딥 러닝 기반의 무손실 영상압축 방법

*이호창 **조남익
서울대학교 뉴미디어통신공동연구소 정보신호처리연구실
*hochang@ispl.snu.ac.kr

Lossless Image Compression Based on Deep Learning

*Rhee, Hochang **Cho, Nam Ik
Department of ECE, INMC, Seoul National University

요약

최근 딥러닝 방법의 발전하면서 영상처리 및 컴퓨터 비전의 다양한 분야에서 딥러닝 기반의 알고리즘들이 그 이전의 방법들에 비하여 큰 성능 향상을 보이고 있다. 손실 영상 압축의 경우 최근 encoder-decoder 형태의 네트웍이 영상 압축에서 사용되는 transform을 대체하고 있고, transform 결과들의 엔트로피 코딩을 위한 추가적인 encoder-decoder 네트웍을 사용하여 HEVC 수준에 버금가는 성능을 내고 있다. 무손실 압축의 경우에도 매 픽셀 예측을 CNN으로 수행하는 경우, 기존의 예측방법들에 비하여 예측성능이 크게 향상되어 JPEG-2000 Lossless, FLIF, JEPG-XL 등의 딥러닝을 사용하지 않는 방법들에 비하여 우수한 성능을 내는 것으로 보고되고 있다. 그러나 모든 픽셀에 대하여 예측값을 CNN을 통하여 계산하는 방법은, 영상의 픽셀 수만큼 CNN을 수행해야 하므로 HD 크기 영상에 대하여 지금까지 알려진 가장 빠른 방법이 한 시간 이상 소요되는 등 비현실적인 것으로 알려져 있다. 따라서 최근에는 성능은 이보다 떨어지지만 속도를 현실적으로 줄인 방법들이 제안되고 있다. 이러한 방법들은 초기에는 FLIF나 JPEG-XL에 비하여 성능이 떨어져서, GPU를 사용하면서도 기존의 방법보다 좋지 않은 성능을 보인다는 면에서 여전히 비현실적이었다. 최근에는 신호의 특성을 더 잘 활용하는 방법들이 제안되면서 매 픽셀마다 CNN을 수행하는 방법보다는 성능이 떨어지지만, 짧은 시간 내에 FLIF나 JPEG-XL보다는 좋은 성능을 내는 현실적인 방법들이 제안되었다. 본 연구에서는 이러한 최근의 몇 가지 방법들을 살펴보고 이들보다 성능을 더 좋게 할 수 있는 보조적인 방법들과 raw image에 대한 성능을 평가한다.

1. 서론

고품질 영상에 대한 요구가 증가함에 따라, 영상 압축의 중요성이 커지고 있다. 고품질 영상은 그 양이 매우 크므로 일반적으로 손실 압축이 선호되지만 무손실 압축 압축이 필요한 경우도 많이 있다. 예를 들어 의료 이미지, 과학 이미지, 기술 도면 및 예술 사진 등에는 무손실 압축을 해야 할 경우들이 있다. 이를 위한 표준안으로서 JPEG2000(무손실 모드)[7]과 같은 방법은 이산 웨이블릿 변환(DWT)을 사용한 변환 코딩을 사용하지만 대부분의 표준/비표준 무손실 압축 방법은 예측 코딩을 사용한다.

표준안 방법들과 마찬가지로 초기의 딥러닝 기반 무손실 압축 알고리즘은 전체적인 구조를 자동 회귀 모델 (auto-regressive model)로 설계하였다. 즉 이전 샘플을 기준으로 다음 픽셀의 확률 분포를 추정할 때 deep neural network (DNN)이 기존의 선형예측방법 방법보다 훨씬 정확한 추정을 한다는 것을 이용하였다. 그러나 이러한 방법은 전체 픽셀 수에 대한 신경망 계산이 필요하므로 추론 시간이 비실용적이며, 구체적인 예로 HD 영상 한 장 압축에도 한 시간 이상의 시간이 소요된다. 따라서, 보다 실용적인 방법으로서 최근에는 개별 픽셀이 아닌 전체 이미지 또는 sub-image 단위로 인코딩을 수행하는 방법들이 제시되어 왔다[11,15]. 이러한 방법들은 이전에 인코딩된 sub-image에 대한 다음 sub-image의 확률 분포를 구하거나 손실 압축된 이미지에 대비 전체 이미지의 확률 분포를 DNN을 통하여 도출함으로서 무손실 압축을 수행한다. 그 결과로 이전의 픽셀 별 DNN 수행 방법들에 비하여 수행시간을 크게 단축시켜서 실용성을 높였다고 할 수 있지만, 압축 성능이 기존의 표준 방법인 JPEG-XL[12]이나 최근의 딥러닝을 이용하지 않는 FLIF[10]에 비하여 성능이 떨어진다는 면에서 여전히 실용적이라 할 수 없었다. 즉, DNN 연산을 위하여 GPU를 사용하면서도 기존의 CPU 기반의 방법에 비하여 우수한 성능이 나오지 않으므로 실제 현장에 사용되기 어려운 방법이라 할 수 있다.

