딥러닝 기반 영상 압축, 복원, 물체 검출 연구

팀: KHUVCM
2019110631 윤규리
2019102210 이유제
2018102227 전현진

1. 서론

1.1. 연구배경

1.1.1 영상 압축

현대사회는 스마트폰, 스마트 TV, PC 등 미디어를 위한 기기와 기술들이 계속해서 발전하고 있기 때문에 사람들이 멀티미디어를 접할 수 있는 기회와 시간이 기하급수적으로 증가하고 있다. 따라서 그에 맞는 다양한 컨텐츠들이 탄생하고 있으며, 이들은 매우 거대한 시장을 이루고 있다.

멀티미디어는 인간의 오감(시각, 청각, 촉각, 후각, 미각)과 관련된 모든 미디어관련 기술/매체이다. 이중 시각 미디어는 사람의 감각기 중에서 받아들이는 정보가 가장 크기 때문에 영상은 멀티미디어에서 가장 중심적인 위치를 차지하고 있다.

또한 현재 각광받고 있는 분야 중 하나인 메타버스는 실감형 멀티미디어로, 가상현실(VR), 증강현실(AR), 인공지능(AI), 초고속통신망(5G/6G) 등을 이용하여 보편적 시각 미디어인 초고화질(4K/8K/NK) 동영상을 중심으로 발전하고 있다.

이러한 시각데이터들은 오래전부터 발전되어 왔기 때문에 현재는 매우 고화질의 영상을 쉽게 만나볼 수 있는데, 이들의 용량은 상당하다. 예를 들어 색상이 풍부하고(10bit), 60Hz의 2시간 분량 4K-UHD 영화는 이를 압축하지 않으면 약 13TB의 어마어마한 용량을 가진다. 따라서 영상의 압축은 선택이 아닌 필수적 요소라 볼 수 있다.

압축 기술은 정지영상, 동영상 구분 할 것 없이 꾸준한 발전을 이루어 왔다. 일반적으로 많이 사용하는 정지영상 압축 기술인 JPEG은 손실을 감수하면 최대 100:1 까지도 압축이 가능하며, 동영상의 경우에는 MPEG-1부터 시작해서 현재는 AVC, HEVC, VVC까지 고화질의 동영상도 웬만하면 문제없이 전송할 수 있도록 오랜 기간 발전을 이루어 왔다.

이제는 압축 기술에 AI기술을 접목시키려는 시도가 시작되었다. JPEG은 2019년3월부터 JPEG-AI 팀을 결성하여 꾸준한 정기회의를 해오고 있다. 그들의 목표는 일반적으로 사용되는 코딩 표준에 비해 상당한 압축 효율성 향상이며, 이는 새로운 패러다임으로 미디어 발전에 가속도를 붙여줄 것이다.

1.1.2 Object Detection의 Feature Map 열화 분석

객체 인식과 같은 task는 작은 물체를 인식해내야 하기 때문에 상당히 많은 양의 feature map을 사용해야 한다. 이미지에서 feature map을 추출한 후, 추출한 모든 feature map을 detector 부분에게 제

공하려면 상당히 많은 양의 정보를 전달해주어야 한다. 최근 모바일 기기와 클라우드 사이에서의 workload를 분배하여 Neural Network의 효율성을 향상시키는 방법인 Collaborative Intelligence이 제안되면서 효율적인 CI를 위해 모바일 기기와 클라우드 간의 데이터 정보량이 과도해지는 것을 방지해야 한다. 이를 위해 Feature Map을 압축하는 방법들이 다양하게 제안되고 있다. 그러나 아직까지 object scale 별 (small object / medium object / large object) 성능을 모두 고려하는 방법은 제안되지 않았다. 본 연구는 object scale 별 성능을 모두 고려하여 Feature Map을 압축했다는 점에서 강점을 가진다.

1.2. 연구목표

1.2.1 영상 압축

전통적인 압축 방식을 파이프라인을 활용하여 AI기반 정지영상의 압축을 목표로 한다. 아직 JPEG-AI의 표준이 없고, 정기회의에서 제안된 기술이 매우 방대하기 때문에 모든 기술을 적용시키는 것은 한계가 있다. 따라서 학습 기반 정지영상 압축의 대표적 논문 중 하나인 Minnen모델을 참고한다.

사람의 눈은 빛(명도)을 감지하는 간상세포가 색상을 감지하는 원추세포보다 약 20배 많아서 전통적인 정지영상 압축시에는 색상 정보로만 이루어진 'RGB' 색공간에서 명도와 색차정보로 이루어진 'YCbCr' 색공간으로 변환하여 사용하게 된다. 그러나 현재 제시된 학습 기반 압축 논문들은 대부분 RGB 색공간에서 학습하고, 오차를 계산한 결과를 제시하고 있다. 연구에서는 YCbCr 색공간 변환 후 제시된 model을 통과시켜서 RGB와 비교하고, 추가적으로 Y를 먼저 압축한 후 생성된 데이터들을 이용하여 CbCr을 압축해서 좀 더 나아진 성능의 인코더를 만들 것이다. 이를 통해 기존 Minnen Model보다 높은 압축률을 보여주는 것을 목표이다..

1.2.1. Object Detection의 Feature Map 열화 분석

본 프로젝트에서는 Object Detection 모델의 Feature Pyramid Network에서 얻어진 Feature Map들의 영향도를 분석하여, 영향도가 높은 Key Feature Map을 찾아내 해당 Feature Map만을 사용하여 object detection task를 수행했을 때 성능 하락이 최소화되는 방법을 제안하고자 한다. 이때 small / medium / large object에 대한 성능을 모두 고려하여 모든 scale에서 성능 하락에 강건한 방법을 찾고자 했다. 또한 Key Feature Map 외의 Feature Map을 열화시켜 전달한 후 복원시키는 방법이 아닌 Key Feature Map을 활용하여 Feature Map으로 복원하는 방법을 제안하여 데이터량을 더욱 감소시키고 성능을 더욱 향상시켰다.

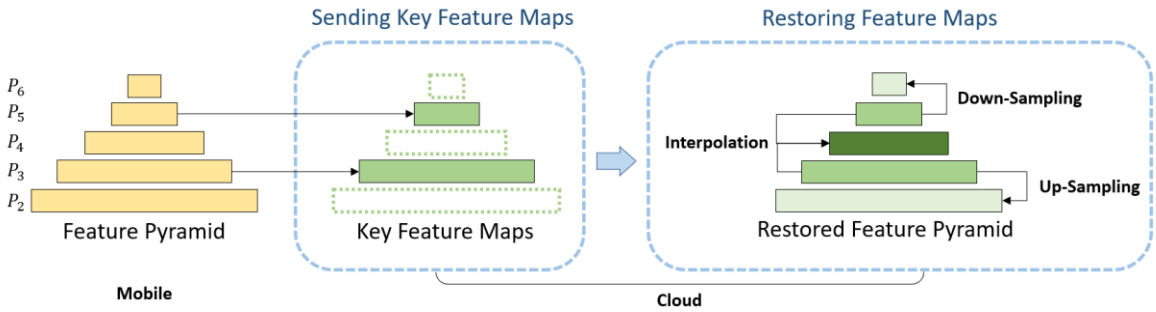
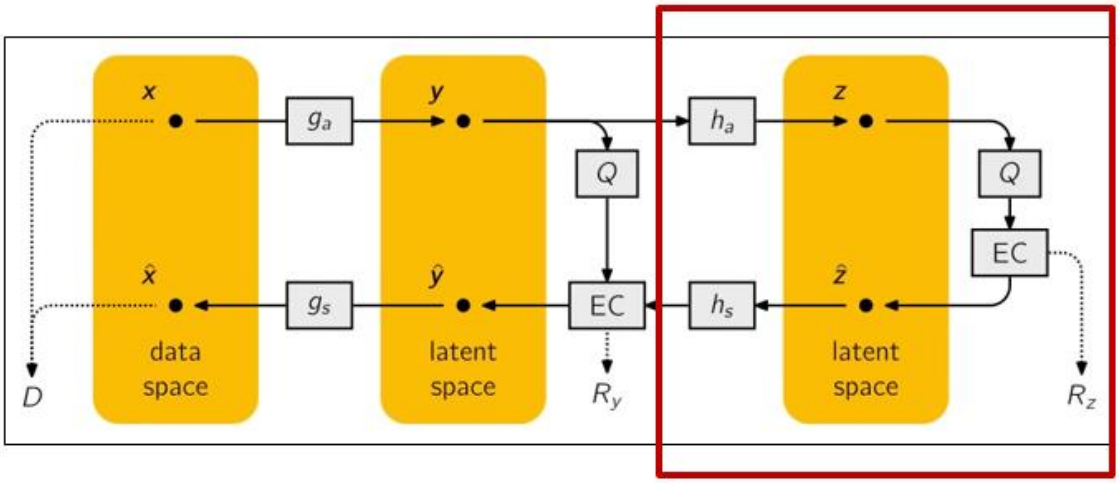


Figure 1 제안 방법

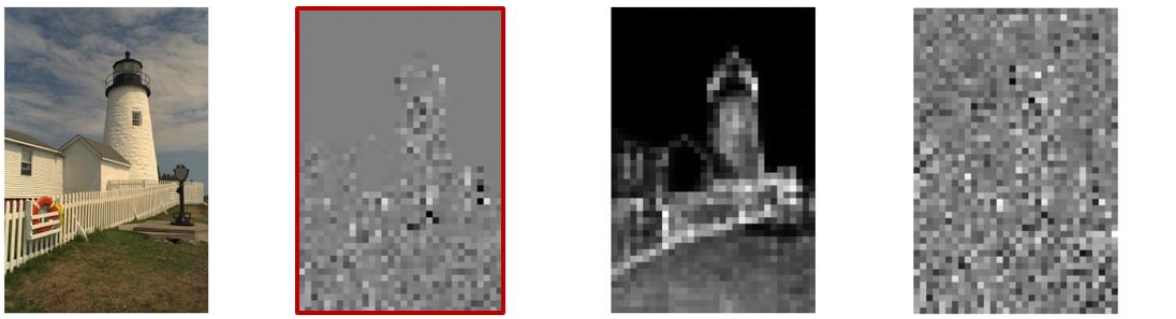
2. 관련연구

2.1 영상 압축

2.1.1 J.Ballé 모델



[그림1]



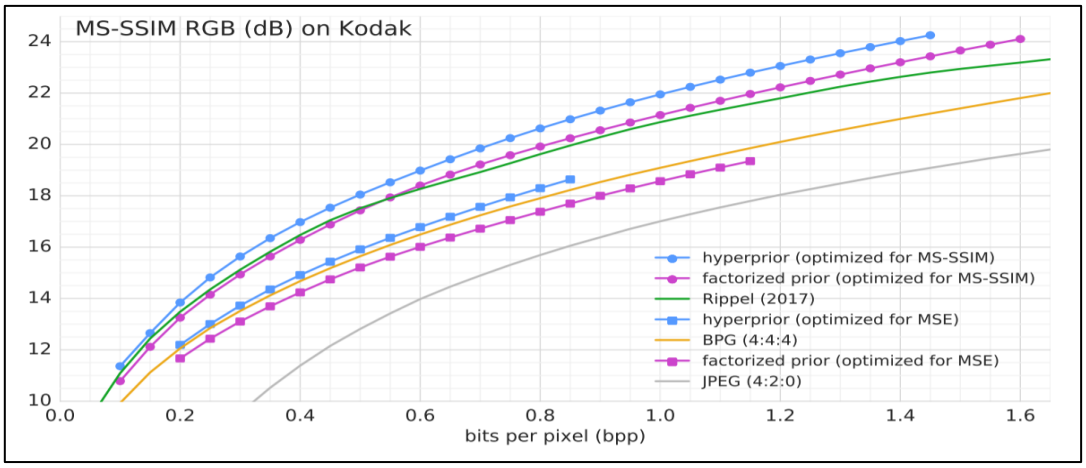
An image from the Kodak dataset

latent representation y of that image, learned by our factorized-prior model

standard deviations $\hat{\sigma}$ of the latents as predicted by the model Augmented with a **hyperprior**

latents y divided elementwise by their standard deviation

[그림2]

