‘let there be color’
그곳에 색이 있으라

자동채색 모델

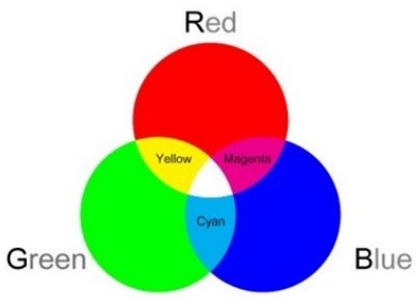
색구분

컴퓨터 컬러 이미지

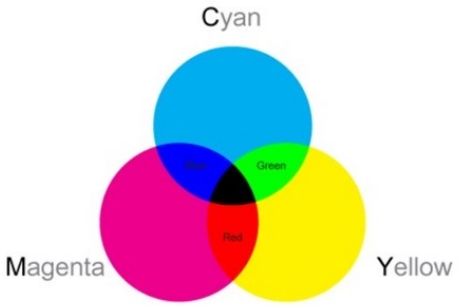
크게 RGB, CMYK, Lab 방식이 있음

RGB : 빛의 삼원색
CMYK : 잉크의 삼원색
주로 인쇄용 작업에 사용됨

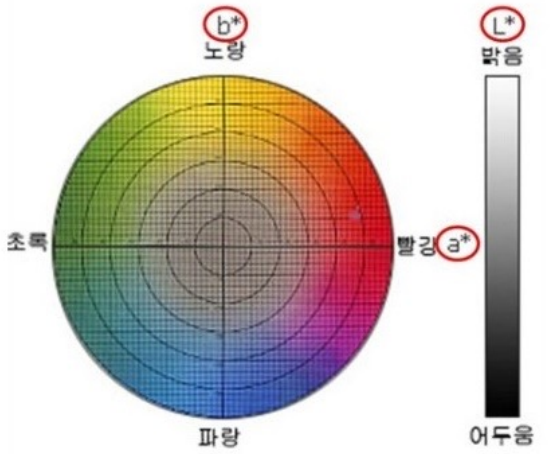
red, green, blue

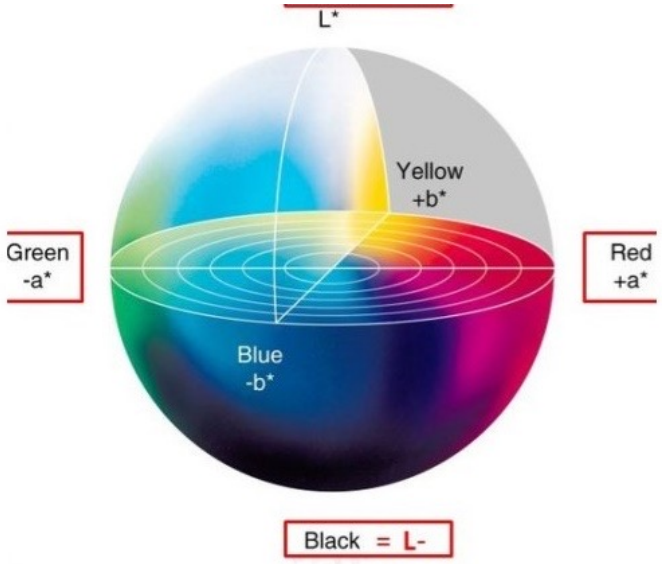


cyan(청색), magenta(적자색), yellow(노랑색), black(검정색)



Lab
구성 : red, yellow, blue, green
L은 명도, a는 red, green을 그리고 b는 blue와 yellow를 표현
장점
: 모니터, 프린터에 좌우되지 않는 독립적인 방법으로 색상 구현
: RGB와 CMYK의 범위를 모두 포함
: 색상과 빛을 분리하기 때문에 채도 변화 없이 색상대비와 밝기 조절





Lab방식이 ground truth와 비교했을 때 가장 유사하게 색이 복원

Ground truth RGB YUV L*a*b*

채색을 학습하는 방법

scribble-based

(방법) 사용자가 미리 사진에 어떤 색을 칠할지 정해주면,
학습 후 적절하게 채색하는 방법

(단점) user의 입력에 대해서 강한 의존성

[Levin+ 2004]

Reference image-based

(방법) 입력과 비슷한 레퍼런스 이미지를 통해 학습 후
적절한 채색을 하는 방법

(단점) 입력과 비슷한 reference 이미지를 찾기 위해
사람이 적절한 전처리를 수행해야 한다는 것에 있어
좋지 않음

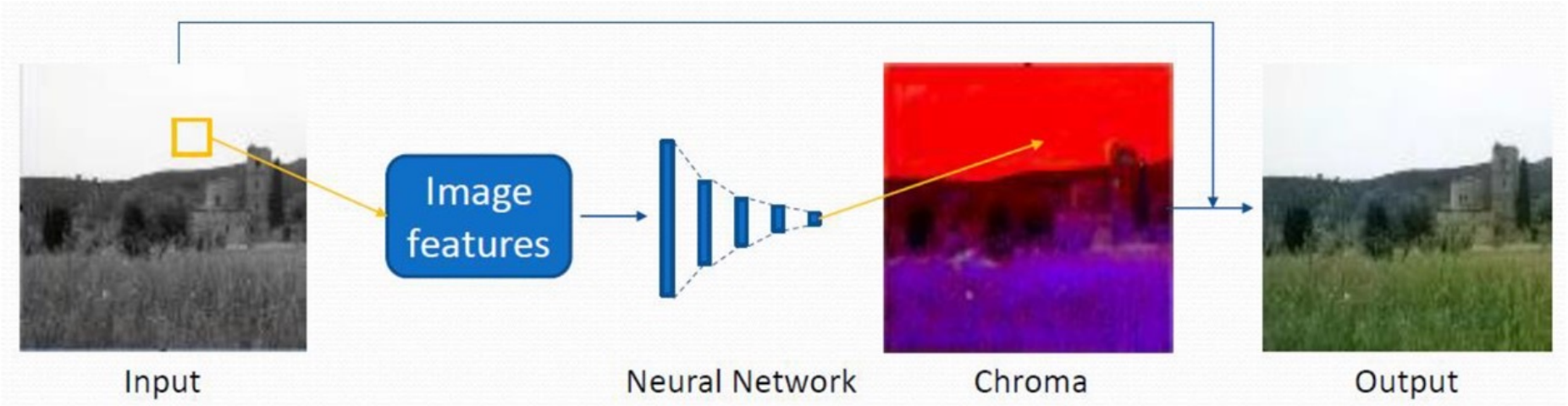
[Gupta+ 2012]