따라서, 최근에 간단한 MLP 만을 이용하여 픽셀값 예측과 함께 엔트로피 코딩을 위한 컨텍스트를 함께 수행함으로써 수행시간을 줄이면서 FLIF보다 성능을 조금 더 올린 방법이 제안되었다[19]. 또한, 즉, DNN이 픽셀값을 아무리 잘 예측해도 영상의 특성에 따라 예측값의 분포가 달라지므로 이의 효과적인 엔트로피 코딩을 위한 정보를 함께 예측함으로써 성능을 높일 수 있음을 보였다. 마찬가지로, 영상을 영역별 특성에 따라 나누고 이들을 별도로 예측하고 인코딩함으로써 성능을 더욱 높이고, 영역별 수행 특성으로 수행시간도 크게 줄인 방법도 제안되었다[20]. 본 논문에서는 이와 같은 최근 방법들을 살펴보고, 이들을 더욱 발전 방향과 이들의 raw image 압축 성능을 평가한다.

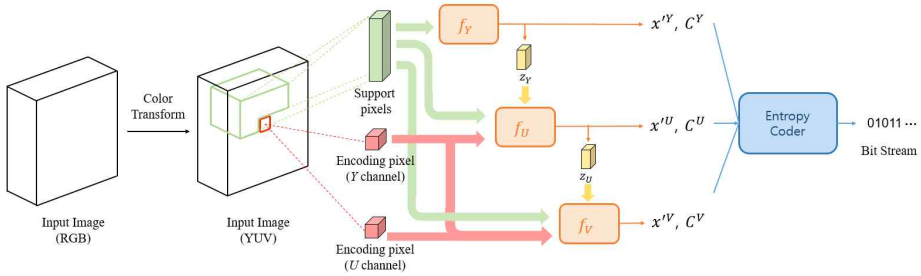

그림 1. 제안하는 방법의 전체적인 구조

2. MLP 기반의 픽셀/컨텍스트 예측 방법[19]

제안하는 방법의 전체적인 구조는 그림 1과 같다. 먼저 입력 RGB 이미지에서 색상 간의 상관관계를 최소화하기 위해서 [1]에서 제안한 가역적 색상 변환을 적용하여 YUV 이미지로 변환한다. 그 후, 압축하고자하는 픽셀 x_i^c에 대하여 픽셀에 대한 예측값 x'^c_i 및 코딩 컨텍스트 C_i^c를 래스터 스캔 순서대로 생성한다. 이 때 i는 픽셀 인덱스이고 $c = \{Y, U, V\}$는 색상 인덱스이다. 여기서 코딩 컨텍스트는 주변 픽셀과의 차이를 나타내는 변수로 이미지의 매끄러운 영역에서는 값이 작고 텍스처 영역에서는 값이 크다. 마지막으로 픽셀 값과 예측값의 차이를 비트스트림으로 압축하기 위하여 adaptive arithmetic coder (AAC) [2]를 활용한다.

먼저 압축하려는 픽셀 x_i^c에 대하여 픽셀에 대한 예측값 x'^c_i 및 코딩 컨텍스트 C_i^c를 얻는 과정을 설명한다. x_i^c에 대한 정보를 예측하기 위해서 우리는 인과적인 이웃 픽셀들을 활용한다. 하지만 모든 인과적인 이웃 픽셀을 활용하는 것은 비효율적이므로 우리는 현재 픽셀과 특정 거리 내에 있는 인과적인 이웃 픽셀들을 지원픽셀이라 칭하고 지원픽셀을 사용한다. 이 때 x_i^c에 대한 지원픽셀은 x_i^{sup}이라 지칭한다. 지원픽셀에 대한 예시는 그림 2와 같다.