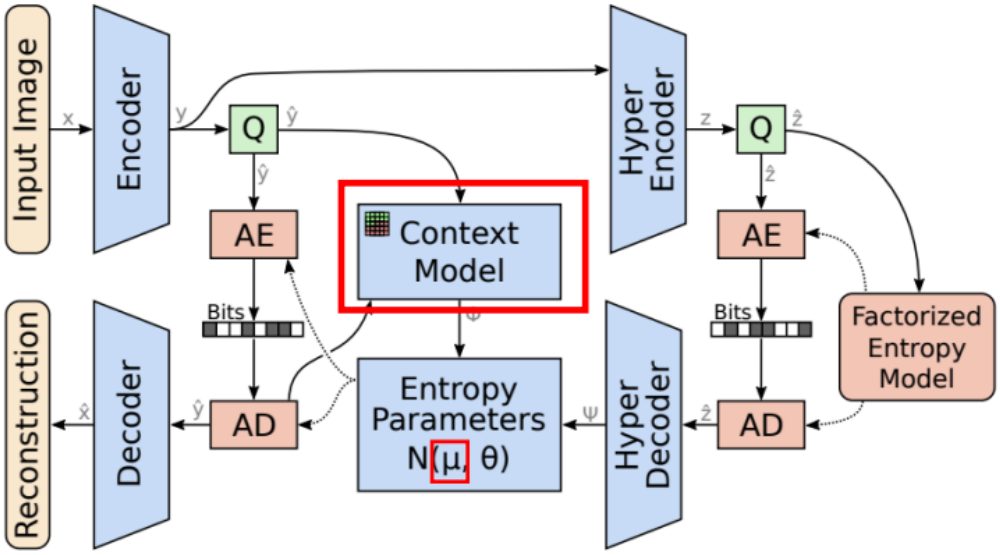


[그림3]

기존 학습 기반 압축에서는 y를 독립이라 가정하고 Entropy model을 통해 학습된 고정 분포를 모든 y에 대응시켰다. 그러나 Ballé는 그림2의 2번째 사진과 같이 edge와 textured 지역에서 y의 공간적 종속성을 확인할 수 있다고 생각하였고, 그림1의 오른쪽 모델을 추가하였다. 이를 hyperprior model이라 부르며, 이전 Entropy model을 이용한 고정분포에서 y가 아니라 z를 생성하고, 이 z를 통해 σ 들을 뽑아내서 각 y마다 평균은 0으로 같고, 서로 다른 분산을 가지는 가우시안 분포를 대응시켜서 그림2의 4번째 사진과 같이 공간적 종속성을 제거하였다. 결과는 그림 3과같이 hyperprior를 사용했을 때 그렇지 않은 경우보다 같은 bpp 대비 좋은 화질을 보여주었다. 이후 나온 학습 기반 압축 논문에서는 대부분 hyperprior를 사용하게 되었다.

2.1.2 D.Minnen 모델

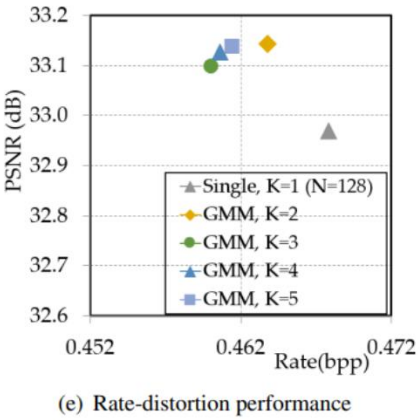
Minnen은 앞서 설명한 Ballé모델에서 두가지 큰 변화를 제시하였다. 첫 번째는 서로 다른 평균으로, hyperprior에서 분산 뿐만 아니라 평균도 뽑아내서 서로 다른 평균과 서로 다른 분산을 가지는 가우시안 분포를 대응시켜서 공간적 종속성을 제거하였다. 두 번째로는 Context model을 제안하였다.



[그림4]

Context model은 이전 정보를 이용하여 예측하는 모델로, 이미 디코딩된 정보를 이용하기 때문에 추가적인 비트량을 할당하지 않아도 된다는 장점이 있으며, 이전 정보를 사용하기 때문에 더 정확한 예측이 가능하게 된다.

2.1.3 Z Cheng 모델



(e) Rate-distortion performance

Cheng은 앞서 설명한 Minnen 모델에 또다시 두가지 큰 변화를 제시하였다. 첫 번째는 mixture Gaussian으로, 원래 사용했던 서로다른 평균, 분산의 single gaussian 분포에서 mixture gaussian 분포를 도입하였다. 여러 경우의 수로 실험했을 때 3개의 가우시안이 성능이 가장 좋았으며, 그 이상은 계산량 대비 효과가 없다고 말했다. 두 번째로 Attention module 이 추가되었다. Attention module은 convolution mask만 보는 것이 아니라 넓게 보고 가중치를 주는 방식으로, 높은 가중치를 가지고 있으면 멀리 떨어져 있어도 영향을 미치게 하는 방식이다. 이때 Softmax 대신 Sigmoid를 사용하여 어떤 정보가 가장 중요한지가 아닌 어떤 정보를 사용할지에 대해 결정할 수 있고, 전체의 가중합을 계산할 필요가 없어 계산량을 줄일 수 있다.

2.2 객체 인식

2.2.1 객체 인식 모델

객체 인식 모델은 classification과 localization을 진행하여 객체의 부류를 판담함과 동시에 객체의 위치를 경계 박스로 검출해내는 task이다.

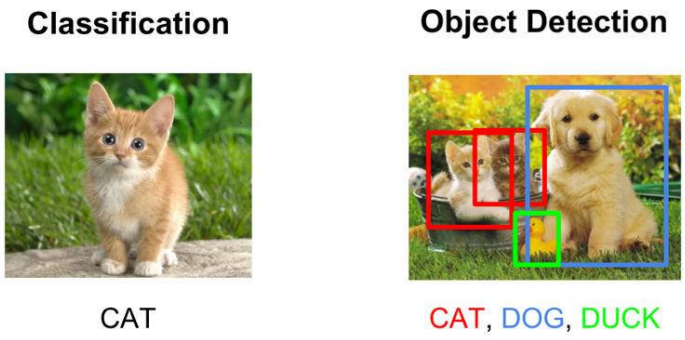


Figure 2 Object Detection Task (Left: Classification, Right: Object Detection)

딥러닝의 성능이 향상됨에 따라, 객체 인식 문제를 딥러닝을 사용하여 해결하려는 시도가 늘어났다. 객체 인식 모델은 작동 방법에 따라 크게 1 stage detector와 2 stage detector로 나눠진다. 객체 인식 모델은 Region Proposal과 classification 과정이 존재하는데, 1 stage detector는 이 두 과정이 한번에

이루어지는 반면, 2 stage detector는 region proposal 과정 이후에 classification 과정이 이뤄진다.

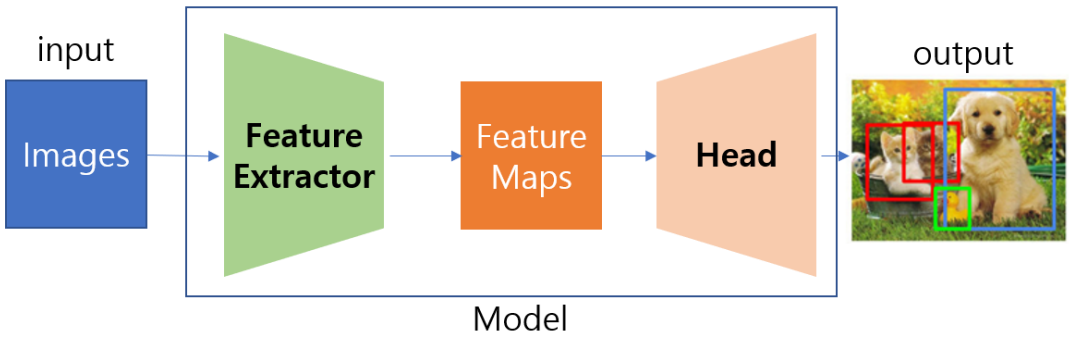


Figure 3 대표적인 Object Detecion 모델 구조

2.2.1.1 2 stage detector : R-CNN

2 stage detector는 이 두과정이 순차적으로 이루어지기 때문에, 1 stage detector에 비해 시간이 더 소요된다는 단점이 존재하지만, 일반적으로 1 stage detector보다 뛰어난 검출 성능을 지니고 있다. 그 중에서, R-CNN[5] 모델은 2 stage detector 초기 모델로, 컴퓨터 비전 문제에서 좋은 성능을 보여주는 CNN (Convolutional Neural Network)를 사용하여 객체를 인식한다. 하지만 R-CNN 모델은 Region Proposal, classification, boundary box를 찾는 과정이 독립적으로 존재하는 multi-stage pipelines 구조이기 때문에 물체 검출까지 걸리는 시간이 너무 오래걸리게 된다. 이를 개선하기 위해서 Fast R-CNN[6] 모델은 CNN 특징 추출부터 classification, bounding box regression까지 하나의 모델에서 학습을 진행하여 성능을 향상시켰다. 하지만, Region Proposal 과정은 R-CNN 모델과 동일한 Selective Search방법을 사용하여 RoI (Region of Interests)를 얻는 과정에서 병목 현상이 발생하게 된다. Faster R-CNN[7]은 RPN (Region Proposal Network)를 도입하여 RoI를 얻어낸다. RPN을 사용하면서 전과정을 GPU 상에서 동작이 가능하게 되어, Fast R-CNN에 존재하던 병목 현상을 제거했다. 기존 Selective Search 방법은 2000개의 RoI를 계산하는 반면, RPN을 통해 800개의 RoI를 계산하면서도 더욱 뛰어난 성능을 보여주었다.

2.2.1.2 1 stage detector : YOLO

YOLO[8]는 대표적인 1 stage detector 모델로, 지금까지 초기 모델인 YOLO[8]를 비롯하여 YOLOv2[9], YOLO3[10], YOLOv4[11], YOLOX[12]와 같이 후속 모델들이 계속하여 연구되어오고 있다. R-CNN[5] 모델들은 Region Proposal 에 대하여 연산을 진행해야 하는데, 하나의 이미지에서 적게는 800개에서 크게는 2000개의 RoI를 얻기 때문에 각 RoI에 대한 연산 과정이 필요하여 속도가 느리다. 그에 반해 YOLO는 전체 이미지를 분할해 해석하지 않기 때문에 빠르게 객체 검출이 가능해진다. 또한 후속 모델에서anchor를 사용하여 객체 검출 능력을 향상시키고, Feature Pyramid Network를 사용하면서 더욱 정보량이 많은 feature map에서 객체를 검출하면서 성능을 향상시켰다.

2.2.2 특징 맵 (Feature Map)

2.2.2.1 Feature Map

딥러닝 모델에서의 특징 맵은 입력으로부터 커널을 사용하여 합성 곱 연산을 통해 나온 결과이다. 일반적으로 딥러닝 모델에서 초기에 발생하는 특징 맵은 low level semantic information을 지니고 있고, 층에 깊어짐에 따라 특징 맵도 high level의 semantic information을 지니게 된다. 이는 강아지 이미지가 딥러닝 모델에 들어가게 되었을 때, 초기 특징 맵에서 대각선 혹은 색에 대한 정보를 가지고 있고, 깊은 층의 커널의 결과로 나온 특징 맵에서는 강아지의 얼굴, 몸통 등과 같은 정보를 지니고 있다는 것을 예시로 들 수 있다.