채색을 학습하는 방법

automatic colorization

(방법) 입력 이미지를 각각의 patch들로 나누고, 그 patch를 입력으로 하는 작은 뉴럴 네트워크를 통해 feature를 얻음
이렇게 얻어진 feature로 최종적으로 채색함

(단점) 이미지를 구분하기 위한 알고리즘이 제데로 작동하지 않으면 채색성능이 떨어질 수 있음



컨셉

2016년 이전의 많은 이미지 colorization 방법과는 다르게 fully automatic으로 이미지의 색을 자동으로 칠해준다는 점이 의미 있는 결과

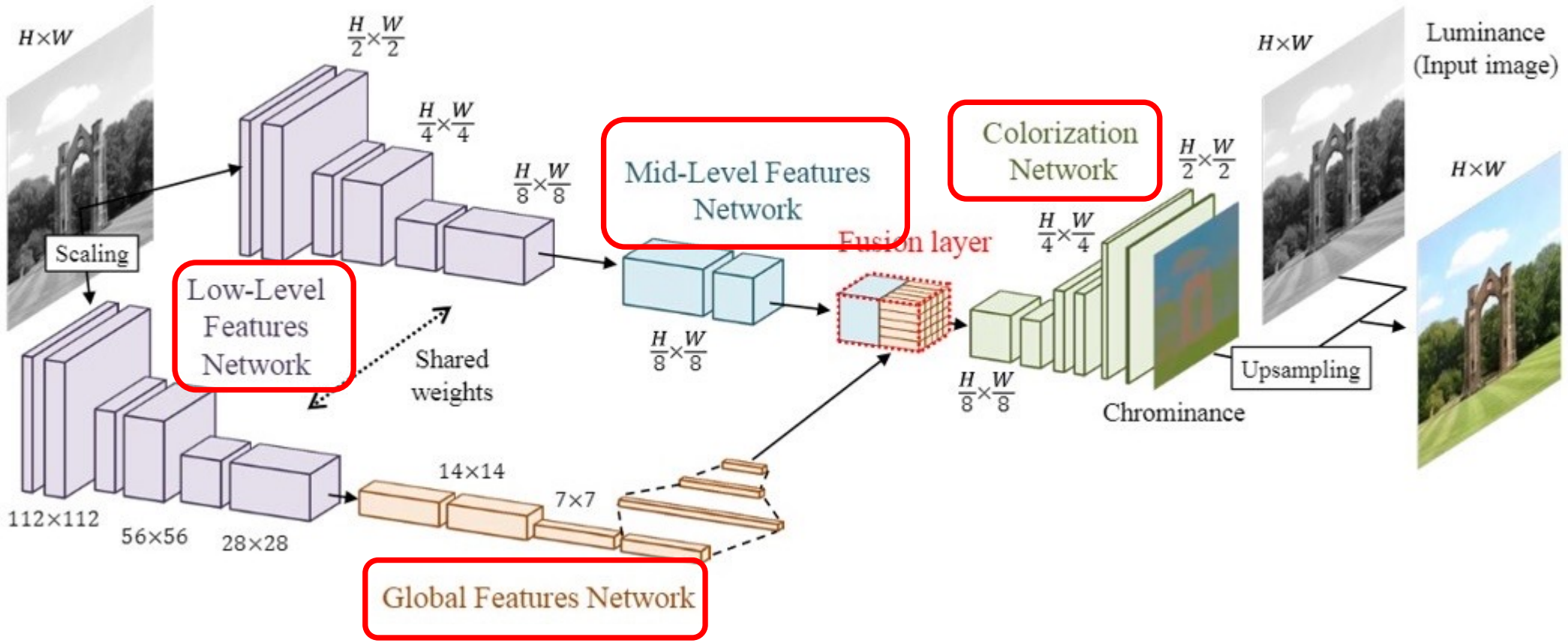
두가지 네트워크의 조합

이 논문에서는 두 가지의 네트워크를 사용함
① Global image의 사전 정보를 학습하는 네트워크
② local image의 작은 patch들을 학습하는 네트워크

