인공지능심화

(deblur project)

전기전자공학부

202110410

조민우

서론

Image Deblurring은 흐릿하게 촬영된 이미지로부터 선명한 이미지를 복원하는 Computer Vision의 중요한 기술이다. 카메라의 손떨림, 피사체의 빠른 움직임 등 다양한 원인에 의해 발생하는 블러(blur)는 이미지의 품질을 크게 저하시킨다. 특히 스마트폰 카메라나 액션 카메라로 촬영된 이미지에서 이러한 문제가 빈번하게 발생하며, 이를 효과적으로 복원하는 것은 실용적으로 매우 중요하다.

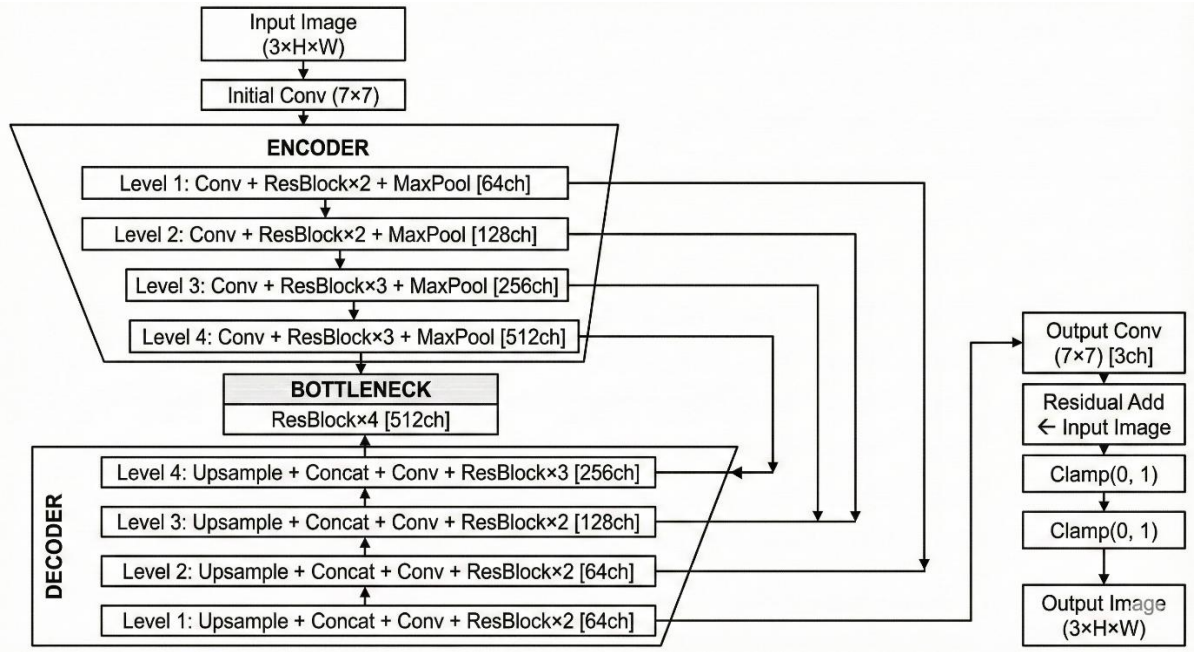
이번 프로젝트는 수업 시간에 학습한 CNN 아키텍처 설계 원리를 바탕으로, 이미지 deblur 문제를 해결하는 네트워크를 직접 설계하고 구현하는 것을 목표로 한다. GoPro Dataset을 활용하여 모델을 학습하고, PSNR(Peak Signal-to-Noise Ratio) 지표를 통해 복원 성능을 평가한다.

해당 프로젝트에서는 local GPU RTX 3080를 활용하여 모델 학습을 진행하였으며, 직접 neural network를 설계함으로써 deblurring 문제에 대한 깊이 있는 이해를 도모하고자 한다. 이를 통해 CNN의 구조 설계, loss 함수 선택, dropout 등 딥러닝 모델 개발의 전반적인 과정을 경험하고, 이론적 지식을 실제 문제 해결에 적용하려 한다.