최근 객체 인식 모델에서는 정보량이 많은 특징 맵들을 얻어내기 위해 Feature Pyramid Network을 사용하여 low level semantic information을 지니고 있는 특징 맵에 high level의 특징 맵의 정보를 추가하여 새로운 high level의 특징 맵을 만들어내기도 한다.

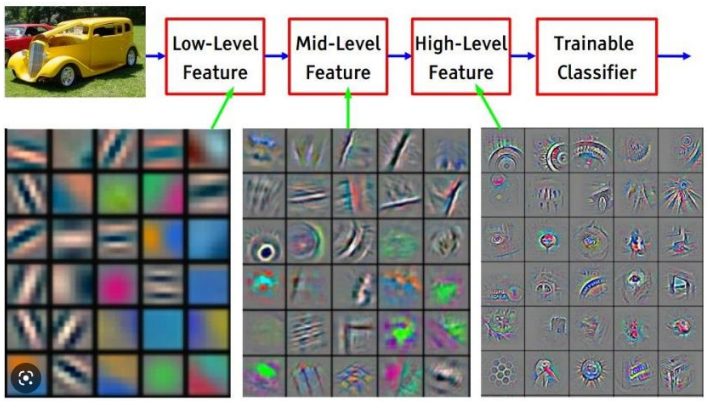


Figure 4 Feature Map 예시

2.2.2.2 Feature Pyramid

최근 Object Detection 모델들은 다양한 scale의 object를 검출하기 위해 2개 이상의 Feature Map을 사용하여 객체를 탐지해낸다. 이때 Feature Map들이 더 깊은 신경망에서 추출될수록 더 작은 사이즈의 크기를 가지고 있어 Feature Pyramid라고 불린다. 그 중 최신 모델들은 모든 Feature Map에서 high semantic information을 가질 수 있도록 하는 Feature Pyramid Network(FPN) 구조로 Feature Map을 얻는다.

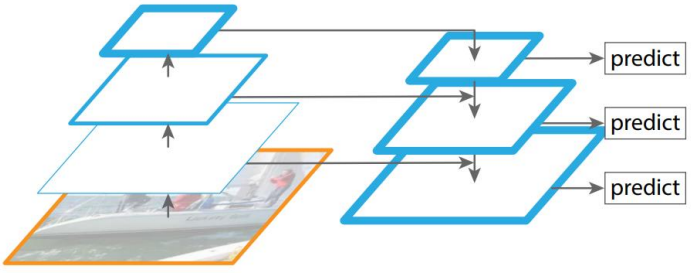
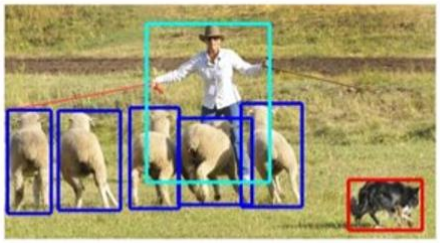


Figure 5 Feature Pyramid Network

2.2.3 Object Scale for Object Detection

Object Detection 모델에서 대표적으로 사용되는 데이터셋인 MS COCO는 이미지의 object scale이 모두 결정되어 있어, 각 scale 별에 대한 성능 또한 얻을 수 있다. 실제 세계에서는 다양한 크기의 객체들이 존재하는 이미지가 들어오기 때문에 모든 scale에 대하여 모델이 객체를 잘 탐지할 수 있어야 한다. 본 프로젝트에서는 이를 위해 모든 scale 별 Feature Map의 성능을 분석하였다.



```
Average Precision (AP):
  AP                % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50        % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75        % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small          % AP for small objects: area < 32^2
  AP^medium         % AP for medium objects: 32^2 < area < 96^2
  AP^large          % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1          % AR given 1 detection per image
  AR^max=10         % AR given 10 detections per image
  AR^max=100        % AR given 100 detections per image
AR Across Scales:
  AR^small          % AR for small objects: area < 32^2
  AR^medium         % AR for medium objects: 32^2 < area < 96^2
  AR^large          % AR for large objects: area > 96^2
```

Figure 6 Left: Real world 이미지 예시, Right: MS COCO 데이터셋의 object scale 구분 방법

3. 프로젝트 내용

3.1 영상압축 시나리오

3.1.1 YCbCr 모델 적용

앞서 설명한 기존 논문들은 모두 RGB 색공간의 이미지를 R/G/B 각각 분리해서 모델에 적용시키고, 다시 합쳐서 오차율을 계산하였지만, 우리는 YCbCr 색공간이 인간의 시각특성을 잘 반영하고 있다는 사실을 알고있다. 따라서 YCbCr이 학습될 수 있도록 기존 모델을 변형한 후, 학습된 결과를 다시 RGB 색공간으로 변경하여 기존 논문에서 나온 결과와 비교해보고 성능평가를 진행한다.

3.1.2 Cross Component Neural Compression

RGB는 서로 특성이 유사하지만, Y와 CbCr은 특성이 매우 다르기 때문에 NN을 두개 설계해야 한다. 이때, 민감한 Y는 distortion의 가중치를 크게 주고, CbCr은 낮게 주어서 주관적 화질이 좋아질 것이라는 가설을 검증한다. 또한 Y를 먼저 처리한 후 나온 정보를 CbCr의 NN에 활용가능한지 연구해본다.

3.2 압축 QUALITY EVALUATION

여러가지 이미지 품질 측정 방법을 사용해서 디코딩 노이즈가 얼마나 끼었으며, 사람이 느끼기에 원본과 얼마나 차이가 있는지 확인한다.

3.2.1 SSIM(Structural Similarity Index Map)

사람 지각 시스템은 이미지에서 구조 정보를 도출하는데 특화되어 있기 때문에 구조 정보의 왜곡 정도가 지각 품질에 가장 큰 영향을 미친다. 이러한 정보를 사용해서 원본 이미지 X와 디코딩된 이미지 Y의 밝기, 콘트라스트, 구조를 비교해서 이미지의 품질을 측정할 수 있다. 사용되는 식은 다음과 같다.

1. 밝기 비교 $l(x,\ y)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}$

2. 콘트라스트 비교 $c(x,\ y)=\frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2}$

3. 구조 비교 $\frac{x-\mu_x}{\sigma_x},\ \frac{y-\mu_y}{\sigma_y}$ Normalize $s(x,\ y)=\frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}$

4. 품질 맵 생성 $\mathrm{SSIM}(x,y)=l(x,y)^\alpha\cdot c(x,y)^\beta\cdot s(x,y)^\gamma$ | α=β=γ=1, C2=2*C3

$$\mathrm{SSIM}(x,y)=l(x,y)\cdot c(x,y)\cdot s(x,y)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}\frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2}\frac{\sigma_{xy}+C_2/2}{\sigma_x\sigma_y+C_2/2}$$

$$=\frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}$$

생성한 품질 맵을 이용해서 SSIM의 값을 구할 수 있고, 0~1 사이의 값을 가지며 1에 가까울수록 원본 이미지와 유사하다.

3.2.2 MS-SSIM(Multi-Scale SSIM)

MS-SSIM은 SSIM에 스케일 스페이스의 개념을 추가한 것이다. 이미지를 큰 스케일로 보면 작은 스케일보다 디테일이 줄고 전반적으로 흐릿해지는데, 이는 블러링과 유사하다고 볼 수 있다. 따라서 가우시안 블러링을 통해 여러 스케일을 만들고, 각 스케일마다 SSIM을 산출한 후 가중곱해서 최종 스코어를 얻는 방법이다.

Large Scale Low Scale Low Scale Large Scale

3.2.3 PSNR(Peak Signal-to-Noise Ratio)

PSNR은 최대신호잡음비로, 신호가 가질 수 있는 최대 전력에 대한 잡음의 전력을 나타낸 것이다. 주로 영상 또는 동영상 손실 압축에서 화질 손실 정보를 평가할 때 사용한다. PSNR은 단순히 원본 이미지와 왜곡된(디코딩된) 이미지 사이의 수치적 차이로 이미지 품질을 평가하기 때문에 사람의 시각 시스템을 잘 반영하지는 못한다. 사용되는 식은 다음과 같다.

1. MSE 계산 $MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,\ j)-K(i,\ j)]^2$

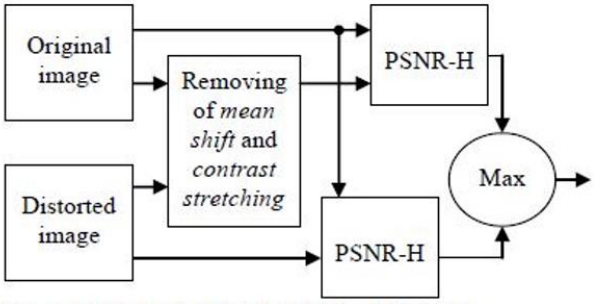
2. PSNR 계산

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

$$= 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

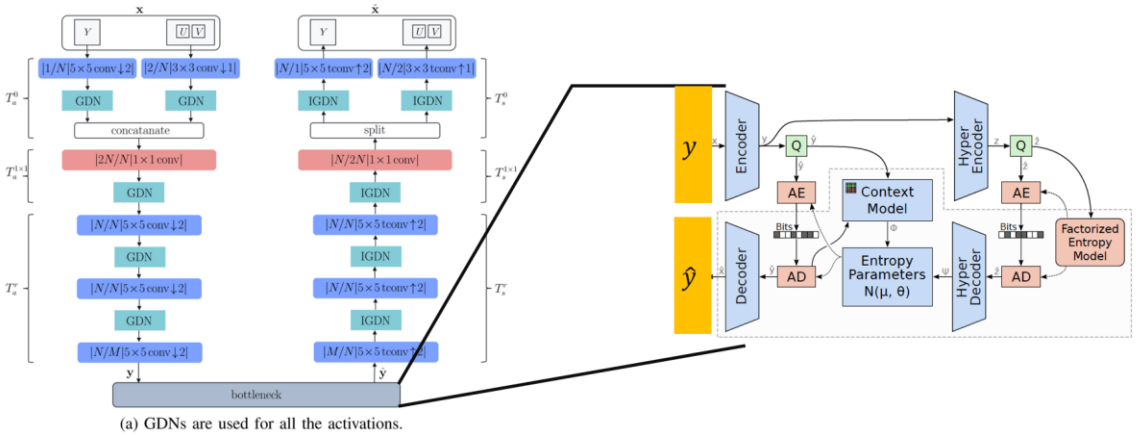
$$= 20\log_{10}(MAX_I)-10\log_{10}(MSE)\ \text{[dB]}$$

3.2.4 PSNR-HVS(PSNR-Human Visual System)



PSNR-HVS는 PSNR이 사람의 시각 시스템을 반영하지 못한다는 단점을 해결하기 위해 HVS를 적용한 것이다. 영상에 DCT변환 후 수행하게 되는데, 인간은 저주파신호의 왜곡(노이즈)에 더 민감하기 때문에 저주파계수에 큰 가중치를 두고 PSNR을 구한다.

3.3 제안된 압축 모델



(a) GDNs are used for all the activations.