최종적으로 이 두 가지 네트워크 조합해서
Colorize 자동으로 실행
*논문에서는 두 네트워크를 조합하는 방법을 fusion layer라고 함

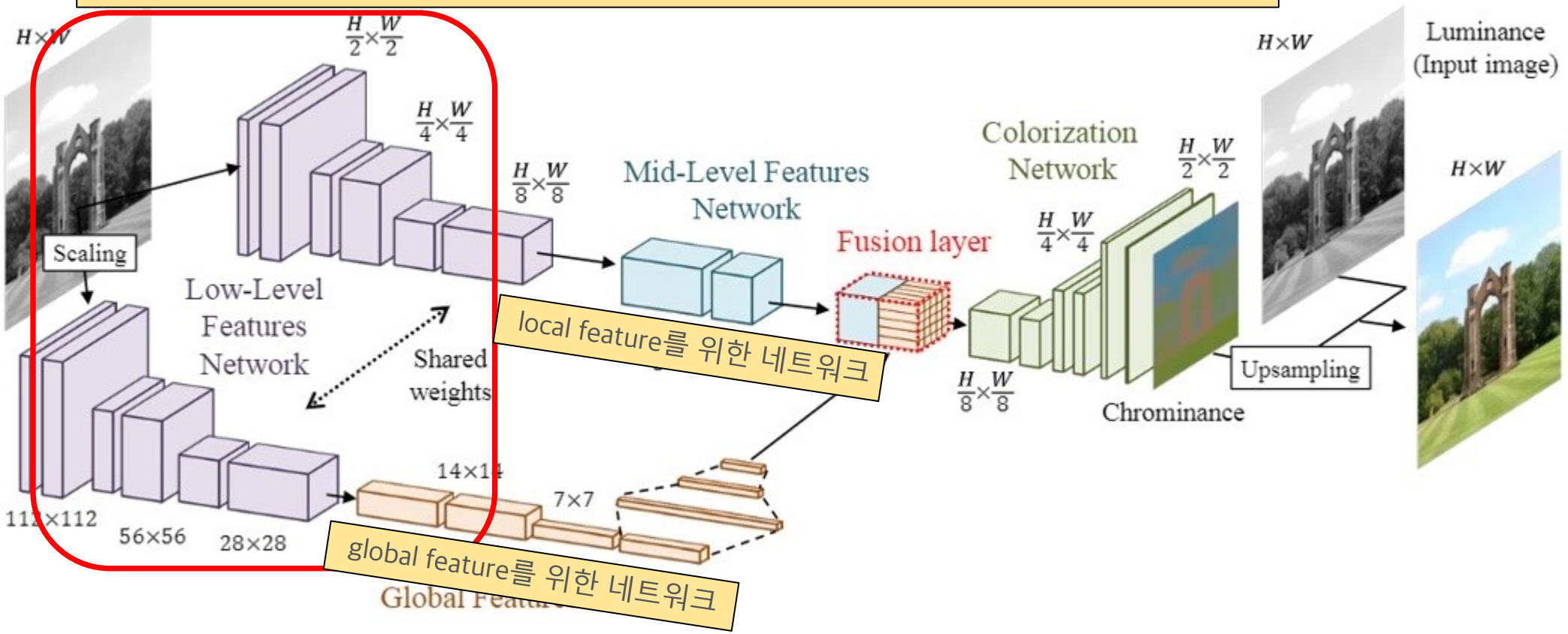
모델 아키텍쳐

4개의 네트워크로 구성

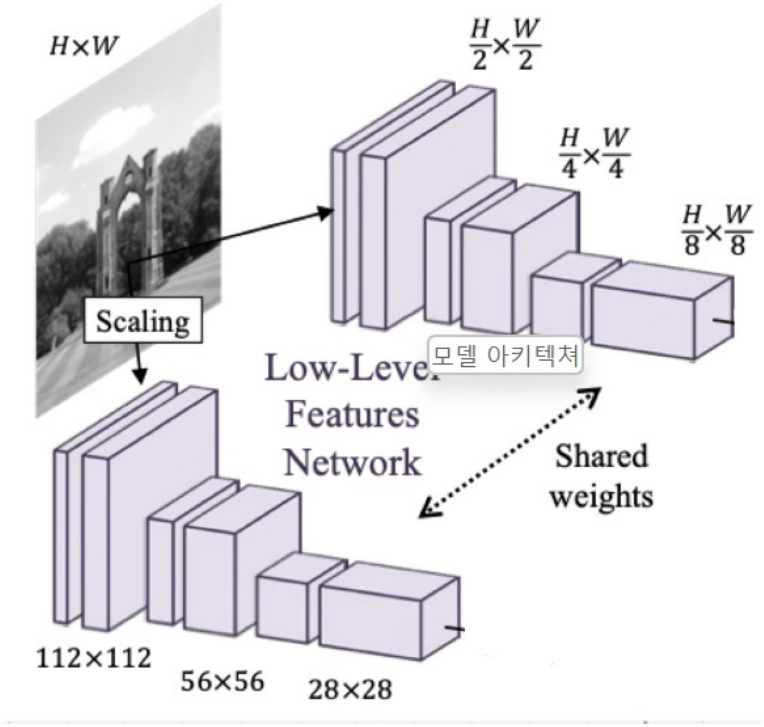


모델 아키텍쳐

low-level feature network : 이미지의 경계나 코너같은 lower features을 학습