* 코드는 train.py, infer.py로 나누어져 관리된다.

본론

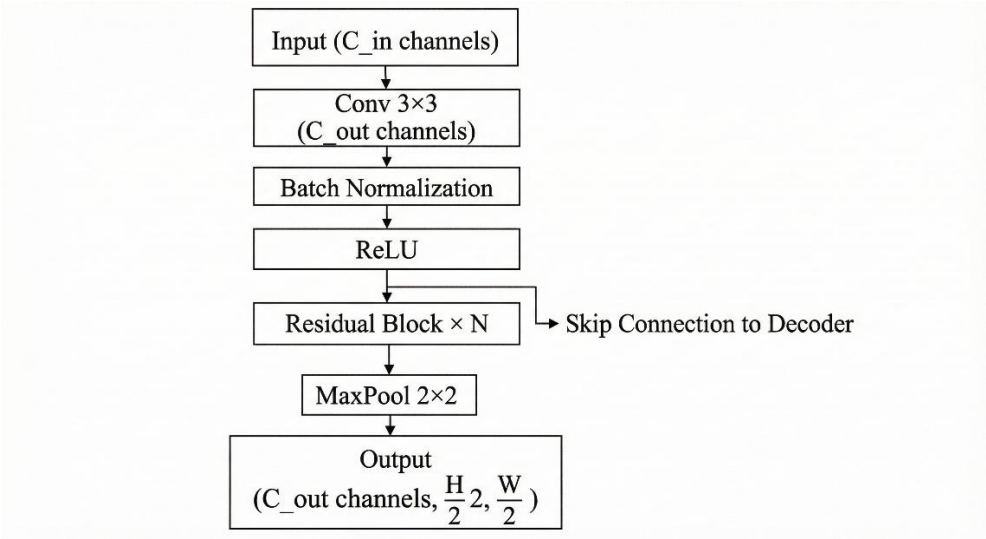
2.1 Network Architecture



<Fig 2.1 Network Architecture>

이번 프로젝트는 ResNet, U-Net 구조를 기반으로 한 image deblurring 모델을 설계하였다. U-Net의 encoder-decoder 구조에 ResNet의 residual block을 결합하여 깊은 네트워크에서도 안정적인 학습이 가능하도록 구성하였다. 모든 구성 요소는 수업에서 배운 CNN, Residual Connection, Batch Normalization, Batch Shuffle, Dropout 등의 기법을 사용하여 구현하였다