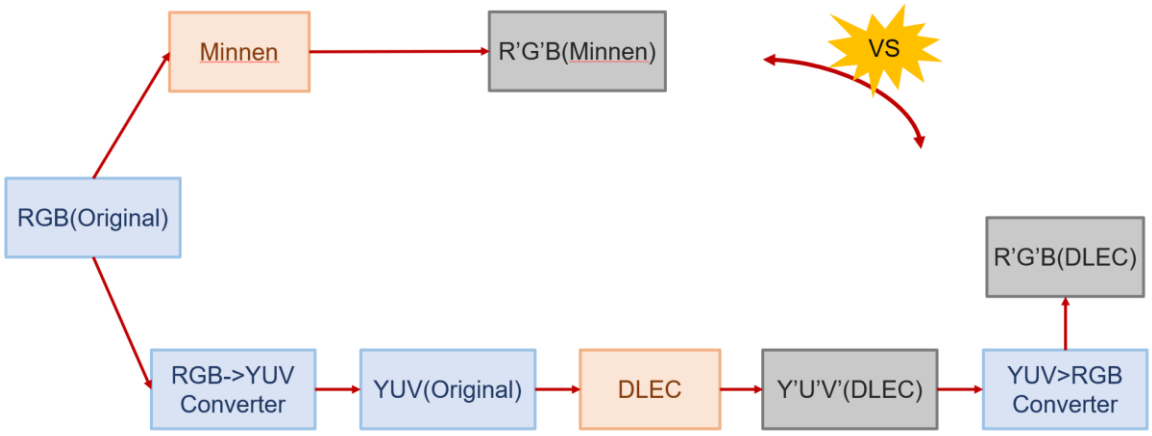
제안된 모델로, 원래 Minnen 모델에 YCbCr이미지를 입력하기 위해 이미지를 전처리 하는 과정을 붙혀놓았다. 기존 RGB 4:4:4 아키텍쳐를 YUV 4:4:4 포멧에서는 단순히 파라미터를 재학습해서 사용할 수 있지만, 앞서 말한 시각적 특성을 이용하기 위해 색차 정보에 다운 샘플링을 수행한 YUV 4:2:0 포멧에서는 간단하지 않고, 새로운 뉴럴네트워크 구조가 필요하다. 우선 Y와 UV를 채널 단위로 분할하고, 두 개의 네트워크를 학습시킨다. Luma(Y) 컴포넌트를 코딩하기 위해 단일 입출력 채널을 사용하고, Chroma(UV) 컴포넌트를 코딩하기 위해 2개의 입출력 채널을 사용한다. 이때, Luma 컴포넌트를 압축하는 것이 Chroma에 비해 더 중요하고 어렵기 때문에(더 많은 비트스트림이 필요) Luma에 5*5 컨볼루션 커널을 사용할 때, Chroma에 3*3 컨볼루션 커널을 사용해도 코딩 효율의 손실이 거의 없다고 할 수 있다.

Minnen 모델에 들어가기 전 모델은 세 개의 빌딩 블록으로 나눌 수 있다. 먼저 인코더 측에서 별도로 Luma채널과 Chroma 채널을 변환하는 첫 번째 단계로, Luma에 다운샘플링을 진행해서 두 브랜치의 출력 채널 차원을 동일하게 해준다. 이러한 분기 네트워크 구조는 서로 다른 콘볼루션을 사용할 수 있게 해주고, 네트워크의 초기 단계에서 Luma와 Chroma 채널의 서로 다른 신호 특성을 포착할 수 있

게 해준다. 두 번째로는 1*1 컨볼루션 레이어로, 이 레이어는 인코더 측에서 별도로 Luma 및 Chroma 정보를 전달하는 브랜치 네트워크의 출력을 결합하는 데 사용한다. 압축 관점에서 보면 1*1 컨볼루션은 luma 채널과 Chroma 채널 간의 상관관계를 이용할 수 있는 cross-channel 변환/예측 프로세스로도 볼 수 있다. 마지막으로 활성화 함수의 선택이 있다. 보통 압축에서는 활성화 함수로 GND을 많이 사용하는데, 이 실험에서는 PReLU를 사용한다. PReLU는 학습 가능한 파라미터 a를 사용해서 negative filter responses를 조정할 수 있다. 압축 목적에서 볼 때 PReLU는 ReLU, Leaky-ReLU보다 적합하다고 볼 수 있는데, ReLU는 음수를 0으로 만들기 때문에 양자화 및 엔트로피 코딩 단계 전에 Transform 단계에서 정보를 잃을 수 있으며, 계층 간에 서로 다른 비선형 함수를 갖는 것(학습된 a)이 Leaky-ReLU(고정된 기울기)보다 adaptation이 좋다고 볼 수 있기 때문이다. 또한 GND보다 복잡도가 상당히 낮기 때문에 PReLU를 사용하는것은 좋은 선택이라고 볼 수 있다.

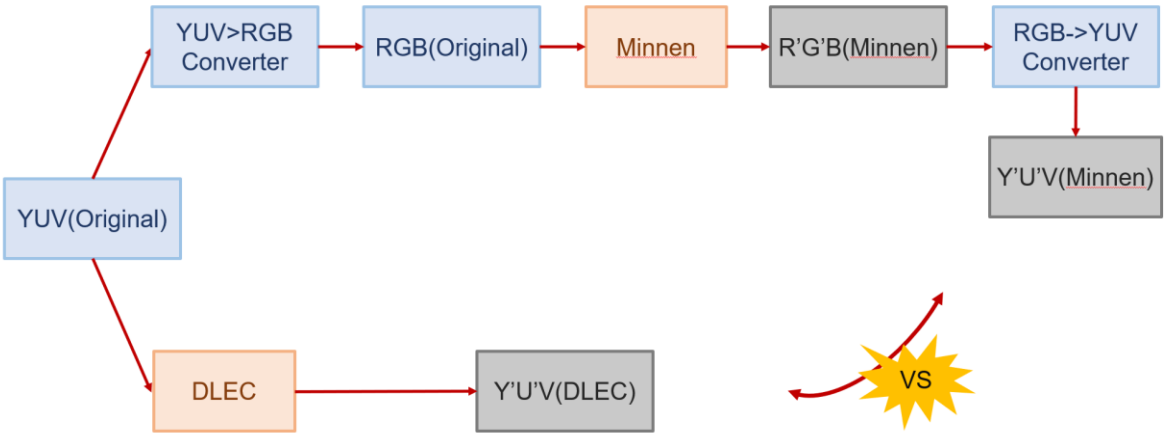
3.4 제안된 평가 방식

3.4.1 PSNR_RGB



위 그림은 PSNR_RGB평가 방식을 도식화 한 것으로, 앞서 설명한 PSNR은 비선형 연산이기 때문에 어떻게 값을 구하느냐가 중요해진다. 먼저 PSNR_RGB는 Minnen 모델과 제안한 DLEC모델의 출력(압축된 사진)을 RGB 색공간에서 비교하기 위해 사용한다. 이 때 계산은 RGB 이미지에 대해 PSNR을 한번에 계산하는 방식으로 했다. 이는 최근 논문에서 가장 흔하게 쓰이는 방식이기 때문에, 향 후 다른 논문의 결과와 비교할때도 편리하게 비교할 수 있다.

3.4.2 PSNR_YCbCR



위 그림은 PSNR_YCbCr평가 방식을 도식화 한 것이다. PSNR_YCbCr는 Minnen 모델과 제안한 DLEC 모델의 출력(압축된 사진)을 YCbCr 색공간에서 비교하기 위해 사용한다. 이 때 계산은 YCbCr 이미지에 대해 Y:Cb:Cr = 4:1:1로 구해서 6으로 나누는 방식을 사용했다. 이는 JPEG 등 표준화 단체에서 YCbCr 색공간 이미지에 대해 PSNR을 측정할 때 사용하는 방식이기 때문에 표준화된 평가 방식이라 생각되어서 사용하였다.

3.5. Object Detection 모델의 Key Feature Map 추출

본 프로젝트에서는 Object Detection 모델의 FPN에서 얻은 많은 layer의 Feature Map 중에서 객체 추정에 필수정인 Key Feature Map을 찾고자 했다. 그렇기 위해 3가지 실험을 통해 Object Detection 모델의 Key Feature Map을 찾아내고, 해당 Key Feature Map만을 사용해 Inference에 사용하고자 했다. 이후 Key Feature Map을 사용해 사용하지 않은 나머지의 Feature Map을 복구하는 방법을 제안했다.

본 프로젝트에서 진행한 실험은 다음과 같다.

1) Feature Map 열화 분석: 각 Feature Map을 독립적으로 열화시켜, object scale 별 성능 하락의 추이를 확인하고자 했다.
2) Feature Map 조합 영향도 분석: Feature Map 열화 분석을 통해 개별적인 Feature Map의 영향도를 파악하였다. 실험 1의 결과를 바탕으로 Feature Map 조합의 영향도를 분석하고자 했다.
3) Key Feature Map 분석: 연구 목적인 Feature Map의 정보를 최대한 보존하면서 데이터량을 최소한으로 줄이고자 영양도가 높은 Key Feature Map을 분석함.

3.5.1 실험 1 – Feature Map 열화 분석

Object Detection 모델에서 전달하는 Feature Map의 데이터량을 최소화하면서 모델의 성능을 유지하는 것이 목표이기 때문에, 각 Feature Map을 열화시켰을 때 성능 변화의 추이를 확인함으로써 각 Feature Map의 특징 및 전체 성능에 대한 영향도를 파악하고자 하였다. Feature Map을 열화시키기 위

해 Bilinear Interpolation 기법을 사용했다. Feature Map을 열화 정도를 k배로 하여 열화시켰다는 말의 의미는, 기존 Feature Map을 Bilinear Interpolation 기법으로 k배 축소시킨 후 다시 Bilinear Interpolation 기법을 활용하여 본래의 사이즈로 되돌렸다는 뜻이다.

열화 목표 특징맵	열화 정도	AP	△AP	Aps	△Aps	Apm	△Apm	Apl	△Apl
-	-	43.047	0	27.203	0	46.094	0	54.895	0
P2	x2	43.00378	-0.04322	27.10084	-0.10216	46.08591	-0.00809	54.89625	0.001246
P2	x4	42.89765	-0.14935	26.92749	-0.27551	46.07111	-0.02289	54.88798	-0.00702
P2	x8	42.36454	-0.68246	25.99015	-1.21285	46.09514	0.001144	54.90562	0.01062
P2	x16	41.66869	-1.37831	24.70684	-2.49616	46.0991	0.005097	54.91116	0.016164
P2	x32	41.34304	-1.70396	24.21728	-2.98572	46.10396	0.009961	54.90502	0.010024
P2	x64	41.20396	-1.84304	24.0514	-3.1516	46.08815	-0.00585	54.9047	0.009705
P3	x2	43.00837	-0.03863	26.95662	-0.24638	46.05906	-0.03494	54.8892	-0.0058
P3	x4	42.91058	-0.13642	26.96121	-0.24179	45.98406	-0.10994	54.89512	0.000121
P3	x8	42.53774	-0.50926	25.70558	-1.49742	46.00831	-0.08569	54.88475	-0.01025
P3	x16	42.21886	-0.82814	24.20138	-3.00162	46.1035	0.009503	54.89092	-0.00408
P3	x32	42.11595	-0.93105	23.85029	-3.35271	46.18104	0.087036	54.92633	0.031332
P4	x2	43.04927	0.002266	27.11645	-0.08655	46.08303	-0.01097	54.96765	0.072647
P4	x4	43.05118	0.004182	27.07498	-0.12802	46.0836	-0.0104	54.90498	0.009985
P4	x8	42.6961	-0.3509	26.94125	-0.26175	45.13915	-0.95485	55.08018	0.18518
P4	x16	42.40513	-0.64187	26.88186	-0.32114	44.36106	-1.73294	55.14009	0.24509
P5	x2	42.99913	-0.04787	27.19323	-0.00977	46.04383	-0.05017	54.68422	-0.21078
P5	x4	42.95382	-0.09318	27.19844	-0.00456	45.84055	-0.25345	54.66529	-0.22971
P5	x8	42.72308	-0.32392	27.19538	-0.00762	45.57339	-0.52061	53.90853	-0.98647
P6	x2	43.046	-0.001	27.204	0.001	46.098	0.004	54.892	-0.003
P6	x4	42.934	-0.113	27.19	-0.013	46.085	-0.009	54.576	-0.319

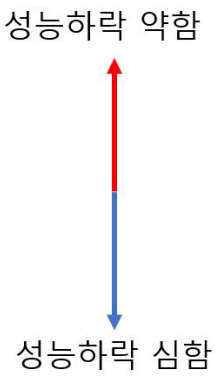


Figure 7 Feature Map 열화 실험 결과

Figure 7은 각 Feature Map의 열화 정도를 2배부터 최대(max)로 설정하여 실험한 결과이다. 이때 P2, P3, P4, P5, P6의 최대 열화 정도는 각각 64, 32, 16, 8, 4 배이다. 이때 각 수치의 배경 색깔의 의미는 짙은 빨간색일수록 성능 하락이 약함 (혹은 성능이 상승함)을, 짙은 파란색일수록 성능 하락이 심함을 의미한다.