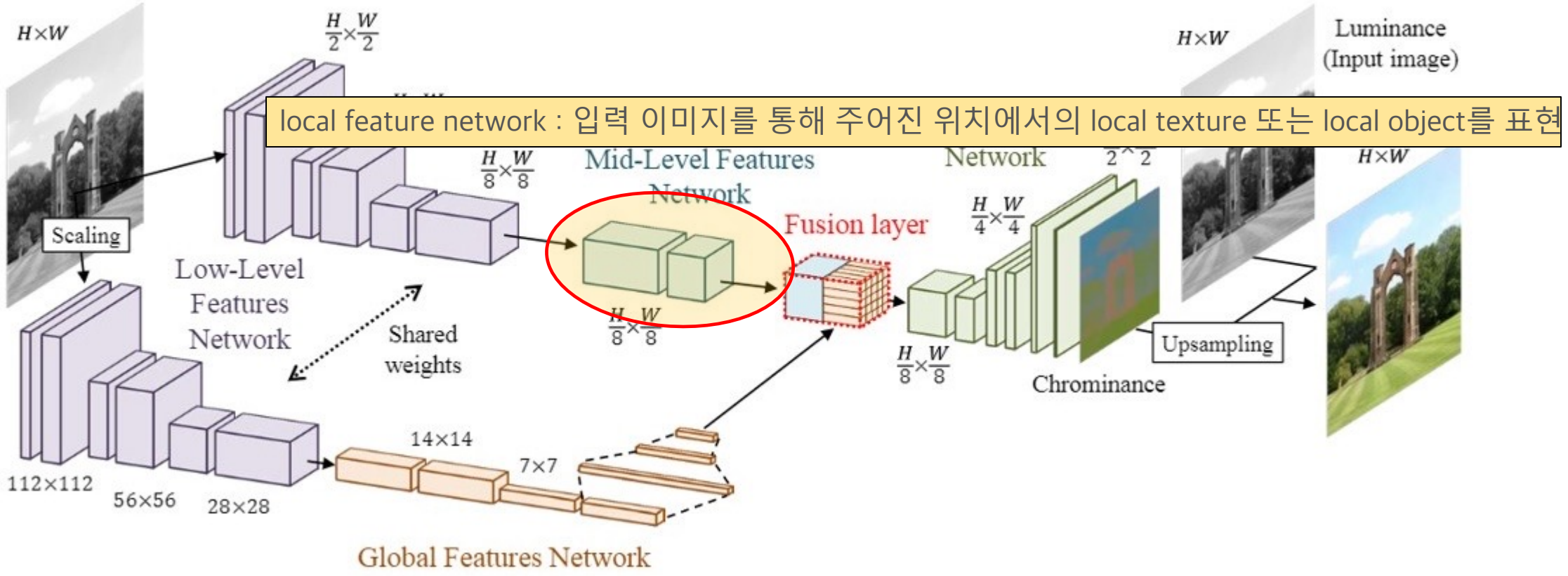
Low-level features network



(a) Low-Level Features network

Type	Kernel	Stride	Outputs
conv.	3 × 3	2 × 2	64
conv.	3 × 3	1 × 1	128
conv.	3 × 3	2 × 2	128
conv.	3 × 3	1 × 1	256
conv.	3 × 3	2 × 2	256
conv.	3 × 3	1 × 1	512