2.1.2 Encoder



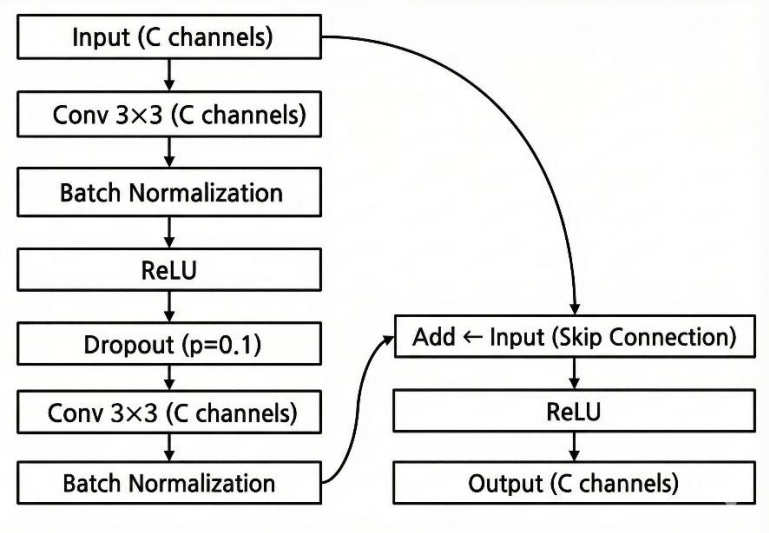
<Fig 2.1.2 Encoder>

모델의 Encoder는 총 4단계(4-stage)로 구성되어 있다. 각 단계는 입력 이미지로부터 갈수록 더 abstract한 feature를 추출한다. 각 Encoder Block은 기본적으로 3×3 Convl Batch Normalization, ReLU 활성화 함수를 포함한 시작한다. 이후 2~3개의 Residual Block이 연속적으로 배치되어, 깊이가 깊어지면서 발생할 수 있는 gradient vanishing problem을 완화하는 역할을 한다. 마지막으로 Max Pooling(2×2)을 적용하여 계산량을 줄임과 동시에 overfitting을 억제한다.

Encoder의 각 단계에선 Level 1에서 64채널로 시작해 Level 2에서 128채널, Level 3에서 256채널, Level 4에서 512채널로 증가한다. 이와 같은 구조는 해상도를 줄이고 채널 수를 늘려 depth를 키우는 일반적인 CNN의 설계를 따른 것으로, 더 풍부하고 추상적인 특징 표현을 학습하는 데 효과적이다.

이러한 구성은 Decoder에서 다양한 스케일의 정보를 결합해 더욱 정교한 복원 및 추론을 가능하게 하는 기반을 제공한다.

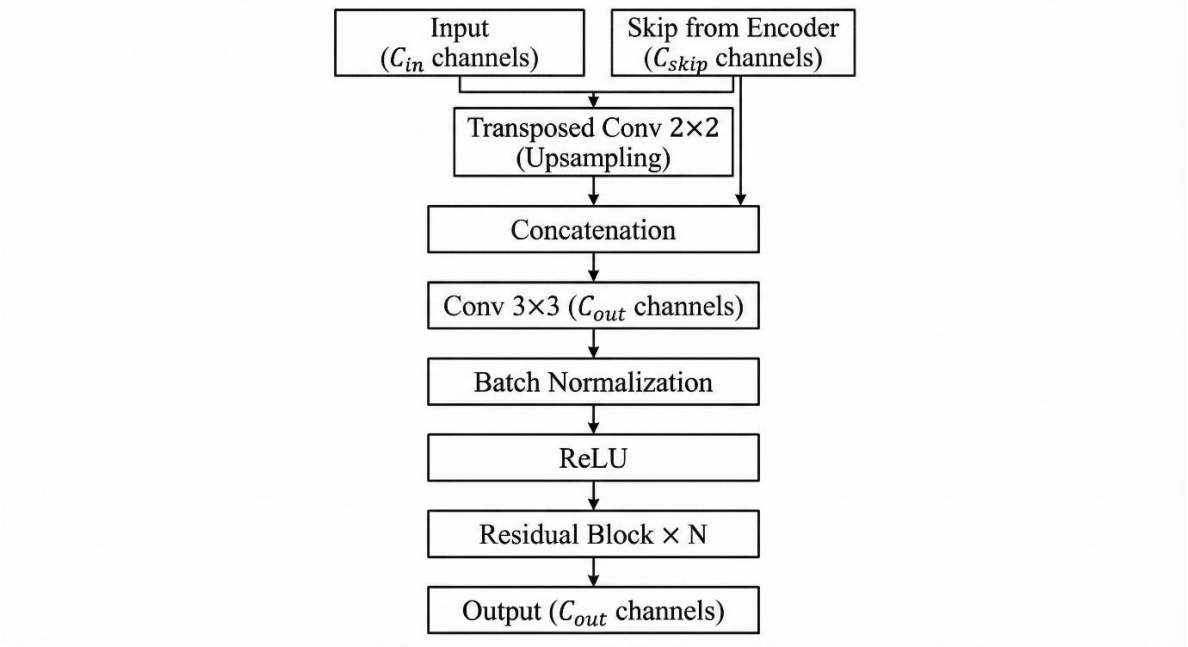
2.1.3 Residual Block(Bottleneck)



