



Epoch 63
psnr : 34.38482
ssim : 0.941603

Epoch 59
psnr : 34.148814
ssim : 0.942044

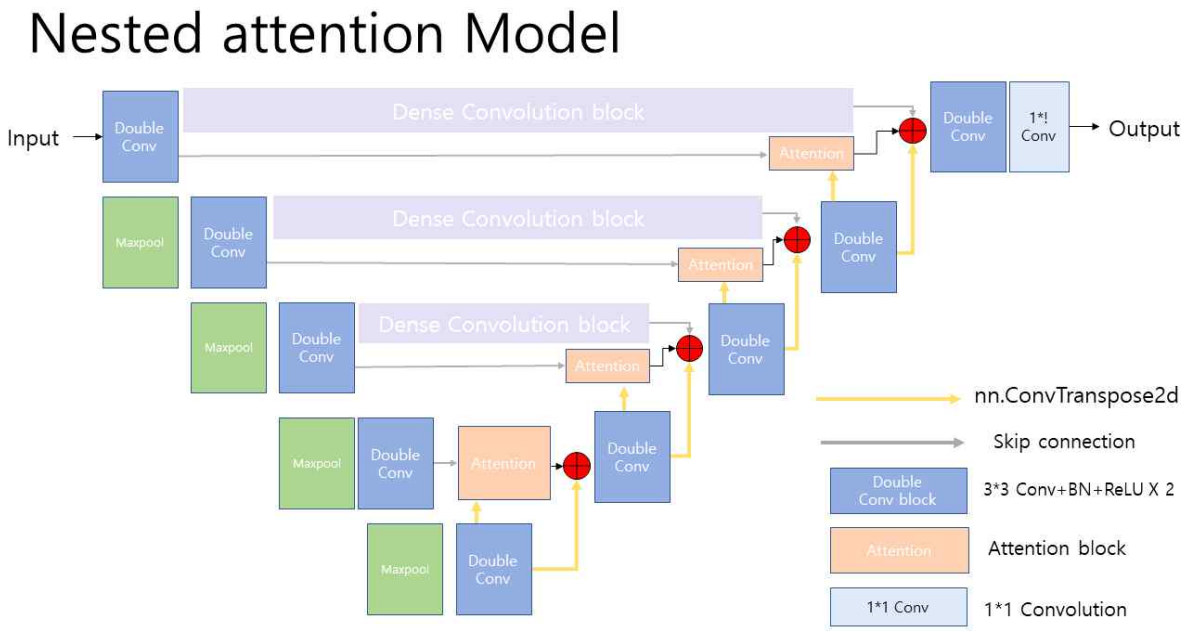
Epoch 46
psnr : 34.118881
ssim : 0.939911

따라서 그다음으로는 Basic Unet, Attention Unet, Nested Unet 세가지 모델 중 가장 성능이 좋았던 Nested Unet을 기반으로 여러 실험을 진행하였다. Dense convolution block연산 중 3*3 convolution + ReLU + batch normalization 으로 이루어진 convolution block의 개수를 조절해보았는데 conv block을 추가할수록 성능이 더 떨어지는 경향을 보였다.

따라서 Expanding path의 Upsampling block 연산이 기존에는 단순히 interpolation 하는 nn.Upsample 함수와 conv block으로 구성되어 있었는데, conv block을 없애고 nn.Upsample을 kernel이 학습하는 nn.Convtranspose2d 함수로 대체하니 모델이 보다 빠르게 최적화하였다.

- Nested Attention UNet

Attention UNet과 Nested UNet에서 아이디어를 얻어 두 모델을 합친 새로운 모델을 구성해보았다. 기존 Nested Unet에 attention block을 추가하여 모델이 Upsampling을 하면서 중요한 정보에 집중할 수 있도록 하였다.