Figure 7에서 Feature Map의 열화 정도가 커질수록 성능 하락이 커지는 것을 알 수 있고, 이는 모든 Feature Map에서 동일하게 나타남을 확인할 수 있다 (노랑 박스). Feature Map 열화 시 각 Feature Map마다 각기 다른 Scale의 Object에 큰 영향을 끼친다는 것을 확인할 수 있으며, 그 중 P2, P3 Feature Map 열화 시 성능 하락이 상대적으로 크다는 것을 알 수 있다 (빨강 박스).

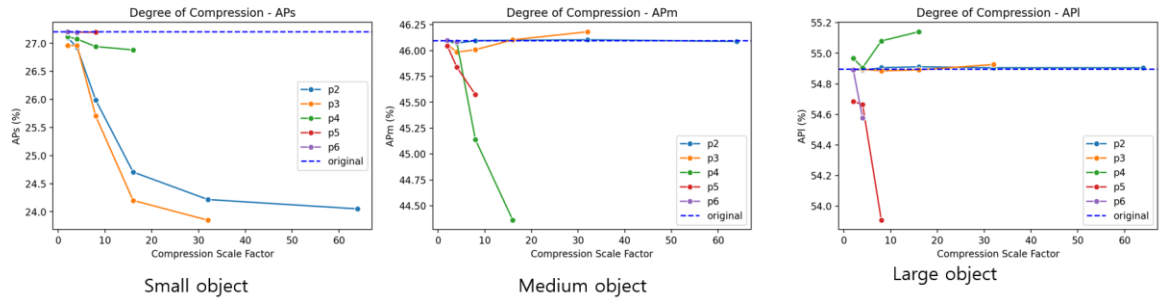


Figure 8 Feature Map 열화에 따른 성능 하락 정도. 왼쪽부터 small, medium, large object

본 프로젝트에서는 AP (모든 scale의 object에 대한)만을 고려하는 것이 아닌 모든 object scale (small object, medium object, large object)를 고려한다. Figure 2는 각 scale 별에 대한 Feature Map 열화 시의 성능을 그래프이다.

Figure 8를 통해 각 object별 영향을 끼치는 Feature Map을 명확히 확인할 수 있다. Small scale의 object같은 경우에는 Feature Map P2, P3의 영향을 가장 많이 받으며, medium object 같은 경우 P4, P5, large object 같은 경우에는 P5, P6의 영향을 가장 많이 받고 있다는 것을 확인할 수 있다. 그렇기 때문에 Key Feature Map을 얻을 때 모든 scale에서 성능 하락을 최소화하기 위해선 각 scale에 큰 영향을 끼치는 Feature Map의 조합을 적절히 선택하여 사용하여야 한다.

3.5.2 실험 2 – Feature Map 조합 영향도 분석

3.1.1 실험 1에서 각 Feature Map의 개별적인 영향도를 분석할 수 있었다. **실험 2**에서는 Feature Map을 조합하여 성능을 분석하여 Key Feature Map 조합을 찾기 위해 필요한 조건을 찾고자 했다.

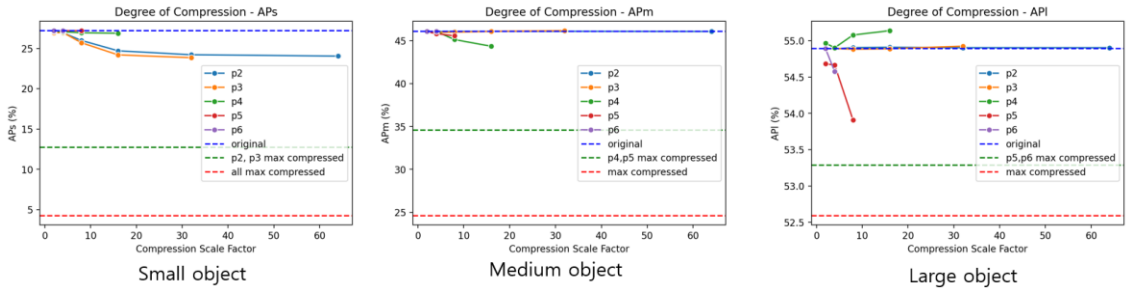


Figure 9 Feature Map 조합 영향도 분석

Figure 3은 object scale 별 영향도가 높은 두 Feature Map을 조합하여 열화를 최대로 진행한 후 성능을 측정했을 때에 대한 그래프이다. 각 그래프의 초록색 점선은 해당 scale object에 큰 영향도를 가지고 있는 Feature Map을 최대 열화시킨 후의 성능이며, 빨간 점선은 모든 Feature Map을 최대 열화했을 때에 대한 성능이다. 각 그래프의 초록 점선을 통해 알 수 있듯이, 각 scale 별 영향도가 높은 두 Feature Map 모두가 열화되었을 때에 대한 성능이 하나의 Feature Map만 열화되었을 때에 비해 상당한 성능 drop이 있다는 것을 확인할 수 있다. Small Object의 경우, P2 혹은 P3를 최대 열화시켰을 시 보다 약 10%의 성능 하락을, medium object 같은 경우에는 P4, P5 최대 열화보다 약 7~8%의 성능 하락을, P5, P6는 각각 약 1%, 0.5% 정도의 성능 하락이 있었음을 확인할 수 있다. 따라서 Key Feature Map을 선정하는 데에 있어서 같은 scale object에 큰 영향을 가지고 있는 [P2, P3], [P4, P5], [P5, P6] 조합을 동시에 열화시키는 것을 지양해야 함을 확인할 수 있다.

3.5.3 실험 3 – Key Feature Map 분석

실험 1, 2에서 얻은 정보를 바탕으로 물체 검출에 필수적인 Key Feature Map을 찾고자 한다. 즉, Feature Map의 정보를 최대한 보존하면서 데이터량을 최소한으로 줄이고자 영향도가 높은 Key Feature Map을 분석하였다.

Feature Map					성능
P2	P3	P4	P5	P6	APs (%)
O	O	O	O	O	27.204
c	O	O	O	O	24.051
O	c	O	O	O	23.85
c	O	c	O	O	23.726
O	c	c	O	O	18.518

O - Original
c – compressed (scale: max)

Figure 10 Feature Map 열화에 따른 성능 하락 정도

데이터량이 최소화하는 목적을 달성하기 위해, 데이터량이 가장 높은 Feature Map인 P2를 기준으로 Feature Map 영향도를 분석하여 **Figure 10**의 표로 작성하였다. 실험 결과, small object에 큰 영향을 끼치는 P2, P3외에도 Feature Map P4가 small object 탐지 성능에 큰 영향을 끼친다는 것을 확인할 수 있다. (**Figure 8** 빨간 박스) Figure 4의 5행을 보면 알 수 있듯이, 열화(압축)되지 않은 P2 Feature Map을 사용해 Inference를 한다고 하더라도, P4 Feature Map이 존재하지 않을 시에 약 9%의 성능 하락이 있다는 것을 확인할 수 있었다. 3행에서 P2와 P4가 열화되지 않았을 시에는 성능하락이 약 3.4%이 있다는 것을 통해, P2가 Key Feature Map으로 사용되기 위해서는 P4또한 필수적으로 Key Feature Map으로 선택되어야 한다. 그러나 **Figure 10**의 4행을 통해 알 수 있듯이, Feature Map P2, P4가 열화되어도 성능이 23.7%로 small object 검출에 대한 성능을 보장해주기 때문에 P3가 Key Feature Map으로 선택되는 것이 적절함을 확인할 수 있다.

Feature Map					성능 (%)									
P2	P3	P4	P5	P6	AP [50:95]	△AP [50:95]	AP [50]	△AP50	APs	△APs	APm	△APm	APl	△APl
O	O	O	O	O	43.047	-	63.66	-	27.203	-	46.094	-	54.895	-
c	O	c	O	c	40.49	-2.557	59.327	-4.333	23.726	-3.477	44.382	-1.712	54.582	-0.313
P3-up	O	P3-down	O	P5-down	41.504	-1.543	60.903	-2.757	24.652	-2.551	45.702	-0.392	55.017	0.122
P3-up	O	P5-up	O	P5-down	40.715	-2.332	59.834	-3.826	24.498	-2.705	43.564	-2.53	55.07	0.175
P3-up	O	P3,P5 Mix-1	O	P5-down	41.561	-1.486	60.952	-2.708	24.591	-2.612	45.899	-0.195	54.97	0.075
P3-up	**O**	**P3,P5 Mix-2**	**O**	**P5-down**	**41.62**	-1.427	**61.015**	-2.645	24.626	-2.577	**46.018**	-0.076	**54.978**	0.083
P3-up	O	P3,P5 Mix-3	O	P5-down	41.601	-1.446	60.96	-2.7	**24.678**	-2.525	45.913	-0.181	54.976	0.081

Figure 11 Key Feature Map만을 사용한 Feature Maps 복원 실험

실험 1에서 Feature Map P5는 medium object와 large object 모두에서 큰 영향을 끼치는 object임을 알 수 있었다. Figure 5에서 알 수 있듯이 Feature Map P4, P6가 열화되었을 당시에도 성능 하락이 급격하게 일어나지 않았다는 것을 확인할 수 있다. (실험 1에서 Feature Map P6가 최대 열화되었을 시 성능 large object에 대한 AP가 약 1% 하락했다.)

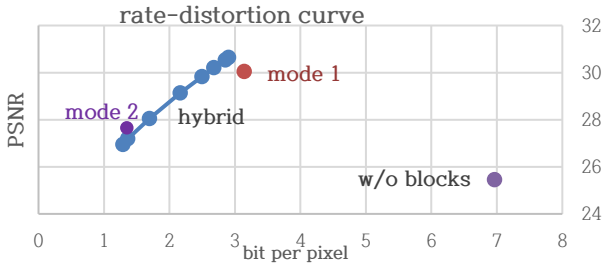
- P4 = a x P3-down + (1-a) P5-up
  - Mix-1: a=0.25
  - Mix-2: a=0.5
  - Mix-3: a=0.75

Figure 12 P4 Linear Interpolation 방법

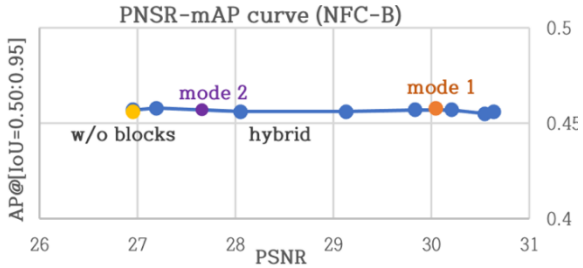
지금까지의 실험을 통해서 Feature Maps P2, P3, P4, P5, P6 중에서 P3와 P5을 Key Feature Map으로 선정할 수 있다. Figure 5의 초록 박스에 대한 실험은 열화되지 않은 Key Feature Map과 열화된 나머지의 Feature Map을 사용하였을 때에 대한 실험 결과이다. 본 프로젝트에서는 열화된 Feature Map을 다시 Bilinear Interpolation기법으로 복원하여 사용하는 것보다 Key Feature Map을 활용하여 Feature Map을 복원하여 사용였다. (Figure 5 빨간 박스) 기본적으로 P2와 P6 Feature Map은 각각 P3 Feature Map을 Up-scaling, P5 Feature Map을 Down-scaling하여 복원하는 방법을 생각할 수 있다. 그러나 P4 Feature Map같은 경우에는 선택된 두 Key Feature Map인 P3와 P5 모두에 접해 있기 때문에 P4 Feature Map은 P3와 P5을 Linear Interpolation을 사용하여 복원하고자 했다. **Figure 11**에서의 Mix에 대한 설명은 **Figure 12**에서 자세히 알 수 있다.