<Fig 2.1.3 Residual Block>

Bottleneck은 4개의 Residual Block이 연속적으로 배치된 구조로 구성되며, 전체 네트워크에서 가장 높은 abstract feature를 학습하는 핵심 단계이다. 이 층은 512채널을 사용하며, Encoder를 통해 축약된 feature map을 low resolution에서 처리하여 깊은 abstract feature를 학습한다. Bottleneck의 주요 역할은 Encoder 단계에서 추출된 다양한 스케일의 특징들을 압축하고 통합하는 것이며, 이를 통해 이미지의 구조적 패턴을 전역적으로 파악할 수 있다. 특히 blur와 같은 전역적 특성은 넓은 receptive region에서 학습되기 때문에, Bottleneck은 이러한 전역적 특성을 효과적으로 포착하여 Decoder에 전달하는 중요한 역할을 한다.

2.1.4 Decoder

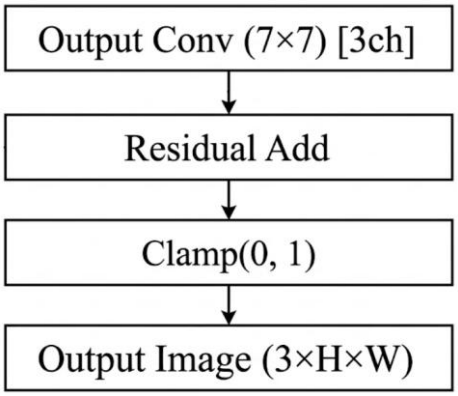


<Fig 2.1.4 Residual Block>

Decoder는 역시 4단계의 구조로 이루어져 있으며, 축소된 이미지를 복원하는 역할을 수행한다. 각 Decoder Block은 Transposed Convolution(2×2)을 통해 해상도를 한 단계 증가시키는 것으로 시작한다. 이후 Encoder의 동일 레벨에서 가져온 특징 맵을 Skip Connection 방식으로 결합하여, 고해상도 복원에 필수적인 low-level 정보들을 보존한다.

Skip Connection으로 결합된 feature map은 3×3 Convol, Batch Normalization, ReLU를 거치고, 이어서 2~3개의 Residual Block이 추가로 적용된다. 해상도는 단계적으로 증가하는 반면 채널 수는 Bottleneck 이후 점차 감소하며, Level 4에서 512→256채널, Level 3에서 256→128채널, Level 2에서 128→64채널로 축소되고, 마지막 Level 1에서는 64채널을 유지하며 Output 단계로 연결된다.

2.1.5 Output



<Fig 2.1.5 Output>

출력층에서는 먼저 3×3 Convolution과 ReLU를 적용하여 Decoder에서 생성된 feature를 RGB img로 조정한다. 그리고, 7x7 Convolution을 통해 deblur 후의 복원된 RGB img를 생성한다. 이 과정에서 Residual Learning을 적용하여 최종 출력은 입력 이미지와 네트워크 출력의 합으로 계산된다. 마지막으로 Clamp(0, 1)을 적용하여 픽셀 값을 안정적인 이미지 범위로 제한함으로써 시각적으로 자연스러운 결과를 얻을 수 있도록 한다.

2.2 Loss Functions

학습 과정에서 품질 향상과 시각적 안정성을 모두를 확보하기 위해 loss function을 3개 사용한다.

1) L1 Loss: 예측 이미지와 정답 이미지 간의 픽셀 단위 절대 오차를 계산하여 전체적인 픽셀 정확도를 보장한다.

2) Perceptual Loss: 사전 학습된 VGG16 네트워크의 중간 feature map을 기반으로 계산되며, 인간과 유사한 관점에서 이미지 품질을 평가한다. 이를 위해 conv1_2, conv2_2, conv3_3의 세 단계 VGG 특징을 활용한다.