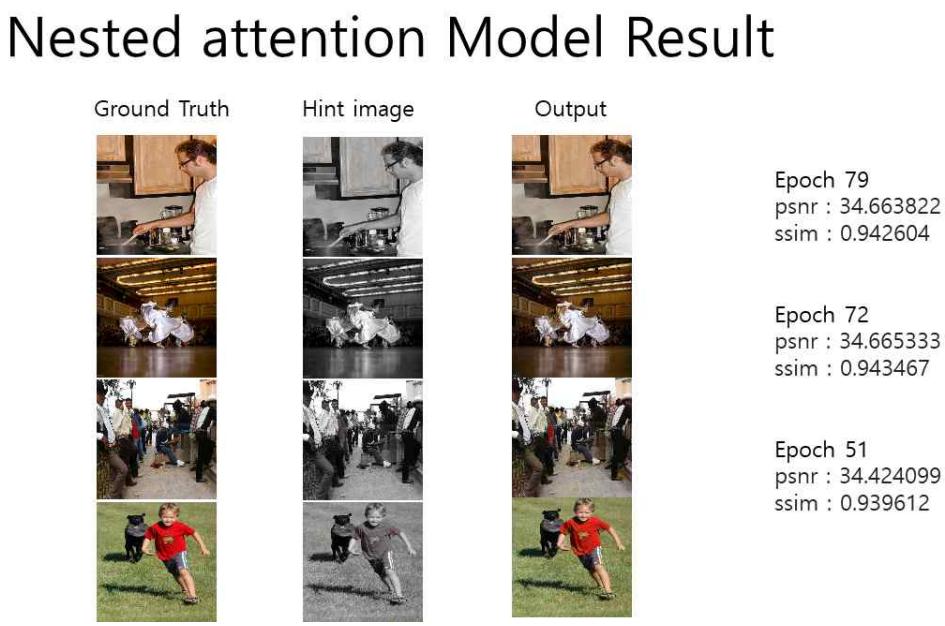


이번 실험에서는 Optuna framework를 이용하여 hyperparameter를 조절하였다. Optuna framework는 사용자가 지정한 parameter들을 최적화하는 방향으로 자동으로 조합하여, 최적의 parameter 조합들을 dataframe 형식으로 반환해준다. 이번 실험에서는 아래와 같이 parameter들을 설정하였다.

<Optuna framework - hyperparameter>
5 epoch
learning rate : 1e-4 ~ 1e-3
optimizer: optim.SGD, optim.Adam ,optim.RMSprop
criterion: MSELoss ,L1Loss

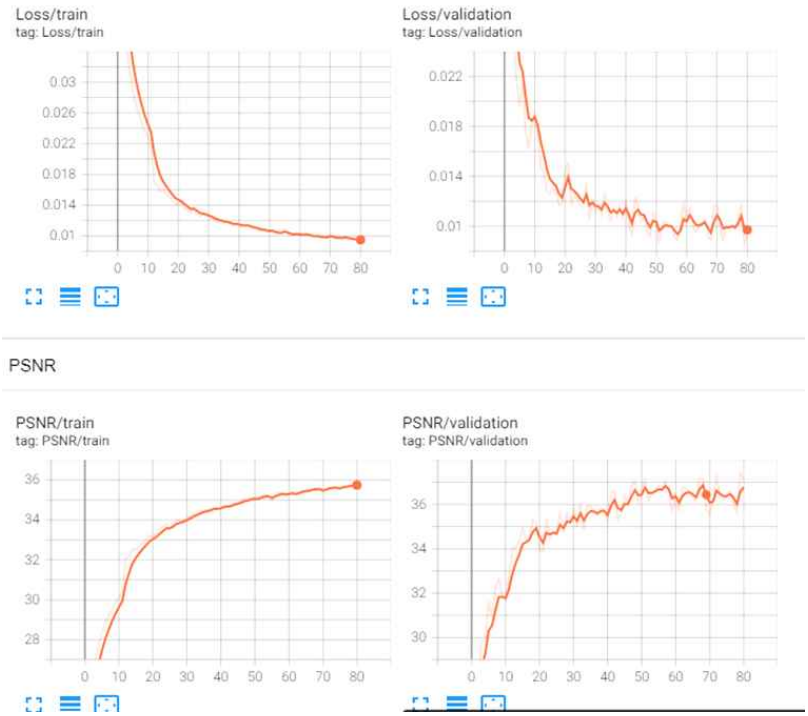
	number	PSNR	params_criterion	params_lr	params_optimizer
5	5	33.006634	L1Loss	0.000508	<class 'torch.optim.rmsprop.RMSprop'>
1	1	32.812950	MSELoss	0.000819	<class 'torch.optim.rmsprop.RMSprop'>
6	6	32.728027	MSELoss	0.000519	<class 'torch.optim.rmsprop.RMSprop'>
2	2	32.670986	L1Loss	0.000301	<class 'torch.optim.sgd.SGD'>
3	3	32.475311	MSELoss	0.000311	<class 'torch.optim.rmsprop.RMSprop'>
14	14	32.415928	MSELoss	0.000801	<class 'torch.optim.sgd.SGD'>
7	7	32.387619	MSELoss	0.000141	<class 'torch.optim.adam.Adam'>
0	0	32.311031	L1Loss	0.000438	<class 'torch.optim.rmsprop.RMSprop'>
13	13	32.299919	MSELoss	0.000959	<class 'torch.optim.rmsprop.RMSprop'>
9	9	32.219170	L1Loss	0.000227	<class 'torch.optim.adam.Adam'>