3.6 Vision task's robustness in feature map distortion

3.6.1 Motivation



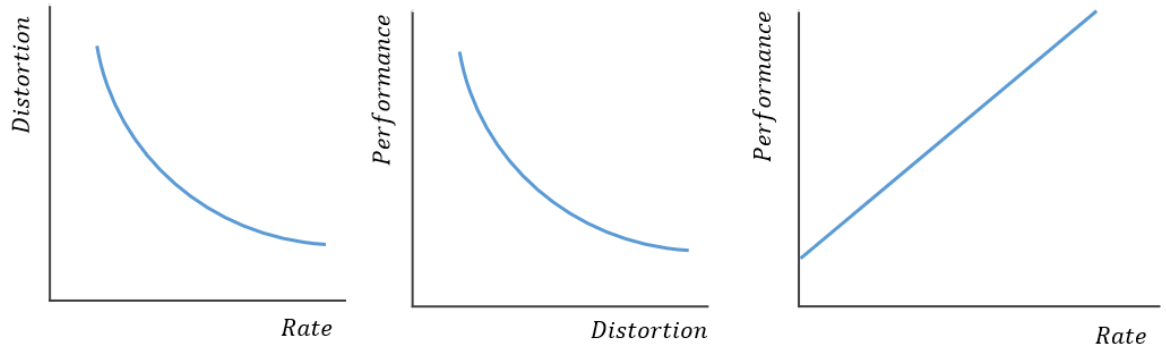
[그래프 1]



[그래프 2]

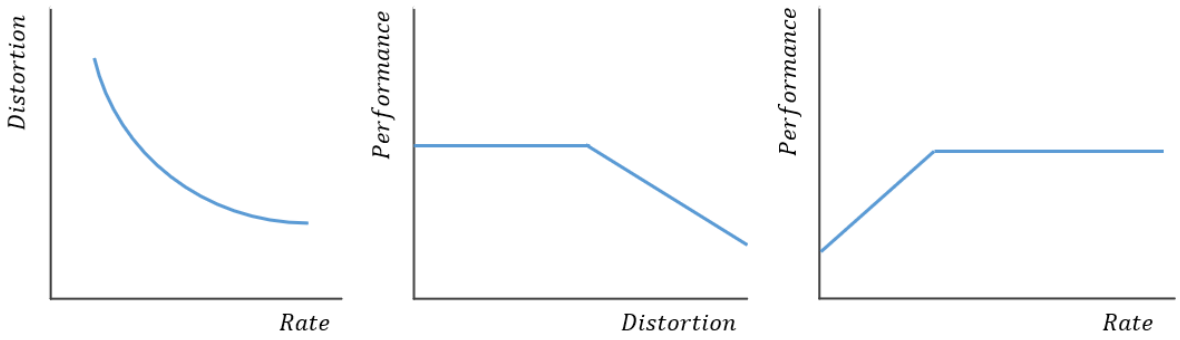
[그래프 1]은 코덱을 거친 후 특징맵의 왜곡을 영상의 왜곡을 측정하는 것과 같은 방식으로 PSNR로 나타낸 것이다. [그래프 2]는 [그래프 1]의 특징맵들을 이용한 객체 탐지 성능을 나타낸 것으로, 서로 다른 왜곡 수준을 가지는 특징맵들의 객체 탐지 성능에 차이가 거의 없다. 따라서 이를 통해 vision

task는 특징맵의 일정 수준의 왜곡에는 강인한 특성이 있을 것이라는 가설을 세울 수 있다. 만약 특징맵의 distortion 값이 0일 때 vision task의 performance가 가장 최적이라면, Rate-Distortion, Distortion-Performance curve와 Rate-Performance curve는 [case 1]과 같이 그려질 것이다. Rate는 압축한 특징맵의 비트스트림의 bit per pixel값, Distortion은 ground truth 특징맵에 대한 Mean Squared Error, Performance는 복원한 특징맵을 입력으로 하는 vision task의 mAP를 의미한다.



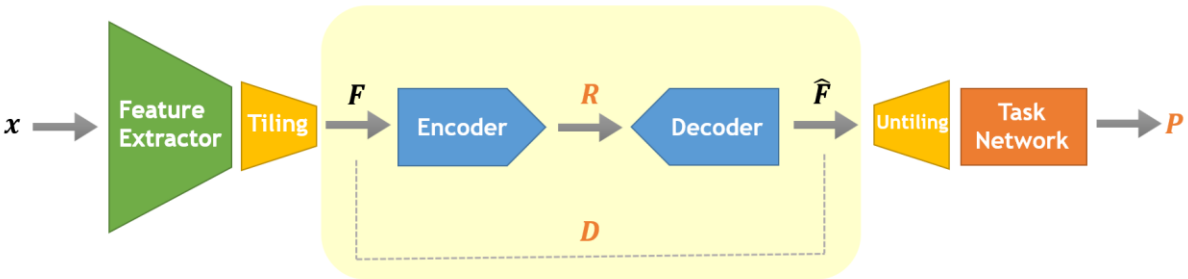
[case 1] distortion=0일 때 task performance가 최적일 경우

반면, task performance가 최적이게 하는 distortion 값이 0이 아닌 일정 threshold값 이하의 모든 값이라면, curve는 다음과 같이 그려질 것이다.

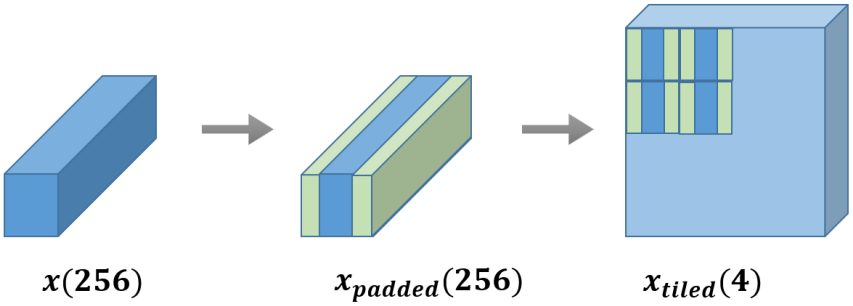


[case 2] distortion $\leq D_T$일 때 task performance가 최적일 경우

3.6.2 실험 과정



실험은 위의 다이어그램과 같은 과정으로 진행되었다. 입력 이미지 x는 OpenImages V6의 validation set을 사용했고, 각각의 이미지로부터 {p2, p3, p4, p5, p6} 스케일의 5장의 특징맵을 Feature Extractor FPN으로 추출했다. 학습과 테스트에는 {p2, p3, p4, p5} 스케일의 특징맵을 사용했고, 가장 작은 해상도의 p6특징맵은 task network에만 사용했다. NNVC codec에 입력하기 전 tiling 과정을 통해 256채널의 특징맵을 4채널의 특징맵 F로 펼친 후 학습 또는 테스트에 사용했다. 압축된 특징맵의 비트스트림 길이는 bit per pixel 단위의 R로 기록했으며, 복원 특징맵 $\hat{F}$와 F의 왜곡값 D는 mean squared error를 사용했다. 마지막으로 복원 특징맵을 256채널로 untiling후 task network에 입력해 vision task performance값 P를 출력한다. Task network는 detectron2의 instance segmentation 모델인 Mask R-CNN을 사용했으며, performance는 mAP@IoU0.5를 사용했다.



[Tiling 과정]

OpenImages V6의 이미지 데이터의 해상도가 각자 다르기 때문에 추출한 특징맵들의 해상도도 편차가 있어 batch 단위의 학습을 하기 전, cropping을 하는 대신 zero padding으로 해상도를 통일했다. p2, p3, p4, p5특징맵의 해상도를 각각 256*256, 128*128, 64*64, 32*32로 padding후, 가로, 세로 해상도를 각각 8배로 늘리고 채널수를 4로 줄이는 tiling을 적용했다. 복원된 특징맵은 반대 과정을 통해 256채널로의 untiling, zero padding을 다시 제거해 task network에 입력했다. 코덱 내부에서 특징맵을 latent로 표현하는데에 4채널 특징맵은 256채널 특징맵에 비해 큰 이점을 가졌다.

4. 프로젝트 결과

4.1 영상 압축

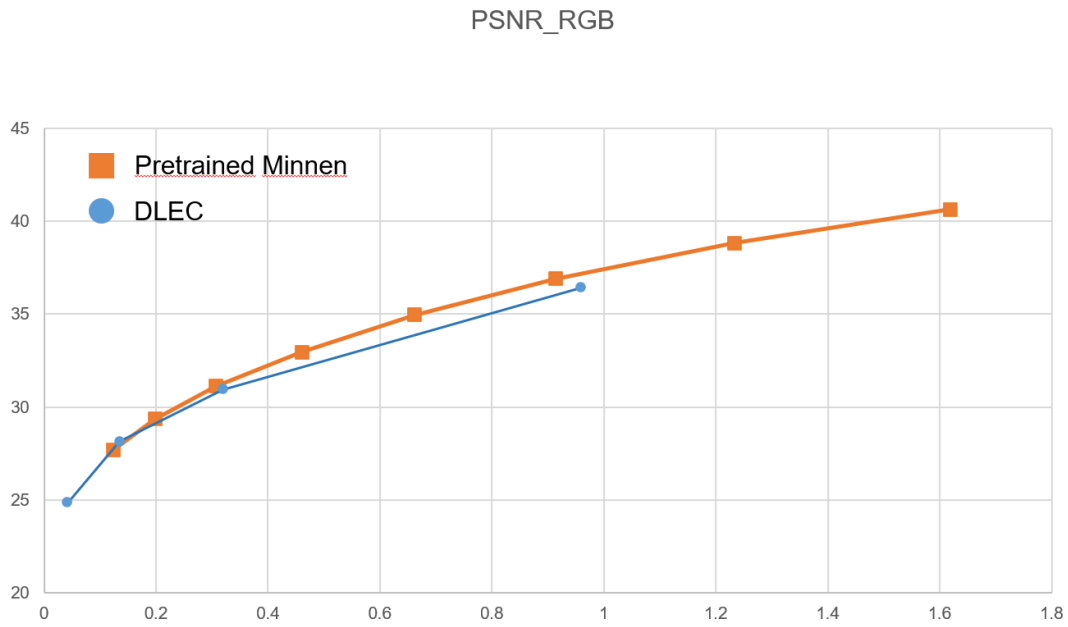
4.1.1 압축 성능 비교 – Minnen vs 제안된 모델

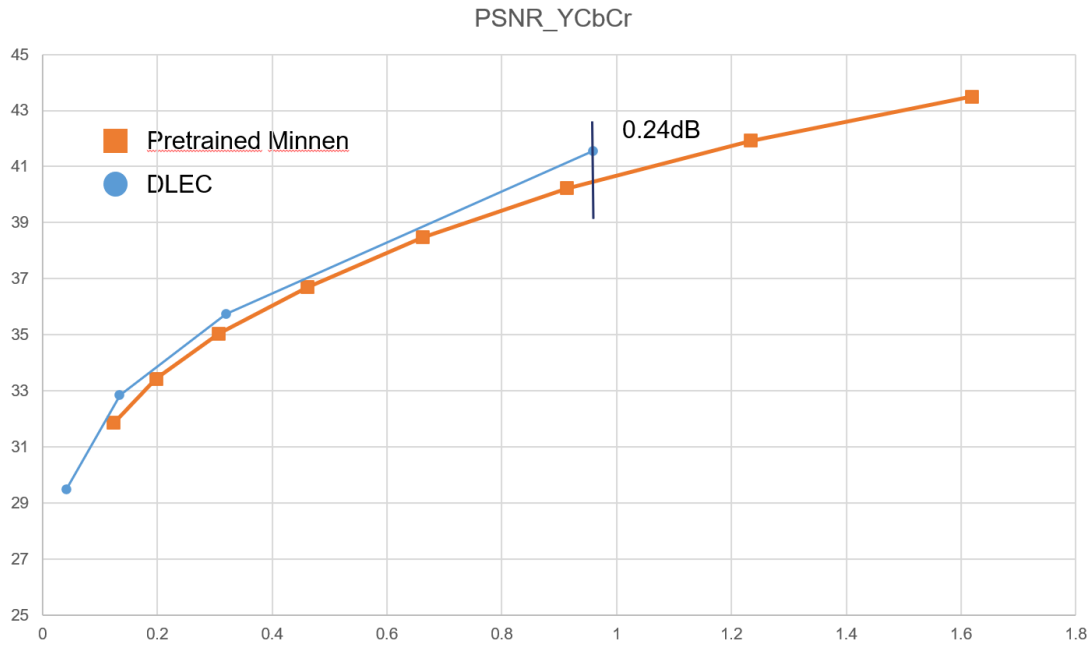
제안된 모델과 Minnen모델과의 압축 성능 비교를 진행하였다. PSNR_RGB에 대해서는 Minnen모델이 같거나 조금 더 좋게 나왔고, PSNR_YCbCr에 대해서는 제안된 모델이 Minnen 모델보다 측정된 bpp에 대해서 최대 0.24dB의 차이를 보였다. RD curve의 추세를 봤을 때, 퀄리티를 높게 해서 bpp가 높아지면 두 모델간의 차이가 더 벌어지는 양상을 보여줬기 때문에 추가적으로 High quality에 대해 학습을 진행하면 결과가 더욱 좋아질 것이라 생각한다.