* vgg = models.vgg16(weights=models.VGG16_Weights.IMAGENET1K_V1).features

3)Total Variation Loss: 인접 픽셀 간의 차이를 최소화하여 이미지의 품질을 향상시키고 노이즈를 억제하는 데 사용된다.

최종적으로 사용되는 Total Loss는 L1 Loss, Perceptual Loss, Total Variation Loss를 각각의 가중치 λ_1, λ_2, λ_3에 따라 선형 결합한 형태로 정의된다. 본 모델에서는 $\lambda_1 = 1.0$, $\lambda_2 = 0.2$, $\lambda_3 = 0.00005$로 설정하여, 전반적으로 균형 잡힌 학습이 이루어지도록 구성하였다.

2.3 Learning Strategies

학습 전략의 최적화 측면에서는 Adam Optimizer(β_1=0.9, β_2=0.999)를 사용하였고, learning rate는 1e-4로 설정하였다. 또한 Weight Decay(1e-4)를 적용하여 L2 정규화를 수행함으로써 모델의 overfitting을 방지하고 weight regularization을 적용하였다.

정규화 기법으로는 모든 Convolution Layer 뒤에 Batch Normalization을 적용해 학습의 안정성과 수렴 속도를 높였으며, Residual Block 내부에는 Dropout(0.1)을 사용하여 모델이 다양한 특징을 학습하도록 유도하였다.

Learning rate 조절은 스케줄링 기법을 활용해, Validation PSNR이 8 epoch 동안 개선되지 않을 경우 learning rate를 절반으로 감소시키도록 설정하였다. 이를 통해 최적점을 탐색할 수 있게 한다. Early Stopping 역시 적용되어, Validation PSNR이 일정 기간 향상되지 않으면 학습을 종료하도록 하여 불필요한 연산을 줄이고 과적합을 방지한다.

또한 학습 효율을 높이기 위해 Mixed Precision Training(FP16)을 적용하였다. Half Precision 학습은 GPU 메모리 사용량을 줄이고 연산 속도를 향상하였다.

2.4 Data Augmentation

학습 이미지는 256×256 크기로 random crop하여 다양한 위치의 패턴을 학습하도록 하였다. 이어서 Horizontal Flip과 Vertical Flip을 각각 50% 확률로 적용하여, 좌우 및 상하 반전에 대해 robust한 표현을 학습할 수 있도록 구성하였다.

또한 90°, 180°, 270° 회전을 50% 확률로 수행하는 Random Rotation을 사용하여 다양한 방향의 blur 패턴이 존재하는 환경에서도 안정적으로 복원할 수 있는 능력을 확보하였다.

2.5 Hyperparameter

BATCH_SIZE	NUM_EPOCHS	LEARNING_RATE	WEIGHT_DECAY	DROPOUT
8	350	1e-4	1e-4	0.1

CROP_SIZE	NUM_WORKERS	PATIENCE	BASE_CHANNELS
256	4	15	64

결론

Input	Output

<Fig 3.1 Original Image vs Deblurred Image in GoPro Dataset>

이번 프로젝트에서는 ResNet U-Net 구조를 기반으로 image deblurring 모델을 구현하여 27.84dB PSNR(validation set을 활용한 자체 평가)을 달성하였다. 강의에서 학습한 CNN, Residual Connection, Batch Normalization, Dropout 등의 기본 기법만으로 의미 있는 성능을 구현하였으며, 추론 결과에서 효과적인 blur 제거를 눈으로 확인할 수 있었다.

추가적으로 성능 개선을 하기 위해서는, Attention mechanism, GAN의 adversarial loss, Transformer 기반 구조 등을 활용할 수 있을 것으로 예상된다.

본 프로젝트를 통해 강의 이론의 실전 적용, PyTorch 모델 구현, 하이퍼파라미터 튜닝 경험을 쌓았으며, 기본 기법만으로도 의미 있는 성능을 달성할 수 있음을 확인하였다.