딥러닝 기반 이산웨이블릿변환 네트워크

*이주원 박찬승 윤영재 김동욱
광운대학교
chagogo9@kw.ac.kr

Discrete Wavelet Transform Network based on Deep Learning

*Ju-Won Lee Chan-Seung Park Young-Jae Yoon Dong-Wook Kim
Kwangwoon University

요약

본 논문에서는 영상 변환 기술인 이산웨이블릿변환(Discrete Wavelet Transform, DWT)를 딥러닝 기반의 네트워크로 구현한다. 딥러닝 기술 중에도 CNN 기반으로 네트워크를 설계하였으며, 본 DWT 네트워크는 해상도에 의존적이지 않은 계층들로만 구성된다. 데이터세트를 구성할 때 파이썬의 라이브러리를 사용하여 레이블 데이터세트를 구성한다. 128x128크기의 gray-scale 영상을 입력으로 사용하고 이에 대응하는 레이블 데이터세트를 구성하여 1-level DWT를 수행하는 네트워크의 학습을 진행한다. 역방향 변환도 네트워크 설계 후 데이터세트를 구성하여 학습을 진행한다. 학습이 완료된 1-level DWT 네트워크를 반복적으로 사용하여 Multi-level DWT 네트워크를 구성한다. 또한 양자화에 의한 간단한 영상압축 실험을 진행하여 DWT 네트워크의 성능과 압축 등의 응용분야에 활용할 수 있음을 보인다. 설계한 DWT 네트워크의 1-level 순방향 변환 성능은 42.18dB의 PSNR을 보였고, 1-level 역방향 변환 성능은 50.13dB의 PSNR을 보였다.

1. 작품의 제작 동기

영상처리 기술의 발달과 다양한 영상 플랫폼 서비스가 개발됨으로 인해 고화질의 대용량 영상 데이터를 빠르고 손실 없이 네트워크로 전송할 수 있는 영상 압축에 대한 연구가 활발하게 진행되고 있다. 그 중 JPEG2000의 표준 변환으로 지정되어 쓰이고 있는 이산웨이블릿변환(Discrete Wavelet Transform, DWT)은 전체 영상을 대상으로 변환을 수행하므로 블록 효과가 없고 사람의 시각적 인식에 따른 처리가 가능하다[1].

딥러닝을 이용하여 기존의 알고리즘적인 영상처리 기술의 성능을 높인 사례가 많이 발표되고 있다. 이 점에 착안하여 DWT를 수행하는 네트워크를 설계하고 학습시켜 DWT 성능 향상을 기대할 수 있을 것이라고 판단하였다. 기존의 딥러닝 기반의 연구에서 전처리 혹은 네트워크 중간 계층으로 DWT를 사용하는 연구가 발표되었다[2]. 이 네트워크는 4개의 부대역에 대해 각각 다른 필터를 정의하고 있어서 네트워크가 복잡하고 네트워크의 깊이가 24층이어서 연산량과 메모리 소요량이 과다하다.

이에 본 연구에서는 DWT의 부대역에 따른 필터를 따로 정의하지 않고 네트워크가 스스로 학습하여 각 부대역을 연산하도록 하는 네트워크를 설계하고 학습시키고자 한다. 또한 네트워크의 깊이를 최소화하여 연산량과 메모리 사용량을 최소화하고, 네트워크에 영상의 해상도와 관련된 계층을 사용하지 않음으로써 학습하지 않은 해상도의 영상에도 적용할 수 있는 해상도-적응적인 DWT 변환 및 역변환 뉴럴 네트워크(neural network)를 제안하고자 한다. 본 연구의 결과로는 1-레벨(1-level) DWT 및 역DWT(inverse DWT, IDWT)의 성능과 다층(multi-level) DWT 및 IDWT의 성능을 보이며, 제안한 네트워크의 활용 가능성을 보이기 위해 양자화에 의한 간단한 압축실험을 수행하여 그 결과를 보인다.

2. 작품의 설계 및 구현

2.1 네트워크 구조

그림 1은 설계한 1-level DWT 네트워크(DWT convolutional neural network, DWTCNN)의 순방향(fDWTCNN (a))과 역방향(iDWTCNN (b)) 구조를 각각 보이고 있으며, 각 네트워크의 세부구조는 표 1과 표 2에 나타내었다. 그림에서 각 ConvBlock은 Convolution, BN(batch-normalization), AF(activation function)이 연속적으로 연결된 구조로 구성되어 있다.