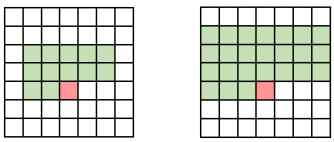

그림 2. 거리에 따른 지원픽셀 예시. 빨강은 현재 픽셀을, 초록은 지원픽셀을 나타낸다. (a)는 거리가 2인 경우, (b)는 거리가 3인 경우.

이 때 우리는 지원픽셀로부터 YUV 성분들을 바로 예측하지 않는다. Y는 지원픽셀을 활용하지만, U는 지원 픽셀과 더불어 앞의 Y 성분도 활용한다. 마찬가지로 V는 지원픽셀과 Y, U 성분을 활용한다. 이렇게 진행 할 경우, U, V 성분들은 Y 성분에 담겨있는 유의미한 정보를 활용하여 보다 더 정확한 예측을 할 수 있다. 추가적으로 x_i^c 값을 바로 예측하는 것보다 $r_i^c = x_i^c - x_{i-1}^c$를 예측하도록 설계하였다. x_i^c 값은 분산이 큰 반면, r_i^c는 평균이 0이고 분산이 작은 분포를 띄므로 네트워크 입장에서 학습이 용이하다. 네트워크는 간단한 다층 퍼셉트론 (MLP)를 활용하여 연산량을 최소화하였다.

네트워크에서 출력된 $r_i^c - r'^c_i$ 는 실제 비트스트림으로 변환이 되고 코딩 컨텍스트 C_i^c는 $r_i^c - r'^c_i$를 비트스트림으로 변환해줄 AAC를 할당한다. 우리는 총 24개의 AAC를 사용하고 $q_{n-1} \leq C_i \leq q_n$ 일 때 C^n 에 할당한다. 이 때 q_n은 0에서 시작하여 0.25 씩 증가하고 1.5 이후로는 0.5씩 증가시킨다.

네트워크의 학습은 총 2가지 손실함수를 사용한다. 네트워크의 첫 번째 목적은 픽셀 값 예측을 정확하게 하는 것이고 이는 다음 손실함수를 통해서 달성한다.

$$L_P^c = \min_{f_Y, f_U, f_V}(\sum_i |r_i^c - r_i'^c|).$$

네트워크의 두 번째 목적은 코딩 컨텍스트 C_i^c가 텍스쳐 영역의 정도에 비례하는 값을 가지게 하는 것이다. 이는 다음 손실함수를 통해서 달성한다.

$$L_C^c = \min_{f_Y, f_U, f_V}(\sum_i |C_i^c - |r_i^c - r'^c_i||).$$

3. MLP 방법의 결과 및 분석

우리는 제안하는 방법을 다음의 3가지 고해상도 데이터셋에 대해서 실험을 하였다 : McMaster [3], Open Images [4] 그리고 DIV2K [5]. 모델의 학습은 DIV2K에서 제공되는 800개의 이미지로 진행하였다. 비교하는 방법으로는 엔지니어링 기반의 코덱들 PNG [6], JPEG2000 [7], WebP [8], LCIC [9], FLIF [10] 와 딥러닝 기반의 방법 L3C [11]를 채택하였다. 압축 성능 측정은 널리 사용되는 bits per pixel (bpp)를 사용했다.

표 1. 데이터셋에 따른 방법 별 성능 표

데이터셋	MCMaster	Open Images	DIV2K
PNG	13.97 +24.4%	12.98 +32.7%	13.25 +57.8%
JPEG2000	12.50 +11.3%	11.71 +19.7%	12.80 +52.3%
WebP	11.86 +5.6%	11.09 +13.4%	11.92 +42.0%
LCIC	12.30 +9.5%	10.98 +12.3%	8.88 +5.8%
FLIF	11.59 +3.2%	10.23 +4.6%	8.29 -1.2%
L3C	12.24 +9.0%	11.18 +14.3%	9.52 +13.4%
Ours	11.23	9.78	8.39

데이터셋과 방법에 따른 성능 표는 표 1과 같다. 표에서 알 수 있듯이 우리의 방법은 엔지니어링 기반의 코덱들과 비교하였을 때 모든 데이터셋에 대해서 대체적으로 성능이 좋다. 그리고 딥러닝 기반 방법인 L3C와 비교하였을 때 모든 데이터셋에 대해서 최소 9% 이상의 성능 차이가 있는 것을 확인할 수 있다.