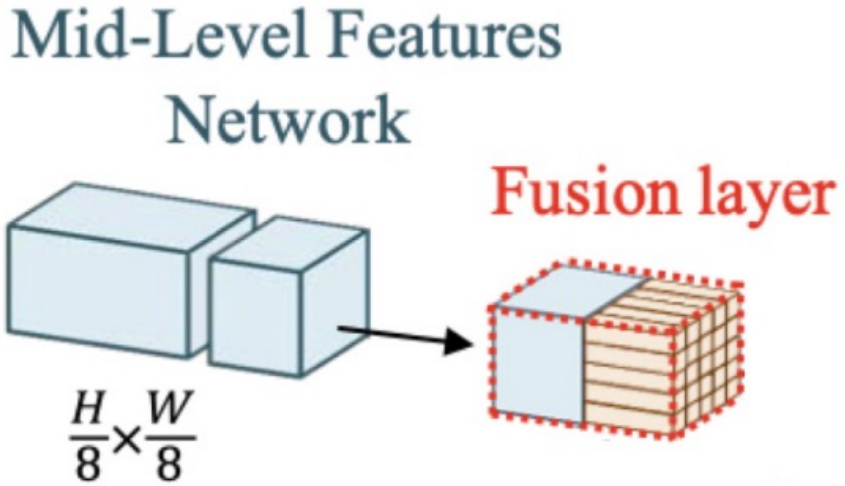
모델 아키텍쳐



local feature network : 입력 이미지를 통해 주어진 위치에서의 local texture 또는 local object를 표현

모델 아키텍쳐

두개의 Conv layer로 구성
output은 low-level features의 scaled version

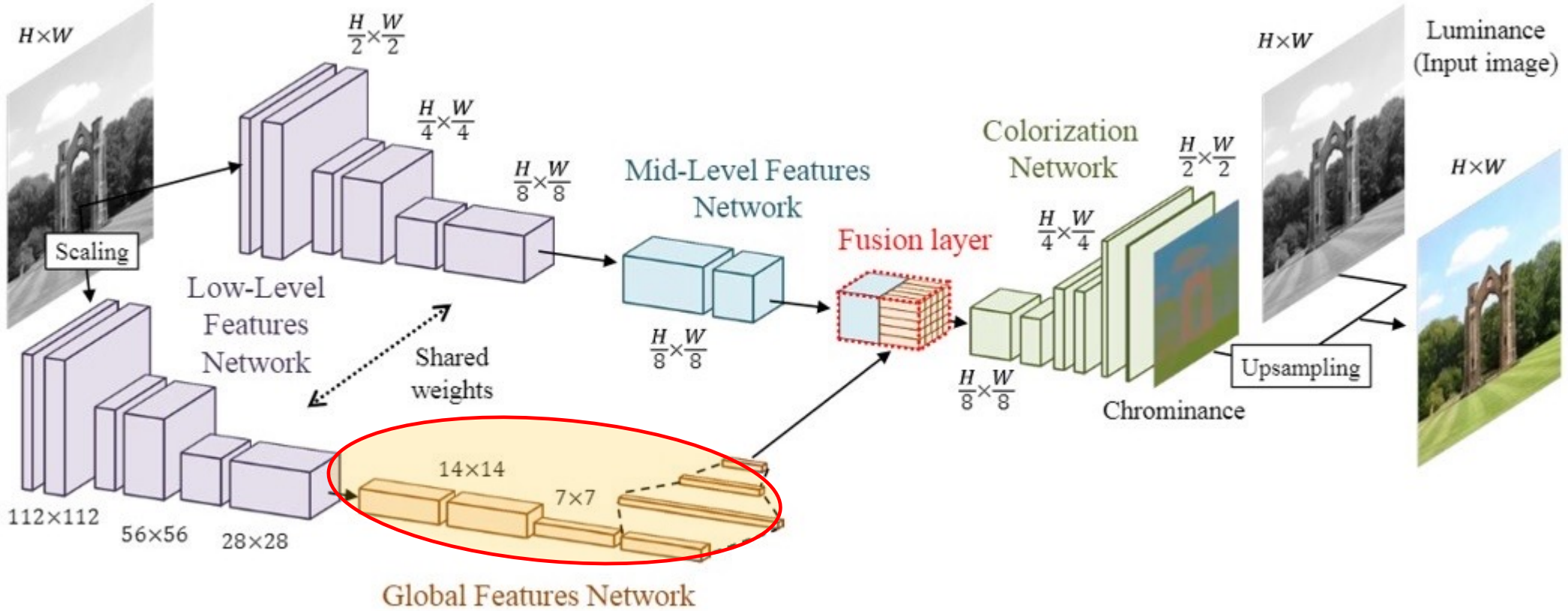


(c) Mid-Level features network

Type	Kernel	Stride	Outputs
conv.	3×3	1×1	512
conv.	3×3	1×1	256

Mid level features network

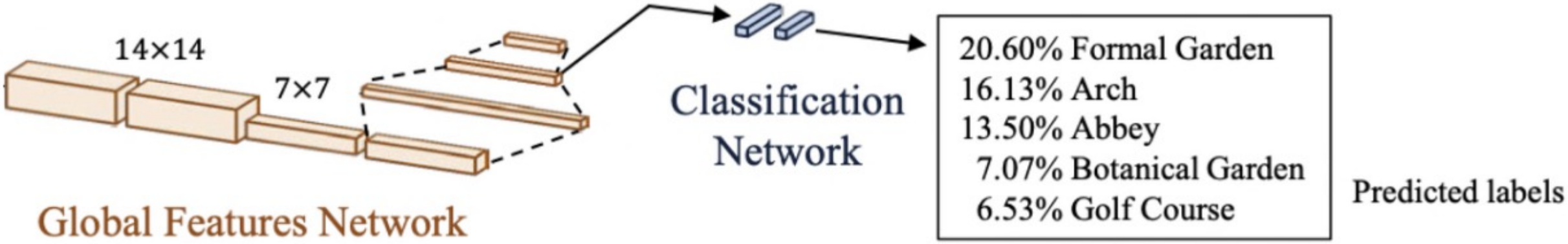
모델 아키텍쳐



global feature network : 전체 학습데이터에 있는 이미지 레벨에서의 정보를 학습
ex. 이미지가 외부에서 촬영된 이미지인지 내부에서 촬영된 이미지인지 또는 낮인지 밤인지 등의 정보를 학습

global features network

image의 global한 특징을 추출하는 역할
(목적) '**Image priors**'를 제공 (해당 이미지의 큰 설명 ex) 실내, 야외…
(구성) Low level features를 4개의 Convolution과 3개의 Fully connected layer에 통과