평가지표를 PSNR로 설정한 뒤 learning rate, optimizer, criterion 각각의 조합을 5 epoch를 학습하여 결과를 dataframe 으로 반환하였다. optuna framework가 수행할 조합의 수는 20으로 설정하였다. 결과는 loss function은 L1 Loss, learning rate는 0.000508, optimizer는 RMSprop 일 때 가장 좋았다. 따라서 해당 hyperparameter들로 커스텀 모델의 80 epoch 학습을 진행하였다.

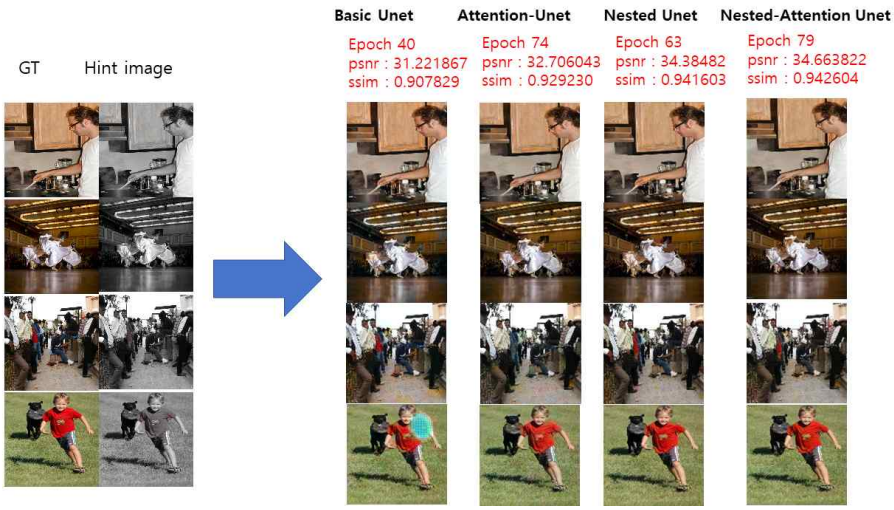


학습결과는 79 epoch에서 PSNR 34.6 db, SSIM 0.94로 가장 높았고, 네 모델 중 가장 높은 성능을 보였다.

또한 80 epoch 중 79 epoch에서 제일 높은 성능을 보인 것에 더해, tensorboard를 통해 학습상황을 시각화하였을 때 loss가 계속 줄어들고, PSNR도 계속 높아지는 것으로 보아 80 epoch 보다 더 많은 학습을 진행하면 성능은 더 좋아질 것으로 보였다.



❑연구 결과



기존의 Basic Unet, Attention Unet, Nested Unet으로 학습을 진행해본 뒤 Nested Unet에 attention block을 더한 Nested-Attention Unet 이라는 새로운 모델을 구성해보았다. 성능을 PSNR과 SSIM으로 측정한 결과, Nested-Attention Unet>Nested Unet>Attention Unet>Basic Unet 순으로 Nested-Attention Unet의 성능이 가장 좋았다.

---

1) https://arxiv.org/abs/1505.04597

2) https://arxiv.org/abs/1804.03999
3) https://arxiv.org/abs/1807.10165