영상에 1-level 순방향 DWT를 수행하면 원본 영상의 1/4 해상도를 갖는 4개의 부대역 (LL, LH, HL, HH)이 생성된다. 따라서 fDWTCNN에서는 5개의 ConvBlock을 거치면서 순방향 DWT에 적합한 특징을 추출하고, 마지막 ConvBlock(F_ConvBlock 6)에서 4개의 필터를 통해 4개의 채널로 해상도가 1/4(stride=2)인 4개의 부대역을 생성한다. DWT를 거친 후의 데이터가 AF에 의해 제한받지 않도록 마지막 ConvBlock에서는 AF와 BN을 사용하지 않으며, 나머지 ConvBlock은 ReLU(rectified linear unit)를 AF로 사용하고, BN을 수행한다.

iDWTCNN은 4개 부대역의 영상 데이터를 4개의 채널로 입력받아 5개의 ConvBlock을 거치면서 역방향 DWT에 적합한 특징을 추출한다. 이후 마지막 ConBlock(I_ConvBlock6)에서는 순방향 네트워크와는 반대로 영상 해상도를 4배(stride=1/2)로 키우는 필터 1개를 통해 원 영상에 해당하는 영상 데이터를 생성한다. iDWTCNN 역시 마지막

ConvBlock은 BN을 수행하지 않지만, 모든 ConvBlock은 ReLU AF을 수행한다.

Multi-level DWT는 이전 level fDWTCNN의 출력 중 LL 부대역 데이터를 fDWTCNN에 입력하여 구할 수 있다. 역방향 Multi-level DWT 또한 i-level iDWTCNN 출력(LL_i)을 i-level의 다른 부대역들(LH_i, HL_i, HH_i)과 함께 iDWTCNN에 입력하여 $(i-1)$-level의 LL 부대역(LL_{i-1})을 생성한다.

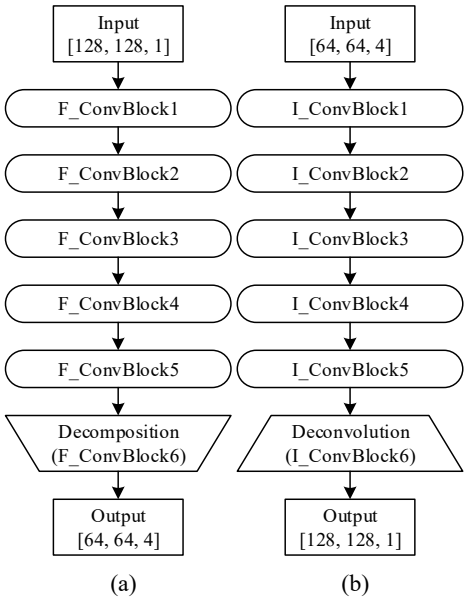

그림 1. (a) 순방향 DWT 네트워크 (b) 역방향 DWT 네트워크

표 1. 세부적인 순방향(Forward) DWT 네트워크

layer	Stride	# Kernels	Activation	BN
F_ConvBlock1	1	32	ReLU	o
F_ConvBlock2	1	32	ReLU	o
F_ConvBlock3	1	32	ReLU	o
F_ConvBlock4	1	32	ReLU	o
F_ConvBlock5	1	32	ReLU	o
F_ConvBlock6	2	4	x	x

표 2. 세부적인 역방향(Inverse) DWT 네트워크

layer	Stride	# Kernels	Activation	BN
I_ConvBlock1	1	32	ReLU	o
I_ConvBlock2	1	32	ReLU	o
I_ConvBlock3	1	32	ReLU	o
I_ConvBlock4	1	32	ReLU	o
I_ConvBlock5	1	32	ReLU	o
I_ConvBlock6	1/2	1	ReLU	x

3. 작품의 구현 및 결과

3.1 구현 및 실험

3.1.1 구현 및 실험환경

본 연구에서 제안하는 네트워크는 텐서플로우 환경에서 파이썬으로 구현하였다. 학습 및 테스트에 사용된 PC의 CPU 는 Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz이며 180GB RAM을 사용하고 있고, GPU는 NVIDIA Tesla V100-SXM2-32GB를 사용하였다.