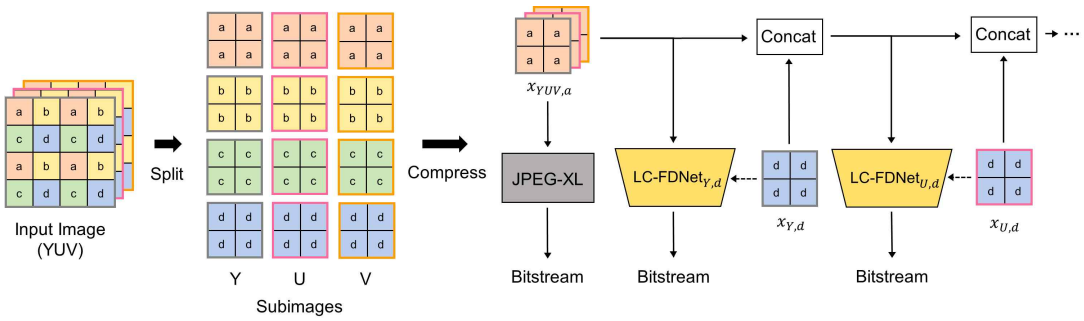

그림 3. 제안하는 방법의 전체적인 흐름

4. 주파수 특성 별 영상분할에 기반한 방법[20]

제안하는 방법은 크게 2가지 직관이 있다. 먼저 이미지를 4분할하여 각 분할마다 단계적으로 압축을 진행한다. 이 때 이미 압축된 분할의 정보를 활용하여 다음 분할의 압축을 진행하여 압축 성능을 높인다. 두 번째는 고주파 성분에서의 압축성능을 높이기 위하여 저주파 성분을 먼저 압축한 후에 압축된 저주파 성분을 이용하여 남은 고주파 성분을 압축한다. 우리는 우리의 방법을 LC-FDNet (Lossless Compression with Frequency Decomposition Network) 이라고 명명한다.

제안하는 방법의 전체적인 흐름은 그림 3과 같다. 먼저 입력 RGB 이미지에서 색상 간의 상관관계를 최소화하기 위해서 [1]에서 제안한 가역적 색상 변환을 적용하여 YUV 이미지로 변환한다. 그 이후 이미지를 그림3과 같이 영역에 따라 a,b,c,d로, 색상에 따라 Y,U,V로 총 12개의 하위 이미지로 분할한다. 12개의 하위 이미지를 단계별로 압축하며 먼저 3개의 하위 이미지 $x_{Y,a}, x_{U,a}, x_{V,a}$는 기존 코덱인 JPEG-XL [12]를 활용하여 압축한다. 그 이후 영역별로는 $a \rightarrow d \rightarrow b \rightarrow c$의 순서로, 색상별로는 $Y \rightarrow U \rightarrow V$의 순서로 압축한다. 결론적으로 12개의 하위 이미지는 $x_{Y,a} \rightarrow x_{U,a} \rightarrow x_{V,a} \rightarrow x_{Y,d} \rightarrow x_{U,a} \rightarrow \dots \rightarrow x_{V,c}$의 순서로 압축이 진행된다. 이 때 각 하위 이미지는 LC-FDNet을 통해 압축하며 이미 압축된 하위 이미지를 활용한다.

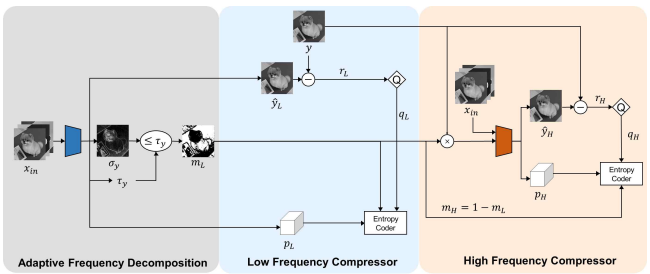

그림 4. LC-FDNet의 구조

LC-FDNet의 구조는 그림4와 같다. N번째 하위 이미지 y를 압축할 때 이미 압축된 $N-1$개의 하위 이미지를 concatenate 하여 입력으로 받는다. LC-FDNet은 크게 3가지로 구성된다. 먼저 적응 주파수 분해 (Adaptive Frequency Decomposition) 에서는 각 픽셀이 저주파 성분인지 고주파 성분인지 나눈다. 그 이후 저주파 성분들은 저주파 압축기 (Low Frequency Compressor)에서 비트스트림으로 압축된다. 마지막으로 고주파 압축기 (High Frequency Compressor) 에서는 $N-1$개의 하위 이미지와 압축된 저주파 성분을 활용하여 남은 고주파 성분들을 압축한다.