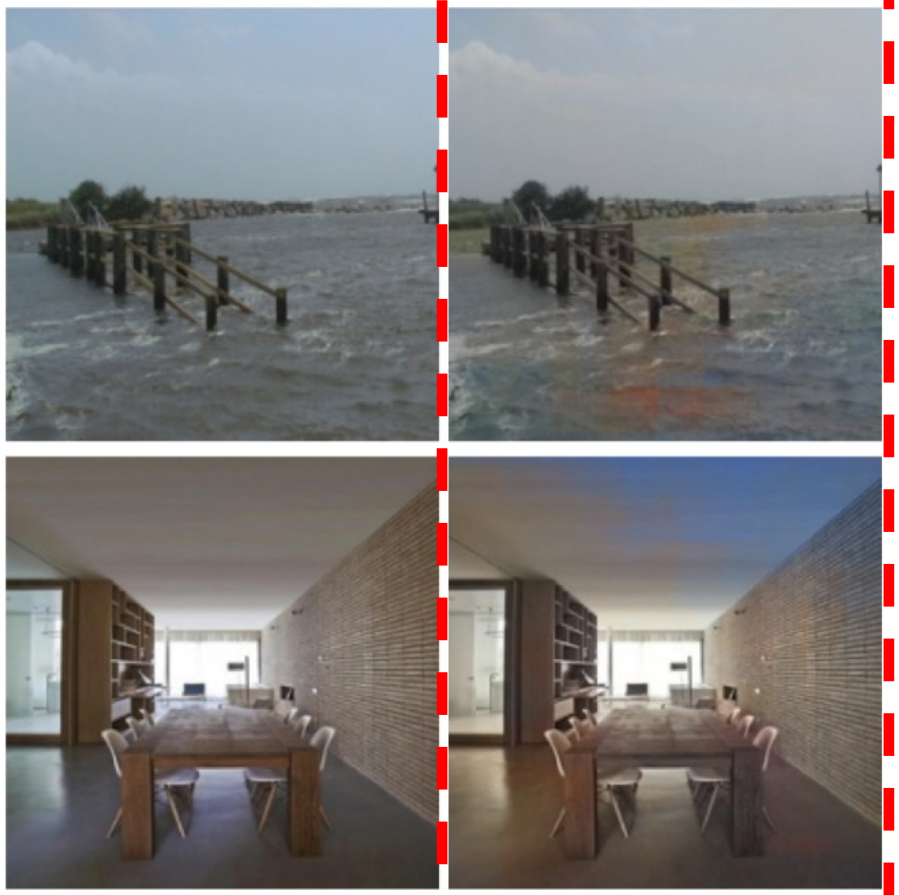


Global image features network

global features network

원본

Global 모델이 없는 경우



Ground truth

Baseline

바다색이 땅처럼 황토색

천장이 실외처럼 하늘색

global features network

원본

Global 모델이 없는 경우

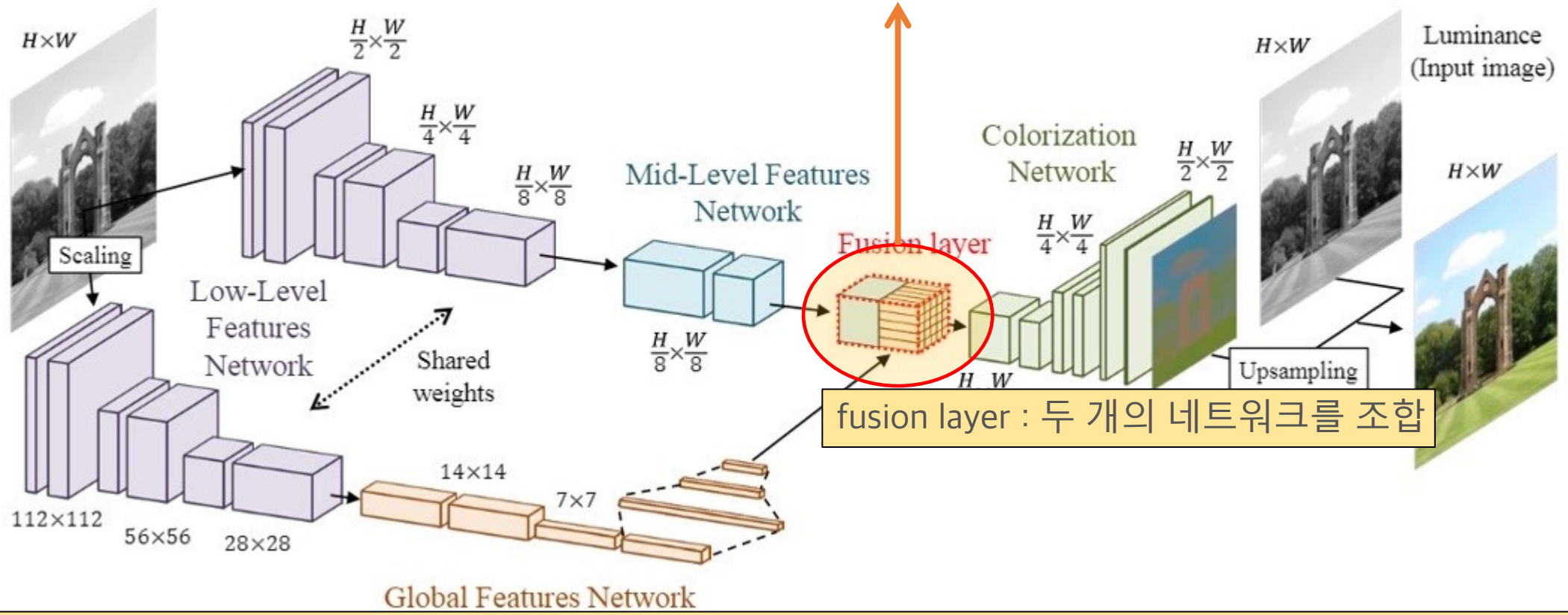
```
class GlobalFeature
    def __init__(se
        super(Globa                    _()
        self.conv1 = nn.Conv2d(in_channels=512, out_channels=512, stride=2, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(in_channels=512, out_channels=512, stride=1, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(in_channels=512, out_channels=512, stride=2, kernel_size=3, padding=1)
        self.conv4 = nn.Conv2d(in_channels=512, out_channels=512, stride=1, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(7*7*512, 1024)
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, 256)

    def forward(self, x):
        y = F.relu(self.conv1(x))
        y = F.relu(self.conv2(y))
        y = F.relu(self.conv3(y))
        y = F.relu(self.conv4(y))
        y = y.view(-1, 7*7*512)
        y = F.relu
        y = F.relu
        out = y
        classification_in = y
        out = F.relu(self.fc3(out))
        return out, classification_in
```

global network에 image label을 target으로 분류하는 **classification net**이 추가로 존재
Loss는 cross entropy를 사용하며, 최대한 global net의 잘못된 판단을 줄이기 위한 시도

모델 아키텍쳐

$$\mathbf{y}_{u,v}^{\text{fusion}} = \sigma\left(\mathbf{b} + W\begin{bmatrix}\mathbf{y}^{\text{global}} \\ \mathbf{y}_{u,v}^{\text{mid}}\end{bmatrix}\right)$$