3.1.2 데이터세트

본 연구에서는 512x512 gray scale 영상인 BoSSBaseDataset[3]으로 데이터세트를 구성하였다. 총 10,000장의 영상 데이터를 학습데이터 8,000장, 학습 시 검증을 위한 검증 데이터 1,000장, 학습이 끝난 후 실제 시험을 위한 시험데이터 1,000장으로 분류하여 사용하였다.

DWT 네트워크를 학습시키기 위해 512x512 크기의 영상을 128x128 크기의 영상으로 변환하고 변환된 128x128 영상 10,000장을 무작위로 선택하여 학습, 검증, 시험 데이터세트를 구성하였다. 파이썬 라이브러리[4]를 이용하여 128x128 영상들을 알고리즘적으로 DWT 취해서 정답 레이블(label)을 구하여 사용하였다.

시험 과정에서는 Multi-level DWT와 해상도 실험을 위해 64x64, 256x256, 512x512 크기의 데이터도 시험 데이터세트와 정답 레이블을 구성하여 준비하였다.

3.1.3 손실함수와 하이퍼 파리미터들의 설정 및 학습

손실함수로는 네트워크 출력과 정답 레이블 간의 평균제곱오차(Mean Square Error, MSE)를 사용한다. 학습은 0.001의 learning rate로 손실함수 값이 최소가 되도록 Adam Optimizer를 사용해서 진행하였다. 학습을 위한 mini-batch size를 100으로 설정하였고, 4000epoch까지 학습을 진행하였다. 이 환경에서 순방향과 역방향 네트워크를 모두 학습시키는데 약 4일의 시간이 소요되었다.

3.2 실험결과

본 절에서는 1-level DWT의 성능 실험, 해상도-적응적인 실험, Multi-level DWT의 성능 실험, 양자화에 의한 간단한 압축 실험 결과를 보인다.

3.2.1 1-level DWT

표 3은 제안한 DWTCNN으로 1-level DWT를 수행한 결과 영상의 평균 화질을 PSNR(peak signal to noise ratio)로 나타낸 것이다. 순방향과 역방향 모두 학습 데이터에 과적합(overfitting)되어 있지 않았다. 전체적으로 순방향은 40dB, 역방향은 50dB를 넘는 높은 PSNR을 보였으며, 순방향보다 역방향 DWT가 약 8dB 정도 더 우수한 성능을 보였다. 순방향 DWT 중 LH와 HL 부대역이 상대적으로 낮은 성능을 보였는데, 그 이유와 해결방법은 아직 찾지 못하고 있다. 그림 2와 그림 3에 순방향과 역방향 DWT 결과의 예들을 보이고 있는데, 두 결과 모두 육안으로는 네트워크 출력과 정답 레이블의 차이를 구분할 수 없다.

표 3. 1-level DWT 결과의 평균 화질 (PSNR[dB])

	Forward DWT					Inverse DWT
	LL	LH	HL	HH	Total	
Train	41.26	35.79	37.19	46.91	42.62	50.84
Validation	41.17	35.33	37.16	46.87	42.26	50.13
Test	41.15	35.17	37.04	46.89	42.18	50.13

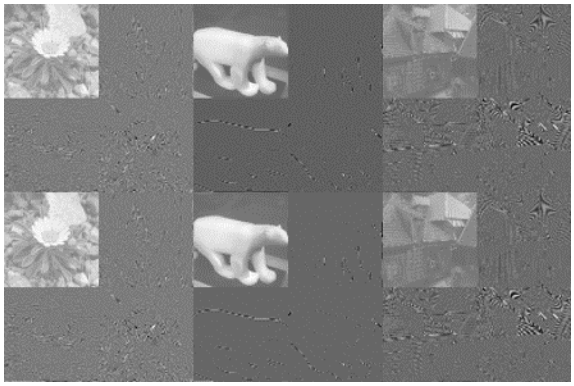

그림 2. 순방향 DWT 네트워크 실험 결과의 예 (위 : 네트워크 출력, 아래 : 정답 레이블)