먼저 적응 주파수 분해와 저주파 압축기에 대해서 설명한다. LC-FDNet은 하위 이미지들을 입력으로 받아 4개의 출력물을 출력한다. 1) 먼저 하위 이미지 y의 저주파 성분에 대한 예측 $\hat{y}_L$를 한다. 차이값인 $r_L = \hat{y}_L - y$ 는 라운드 함수로 양자화하여 q_L을 얻는다. 2) 양자화된 차이값 q_L 에 대한 확률분포인 p_L을 얻는다. 이 때 p_L은 Y 색상에 대해서 채널차원의 크기는 511이고 U, V는 1021이다. p_L을 얻기 전에 소프트맥스 함수를 적용시켜 확률값의 합이 1이 되도록 한다. 3) 오차 분산 맵인 σ_y는 예측 오차의 크기를 나타낸다. 오차 분산 맵이 예측 오차의 크기에 비례하도록 다음과 같이 오차 분산 손실함수를 적용한다.

$$L_{ev} = \|\sigma_y - |y - \hat{y}_L|\|_1$$

4) 얻은 오차 분산 맵을 통해 단순한 임계값을 적용하여 해당 픽셀이 저주파 성분인지 고주파 성분인지 나눌 수 있다. 하지만 임계값은 하위 이미지에 따라 다르므로 우리는 네트워크가 임계값을 예측하도록 하고 이를 오차 분산 임계값 τ_y라고 명명한다.

우리는 얻은 오차 분산 맵과 오차 분산 임계점을 통해서 다음 식을 통해 픽셀을 저주파 성분과 고주파 성분으로 분류할 수 있다.

$$m_L^i = \begin{cases} 1 & if \quad \sigma_y^i \le \tau_y \\ 0 & else \end{cases}$$

저주파 압축기에서는 저주파 성분에 해당하는, $m_L^i = 1$인 픽셀들에 대하여 q_L을 p_L에 따라서 압축한다.

고주파 압축기는 저주파 압축기와 비슷한 구조다. 차이점은 입력으로 이미 압축된 현재 하위 이미지의 저주파 성분을 추가 정보를 받는다는 것이다. 네트워크의 출력은 하위 이미지 y의 고주파 성분에 대한 예측 $\hat{y}_H$와 그에 해당하는 확률 분포 p_H이다. 최종적으로 고주파 성분에 해당하는 $m_L^i = 0$인 픽셀들에 대하여 q_H을 p_H에 따라서 압축한다.

LC-FDNet은 3가지 손실함수로 학습한다: 위에 언급된 오차 분산 손실함수, 재건 손실함수 그리고 비트레이트 손실함수. 재건 손실함수와 비트레이트 손실함수는 각각 다음과 같다.

$$L_{rec} = m_L \bullet \|y - \hat{y}_L\|_1 + m_H \bullet \|y - \hat{y}_H\|_1$$

$$L_{br} = m_L \bullet \| - \log p_L(q_L)\|_1 + m_H \bullet \| - \log p_H(q_H)\|_1$$

5. 주파수 분할 방법 실험 결과 및 분석

우리는 LC-FDNet을 다음의 3가지 고해상도 데이터셋에 대해서 실험을 하였다 : Clic.m[13], Clic.p [13] 그리고 DIV2K [5]. 모델의 학습은 FLICKR2K [14]에서 제공되는 2,000개의 이미지로 진행하였다. 비교하는 방법으로는 기존 실험에서 딥러닝 기반의 방법인 RC [15]와 Near-Lossless [16]를 추가하였다.