아래 그림에서 PSNR_RGB와 PSNR_YCbCR에 대한 RD 커브를 볼 수 있으며, 압축된 이미지 결과도 볼 수 있다.

4.1.1.1 RD Curve

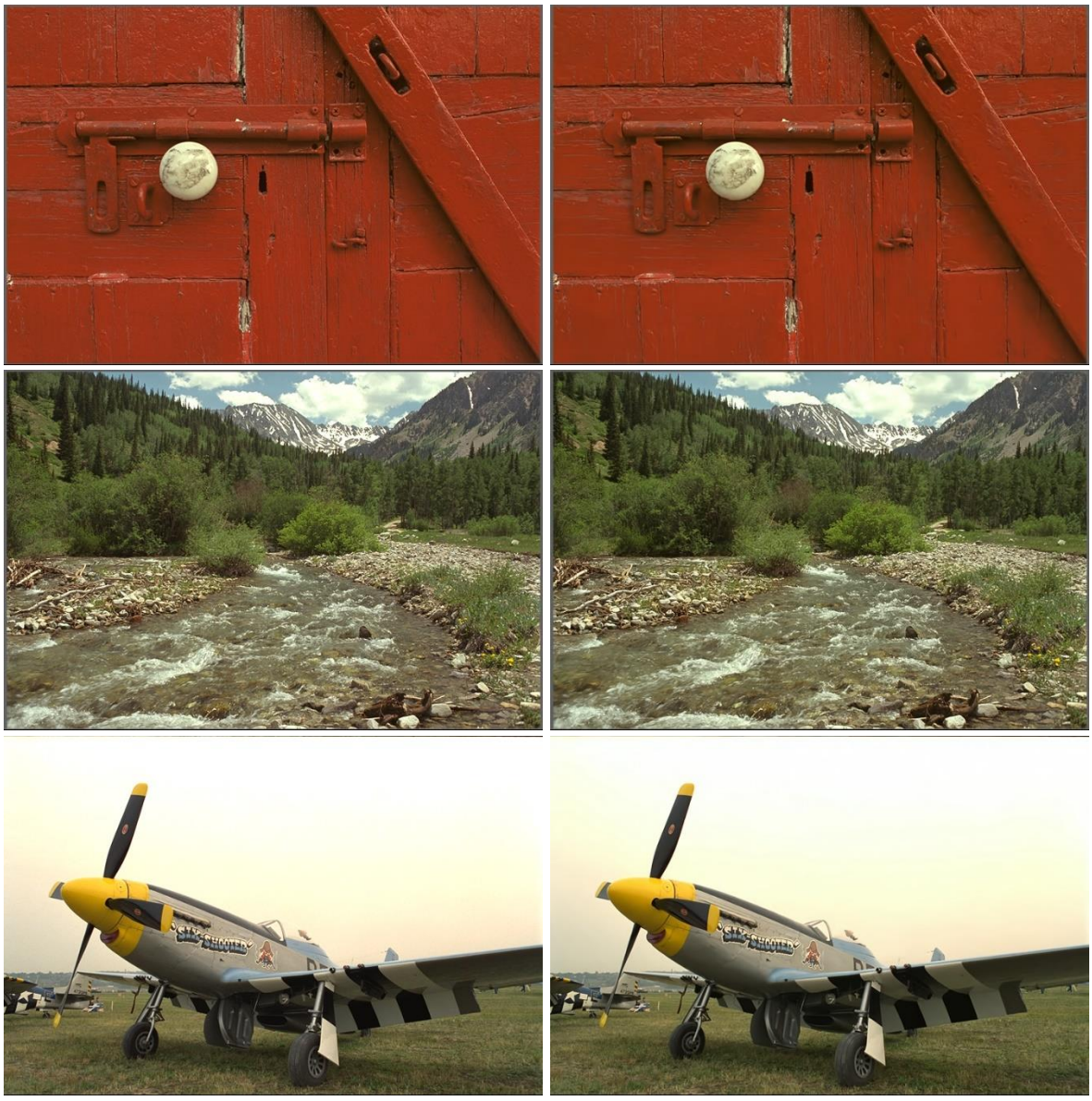
- **Dataset**
  - Train : vimeo
  - Test : Kodak24





4.1.1.2 압축된 이미지

- 왼쪽 : 원본 이미지
- 오른쪽 : 압축된 이미지
- bpp : 0.959



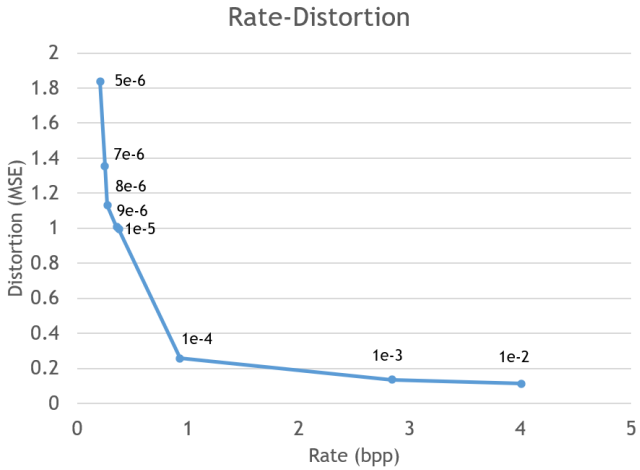
4.2 Object Detection 모델의 Feature Map 열화 분석

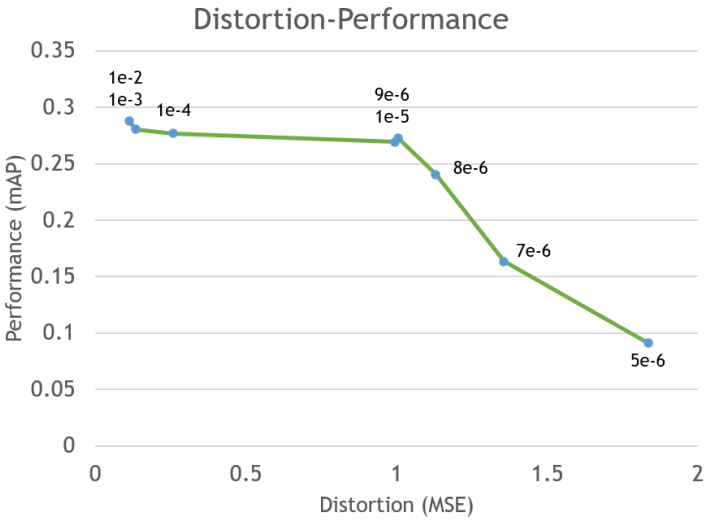
본 프로젝트에서는 Feature Map들을 독립적, 조합적으로 각 scale object에 주는 영향도를 파악하였다. 그 후 각 scale 별 영향도가 높은 Feature Map을 분석하여 이를 토대로 객체 인식에 필수적으로 사용되어야 하는 Feature Map인 Key Feature Map을 파악하였다. 이후 Key Feature Map 만을 활용하여 전달되지 않은 Feature Map들을 복원하여 모든 object scale의 성능 하락을 최소화하며 5.02배의 데이터량을 감소시킬 수 있었다. 이때 AP는 1.4% 하락하였으며, small object는 2.58%, medium object는 0.08%의 하락이, large object는 오히려 0.08%의 성능이 상승하는 결과를 얻을 수 있었다.

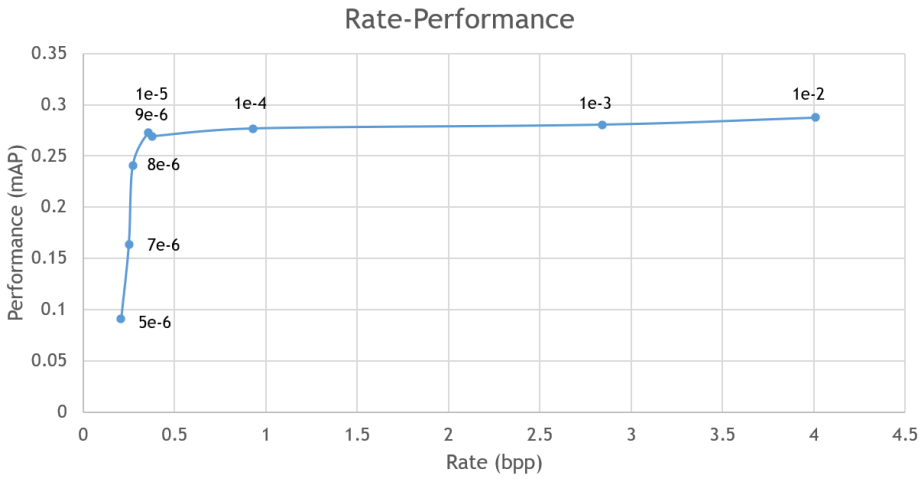
본 연구에서는 Object Detection 모델에서 효과적으로 데이터 전달량을 줄일 수 있는 방법을 제안했다. 하지만 이번 연구에서는 ResNext101 모델을 선정하여 해당 모델에 적합한 Key Feature Map을 선정하였다. 이후 연구에서 다양한 모델에 대한 Feature Map 영향도를 분석하여 Object Detection 모델에 general한 Key Feature Map 선택 알고리즘을 제안할 수 있을 것이다.