그림 3. 역방향 DWT 네트워크 실험 결과의 예 (위 : 네트워크 출력, 아래 : 정답 레이블)

3.2.2 해상도-적응적성 실험

본 연구에서 설계한 네트워크는 해상도에 의존적인 계층이 사용하지 않기 때문에 입력 영상의 어떤 해상도에도 DWT를 수행할 수 있다. 그림 4에 다양한 입력영상의 해상도에 대한 실험 결과를 보이고 있다. 이 실험은 입력영상의 해상도를 64×64에서 512×512까지 수행한 것으로, 해상도가 커질수록 성능이 약간이나마 증가하는 경향성을 보였다. 하지만 PSNR이 가장 낮은 64x64 해상도의 순방향 DWT 결과도 40dB 정도의 성능으로 1-level DWT의 성능이 낮은 해상도에서도 우수함을 알 수 있다. 표 3에서 본 바와 같이, 모든 해상도에서 순방향보다 역방향 DWT의 성능이 우수하였으나, 해상도가 증가할수록 그 격차가 줄어드는 경향성을 보였다.

3.2.3 Multi-level DWT

3.2.1의 결과로 1-level DWT 네트워크를 반복적으로 사용하여 multi-level DWT 네트워크를 구현하는 것 또는 단일 DWTCNN으로 multi-level DWT 수행이 가능하다는 것을 확인하였다. 표 4는 multi-level DWT 실험결과를 보이고 있다. 원본 영상으로는 512x512 해상도의 영상을 사용했고, 3-level까지 실험을 진행했다. 3-level에서도 43dB이상의 높은 PSNR값을 보이는 것을 볼 수 있다.

그림 4. 입력영상의 해상도 변화에 대한 실험 결과

표 4. Multi-level DWT 실험결과

fDWT/iDWT	1-level	2-level	3-level
forward DWT	47.44dB	46.02dB	45.46dB
inverse DWT	52.32dB	46.82dB	43.26dB

3.2.4 양자화에 의한 간단한 압축실험

본 연구에서는 제안한 fDWTCNN과 iDWTCNN의 성능을 DWT의 활용분야에서 확인하기 위하여 DWT와 간단한 양자화를 이용한 영상압축 실험을 수행하였다. 그림 5는 양자화를 이용한 압축실험의 과정을 나타내고 있다. 압축 scheme은 영상을 1-level DWT한 후 LL 부대역을 제외한 나머지 세 부대역을 양자화기로 양자화하고, 그 결과를 역방향 DWT를 수행하여 압축된 영상을 얻는 것이다. 여기서 양자화는 LL 부대역을 제와한 나머지 세 부대역을 같은 비트를 사용하여 양자화하며, 양자화기는 비대칭(middle-rise) 균등 양자화기를 사용하였다. 그림 6은 압축실험의 결과를 나타내고 있는데, 그림에서 Q-level(Quantization level)은 양자화 level을 나타내는 값으로, n비트로 양자회한 경우 Q-level은 2^n을 갖는다. 그림 6의 PSNR은 계산에 의한 결과와 제안한 네트워크에 의한 결과 각각을 원 영상에 대해 구한 값이다.