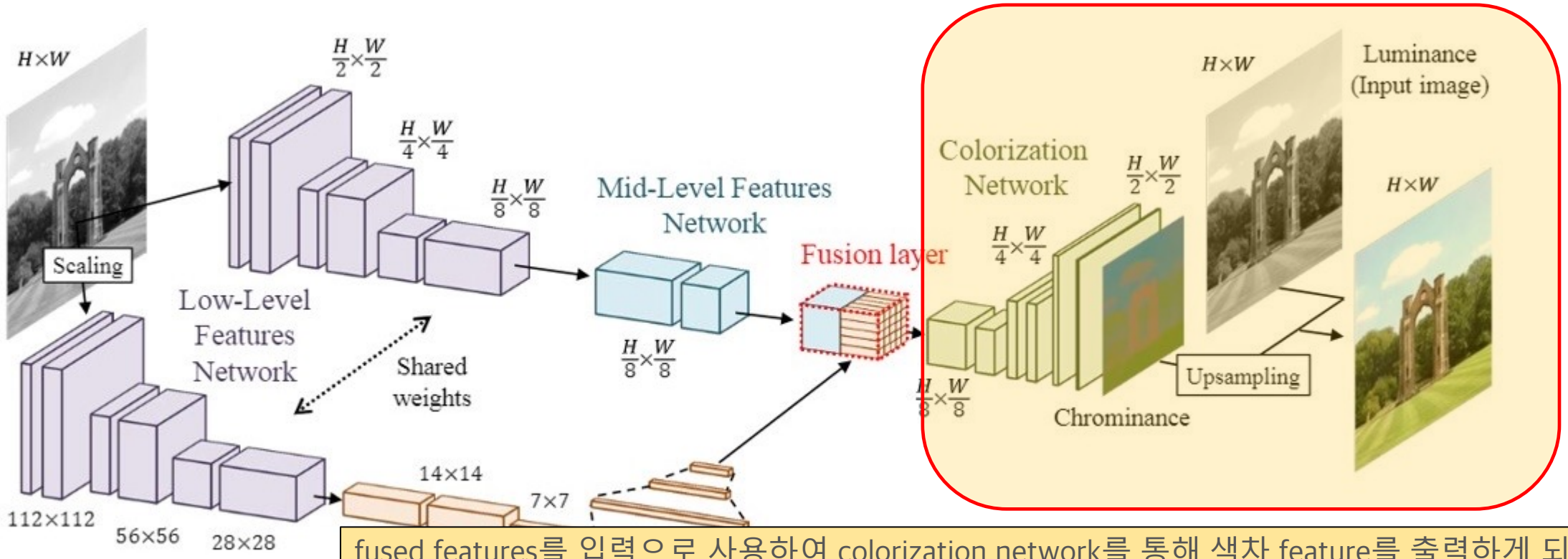


fusion layer : 두 개의 네트워크를 조합

global feature network에서의 결과인 1차원 vector를 mid-level features network의 각 location마다 뎁스 방향으로 concatenate

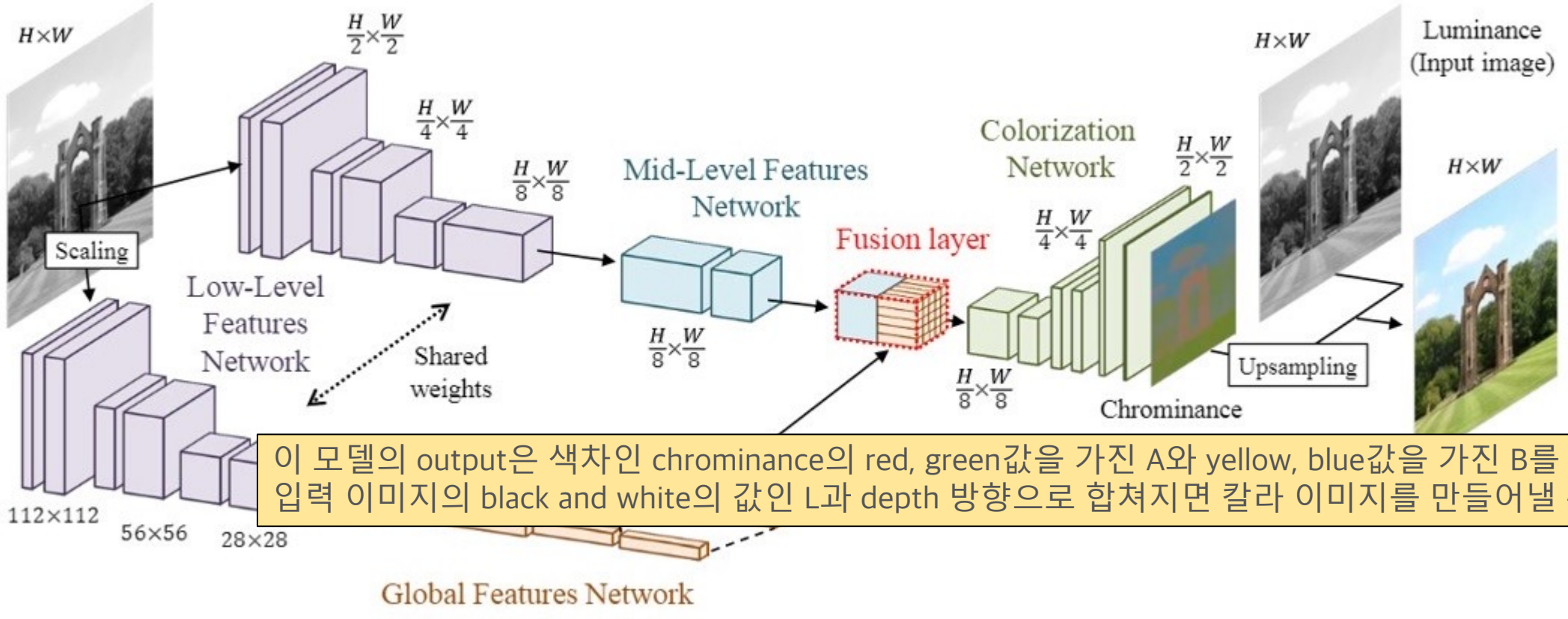
gobal feature network의 결과로 256vector가 나오고 mid-level feature network의 결과로 256뎁스가 나오는데,
이 둘을 뎁스 방향으로 concat하면 fusion layer는 512사이즈의 뎁스를 가지게 됨을 알 수 있습니다.