표 2. 데이터셋에 따른 방법 별 성능 표

데이터셋	Clic.m	Clic.p	DIV2K
PNG	11.79 +70.9%	11.79 +50.0%	12.69 +55.9%
JPEG2000	7.59 +10.0%	8.79 +11.8%	9.36 +15.0%
WebP	8.19 +17.8%	8.70 +11.8%	9.33 +15.0%
LCIC	7.88 +14.2%	9.02 +14.8%	9.35 +14.9%
FLIF	7.44 +7.8%	8.16 +3.8%	8.73 +7.2%
JPEG-XL	7.20 +4.3%	8.19 +4.2%	8.49 +4.3%
L3C	7.92 +14.8%	8.82 +12.2%	9.27 +13.9%
RC	7.62 +10.4%	8.79 +11.8%	9.24 +13.5%
Near-Lossless	7.53 +9.1%	7.98 +1.5%	8.43 +3.6%
Ours	6.90	7.86	8.14

데이터셋과 방법에 따른 성능 표는 표 2와 같다. 표에서 알 수 있듯이 우리의 방법은 엔지니어링 기반의 코덱들과 딥러닝 방법들과 비교하였을 때 모든 데이터셋에 대해서 성능이 좋다.

감사의 글

This research was financially supported by the Ministry of Trade, Industry, and Energy (MOTIE), Korea, under the "Regional Specialized Industry Development Program (R&D, P0002072)" supervised by the Korea Institute for Advancement of Technology (KIAT)P0002072 and BK21 FOURprogram of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2022.

참고문헌

[1] Soo-Chang Pei and Jian-Jiun Ding, "Improved reversible integer-to-integer color transforms," in 2009 16th IEEE International Conference on Image Processing (ICIP). IEEE, 2009, pp. 473-476.

[2] Ian H Witten, Radford M Neal, and John G Cleary, "Arithmetic coding for data compression," Communications of the ACM, vol. 30, no. 6, pp. 520-540, 1987.

[3] Lei Zhang, Xiaolin Wu, Antoni Buades, and Xin Li, "Color demosaicking by local directional interpolation and nonloacl adaptive thresholding," Jorunal of Electronic imaging, vol. 20, no. 2, pp. 023016, 2011.

[4] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, et al., "Openimages: A public dataset for large-scale multilabel and multi-class image classification," Dataset available from https://github.com/openimages, vol. 2, pp. 3, 2017.

[5] Eirikur Agustsson and Radu Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 126-135.

[6] Thomas Boutell, "Png (portable network graphics) specification version 1.0," 1997.

[7] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi, "The jpeg 2000 still image compression standard," IEEE Signal processing magazine, vol. 18, no. 5, pp. 36-58, 2001.

[8] WebEngines Blazer Platform Version, "1.0 hardware reference guide, xp-002202892, network engines," Inc., Jun, vol. 1, pp. 92, 2000.

[9] Seyun Kim and Nam Ik Cho, "Hierarchical prediction and context adaptive coding for lossless color image compression," IEEE Transactions on image processing, vol. 23, no. 1, pp. 445-449, 2013.

[10] Jon Sneyers and Pieter Wuille, "Flif: Free lossless image format based on maniac compression," in 2016 IEEE International Conference on Image Processing (ICIP). IEEE, 2016, pp. 66-70.

[11] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10629-10638, 2019.

[12] Jyrki Alakuijala, Ruud van Asseldonk, Sami Boukortt, Martin Bruse, Iulia-Maria Coms, a, Moritz Firsching, Thomas Fischbacher, Evgenii Kliuchnikov, Sebastian Gomez, Robert Obryk, et al. Jpeg xl next-generation image compression architecture and coding tools. In Applications of Digital Image Processing XLII, volume 11137, page 111370K. International Society for Optics and Photonics, 2019. 1, 3, 6

[13] Workshop and challenge on learned image compression. https://www.compression.cc/challenge/. 5

[14] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pages 136-144, 2017. 5

[15] Fabian Mentzer, Luc Van Gool, and Michael Tschannen. Learning better lossless compression using lossy compression. In Proceedings of the IEEE/CVF Conference on Com- puter Vision and Pattern Recognition, pages 6638-6647, 2020. 1, 2, 3, 6

[16] Yuanchao Bai, Xianming Liu, Wangmeng Zuo, Yaowei Wang, and Xiangyang Ji. Learning scalable ly=-constrained near-lossless image compression via joint lossy image and residual compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11946-11955, 2021. 1, 6