4.3 Vision task performance를 최적화하는 distortion threshold

실험은 256채널의 특징맵을 4채널로 펼쳐 가로, 세로 해상도가 각각 8배로 늘어나도록 tiling한 특징맵으로 코덱을 학습, 테스트했다. $Loss = lambda * MSE + bpp\ where\ lambda = \{1e-2, 1e-3, 1e-4, 1e-5, 9e-6, 8e-6, 7e-6, 5e-6\}$ 로 총 8가지 모델을 통해 얻은 Rate-Distortion, Distortion-Performance, Rate-Performance curve는 아래와 같다.







lambda	distortion	rate	performance
5.00E-06	1.836116	0.206	0.091078
7.00E-06	1.356125	0.25	0.16333
8.00E-06	1.130316	0.272	0.240556
9.00E-06	1.00531	0.356	0.272489
1.00E-05	0.994231	0.377	0.269231
1.00E-04	0.25708	0.928	0.276794
1.00E-03	0.134617	2.84	0.280385
1.00E-02	0.112621	4.008	0.287582

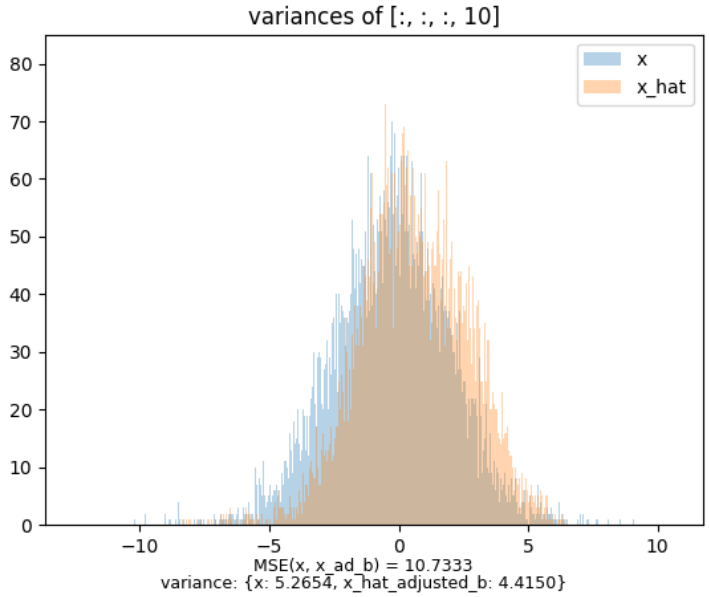
Rate-Distortion curve에서는 예상과 같이 Rate(bpp)와 Distortion(MSE)가 반비례 관계를 보임을 알 수 있었다. MSE=1에 가장 가까웠던 모델은 lambda=1e-5모델로, bpp값은 약 0.377이었다. Distortion-Performance curve는 가설을 뒷받침하는 강력한 근거로 볼 수 있다. Lambda=1e-5, 1e-4 모델에서 MSE는 0.99에서 0.25로 떨어지지만, vision task performance는 두 모델의 차이가 거의 없음을 알 수 있다. 이 현상은 더 큰 값의 lambda를 가지는 고해상도 모델들에서도 동일하게 관찰할 수 있는데, lambda=1e-2모델과 lambda=9e-6 모델은 Rate(bpp)가 10배 넘게 차이남에도 vision task performance는 거의 동일함을 확인할 수 있다.

Rate-Performance curve는 해당 현상을 더 명확하게 나타낸다. Performance gain은 이미 lambda=9e-6, 1e-5 부근에서 멈추지만 performance가 아닌 distortion을 최대한 최적화하는 고해상도 모델들은 performance gain이 없음에도 10배 이상의 bpp를 감수하며 최적화됐음을 알 수 있다. 이와 같은 global distortion threshold는 더 나아가 elementwise distortion thresholding을 이용해 element 단위로 불필요한 distortion 최적화를 막고, 그 대신 rate을 더 줄여 최종적으로 더 좋은 rate-performance curve를 기대할 수 있음을 보여준다.

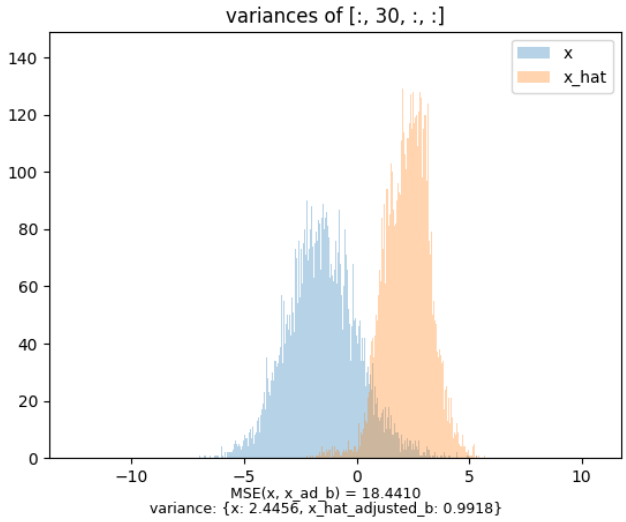
5. Appendix

5.1 Tiling Feature maps

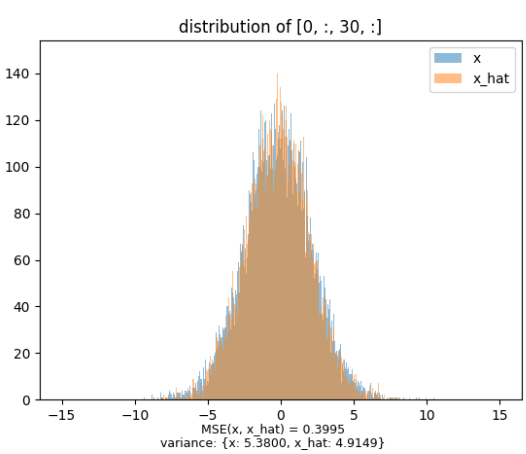
NNVC는 본래 3채널의 이미지/영상을 내부에서 256채널의 저해상도 latent로 변환해 압축하는 코덱으로, 256채널 특징맵을 입력으로 받았을 때는 latent의 표현력이 떨어져 복원된 특징맵의 채널별 확률분포가 ground truth와 괴리가 크고, 채널별 확률분포의 variance를 표현하지 못해 채널별 오차가 커 최종 vision task의 성능도 매우 떨어지는 결과를 보였다.

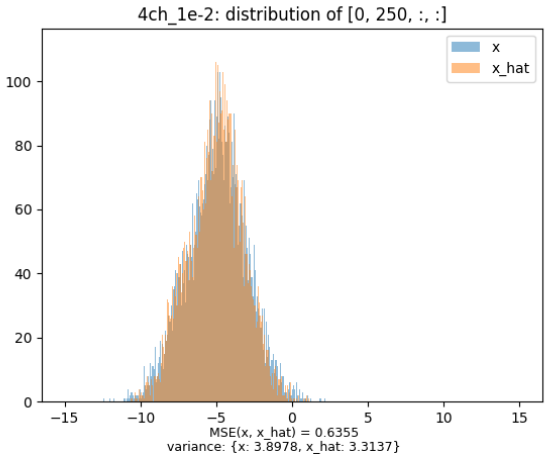


위는 width index를 10으로 고정시킨 후 비교한 ground truth와 reconstructed 특징맵의 확률분포이다. 채널별 확률분포가 corrupt 한 경우에도, global 확률분포 또는 width 나 height 의 index 를 고정시키고 모든 256 채널에 대해 비교한 확률분포는 ground truth 와 비교적 오차가 크지 않다.



위는 channel index 를 30 으로 고정시키고 비교한 두 특징맵의 확률분포이다. 파란색 ground truth 특징맵과 주황색 reconstructed 특징맵의 확률분포의 오차가 큰 것을 확인할 수 있다.



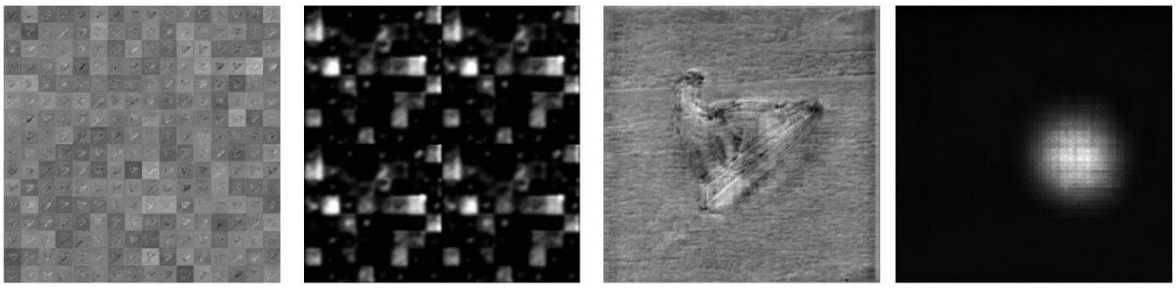


이에 따라 3채널 이미지/영상 입력과 유사한 입력을 만들어주기 위해 256채널의 특징맵을 4채널로 펼쳐 가로, 세로 해상도가 각각 8배로 늘어나게 tiling을 한 후 코덱을 학습, 테스트한 결과 위와 같이 global channel의 확률분포와 채널 하나하나의 확률분포가 모두 ground truth와 유사하게 복원됨을 확인했다. Distortion의 가중치 lambda가 충분히 큰 환경에서 NNVC는 매우 높은 복원 성능을 보였다. 1채널 또는 2채널로의 tiling도 가능하나, 특징맵을 추출한 이미지 데이터셋이 아주 낮은 해상도를 가지지 않는 이상 tiling된 특징맵의 해상도가 너무 커져 Out of Memory 에러가 발생해 practical하지 않았다.

5.2 Normalization

learned parameter를 사용하지 않고 fixed codec으로 특징맵을 압축하는 VTM은 VVC(H.266, Versatile Video Coding)를 기반으로 한 VCM 코덱이다. 일반적으로 이미지 또는 영상을 압축할 때 입력 데이터 값은 [0, 255]범위의 정수 또는 [0,1] 범위의 실수이므로, VTM에서도 사전에 알고 있는 특징맵들의 최솟값과 최댓값 범위를 이용해 압축 전 특징맵 데이터를 [0,1] 범위의 실수로 normalization 한다. 이와 같은 코덱에서는 데이터 값의 범위가 작아지더라도 양자화 스텝 크기를 직접 조정할 수 있기 때문에 normalization이 특징맵 복원 품질에 영향을 미치지 않는다. 하지만 NNVC와 같은 learned compression model은 직접 양자화 스텝을 조절하지 못하고, 대신 모델의 내부 채널수를 조정하거나 training loss의 distortion의 가중치를 조정하는 방법 등으로 QP를 조정한다. 따라서 VTM의 입력처럼 특징맵의 scale을 [0,1] 범위로 normalize시 element들 간의 variance가 매우 작아지고, 코덱에서는 작아진 variance를 잘 인식하지 못해 아래와 같이 학습이 잘 이루어지지 않는다. [0,1] 범위의 특징맵을 복원했을 때는 blurry한 특징맵을 얻게 되고, 이를 task network에 입력하기 위해 denormalize 과정을

거치면 blurry한 특징맵이 saturated 되는 현상이 일어난다. 1, 3번째는 ground truth 특징맵의 전체 채널을 타일링해 나타낸 모습과 특정 채널을 확대해 나타낸 모습이고, 2, 4번째는 복원된 특징맵의 모습이다.



작아진 variance scale을 코덱이 잘 인식하지 못하는 문제는 training loss에서 distortion의 가중치를 높여도 해결되지 않았다. normalize하지 않은 특징맵은 distortion의 가중치가 1e-4, 1e-3수준에서도 매우 높은 복원 품질을 보였지만, normalize한 특징맵은 동일 모델에서 distortion의 가중치를 100, 1000까지 높여도 위와 같은 현상이 일어났다.

6. 참고문헌

[1] Ballé, Johannes, et al, "Variational image compression with a scale hyperprior", 2018

[2] D.Minnen, et al, " Joint Autoregressive and Hierarchical Priors for Learned Image Compression", 2018

[3] Zhengxue Cheng, et al, " Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules ", 2020

[4] Ballé, Johannes, et al, "Efficient Nonlinear Transforms for Lossy Image Compression", 2018

[5] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation."Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[6] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation."Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[7] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks."Advances in neural information processing systems28 (2015).

[8] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection."Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[9]Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger."Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[10]Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement."arXiv preprint arXiv:1804.02767(2018).

[11]Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection."arXiv preprint arXiv:2004.10934(2020).

[12]Ge, Zheng, et al. "Yolox: Exceeding yolo series in 2021."arXiv preprint arXiv:2107.08430(2021)
[13] Hilmi E. Egilmez, et al, “Transform Network Architectures for Deep Learning Based End-to-End Image/Video Coding in Subsampled Color Spaces”, IEEE Open Journal of signal processing, 2021