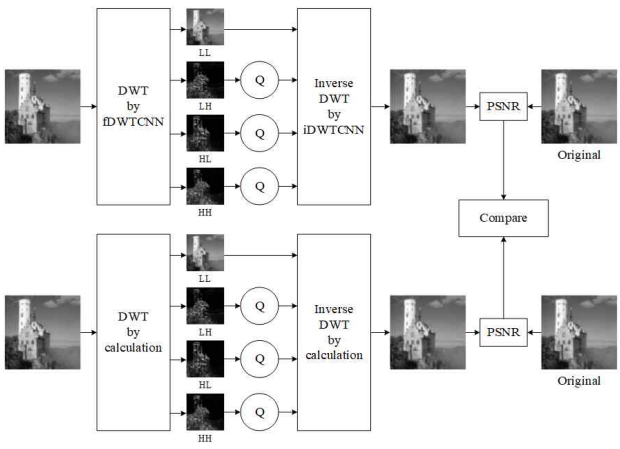

그림 5. 양자화를 이용한 영상 압축실험 개략도

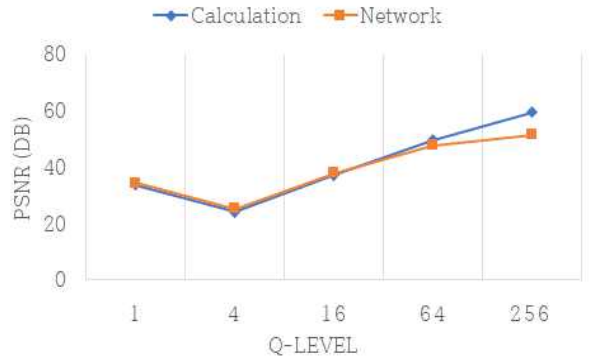

그림 6. 압축실험결과

그림에서 보듯이, 두 경우 모두 Q-level이 4인 경우(2비트로 양자화한 경우) 경향성에 벗어나 낮은 값을 보였으나, 그 외의 경우는 모두 양자화 강도가 증가함에 따라(Q-level 수가 작아짐에 따라) 화질의 열화가 심해지는 일반적인 특징을 그대로 보이고 있다. 제안한 네트워크를 사용하는 경우가 계산에 의한 경우보다 Q-level=16까지는 계산에 의한 경우보다 좋은 결과를 보였으나, 그보다 높은 Q-level의 경우는 제안한 네트워크의 성능이 다소 떨어지는 경향을 보였다. 이것은 표 4에서 보인 것과 같이, 네트워크의 DWT와 역변환 자체가 일으키는 열화 때문인 것으로 사료된다. 즉, 계산에 의한 경우 DWT는 역변화에 의해 완벽한 복원이 가능하여 양자화에 의한 열화현상만 포함되지만, 네트워크로 DWT와 역변환을 수행하는 경우 DWT와 역변환에 의한 열화와 양자화에 의한 열화가 더해져 나타난 결과이다. 따라서 네트워크에 의한 DWT/iDWT를 이용하는 경우 네트워크의 열화보다 더 강한 압축에 더욱 유용할 것으로 판단된다.

4. 작품의 기대효과

본 작품은 DWT의 부대역에 따른 필터를 따로 정의하지 않고 네트워크가 스스로 학습하여 각 부대역을 연산하도록 하는 1-level DWT 네트워크를 설계하였다. 본 네트워크는 영상의 해상도와 연관된 계층을 사용하지 않음으로써 학습하지 않은 해상도의 영상에도 적용할 수 있다는 것을 3.2절의 실험을 통해 입증했다. 또한 네트워크 깊이가 깊지 않아 연산량과 메모리 사용량이 적다는 장점이 있다. 이러한 이유로 다른 딥러닝 기반 영상처리 네트워크와의 효율적인 호환성을 기대할 수 있다.

본 연구의 실험결과로는 1-level DWT의 결과에서 순방향 및 역방향 모두 42dB가 넘는 높은 PSNR 결과를 보였다. 또한 3-level을 시행한 Multi-level DWT의 결과에서도 순방향과 역방향 모두 43dB 이상의 성능을 보이며, DWT를 수행하는 네트워크로서의 성능을 입증했다. 또한 양자화에 의한 간단한 압축실험을 수행한 결과를 통해 압축과 같은 활용 분야에서도 충분히 이용 가능할 것이라고 사료된다.

5. 참고문헌

[1] D. Sundararajan, “Discretewavelet Transform: A Signal Processing Approach”, 2015 John Wiley & Sons, Singapore Pte. Ltd., 2015

[2] P. Liu, H. Zhang, W. Lian and W. Zuo, "Multi-Level Wavelet Convolutional Neural Networks," in IEEE Access, vol. 7, pp. 74973-74985, 2019, doi: 10.1109/ACCESS.2019.2921451.

[3] 512x512 grayscale BoSSBaseDataset url : http://agents.fel.cvut.cz/boss/index.php?mode=VIEW&tmpl=materials

[4] Gregory R. Lee, Ralf Gommers, Filip Wasilewski, Kai Wohlfahrt, Aaron O’Leary (2019). PyWavelets: A Python package for wavelet analysis. Journal of Open Source Software, 4(36), 1237, https://doi.org/10.21105/joss.01237.