모델 아키텍쳐

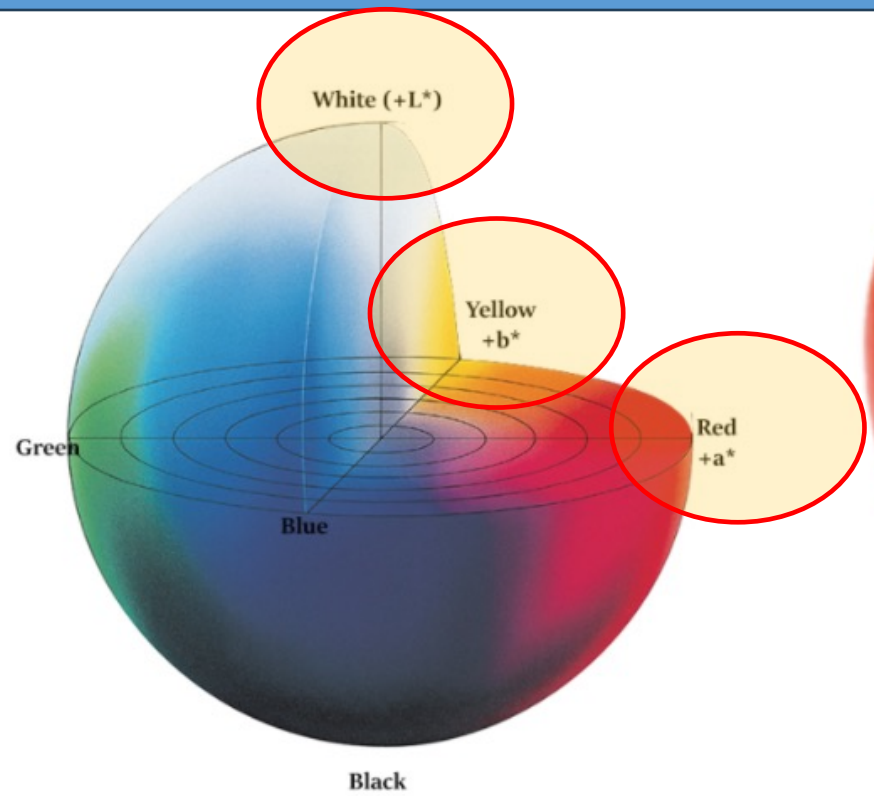


fused features를 입력으로 사용하여 colorization network를 통해 색차 feature를 출력하게 되고
이 결과를 input 이미지와 합쳐서 최종 결과물인 흑백 이미지를 색이 있는 이미지로 만들게 됩니다.

모델 아키텍쳐



이 모델의 output은 색차인 chrominance의 red, green값을 가진 A와 yellow, blue값을 가진 B를 출력하므로 입력 이미지의 black and white의 값인 L과 depth 방향으로 합쳐지면 칼라 이미지를 만들어낼 수 있습니다.



L* = 43.31

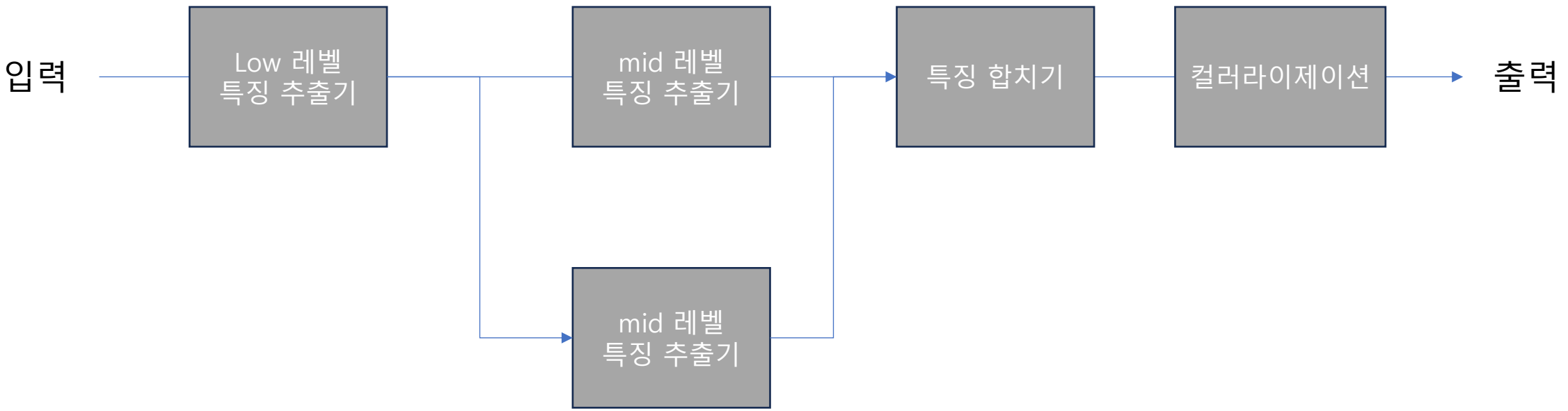
a* = 47.63

b* = 14.12

CIE L*a*b* color space

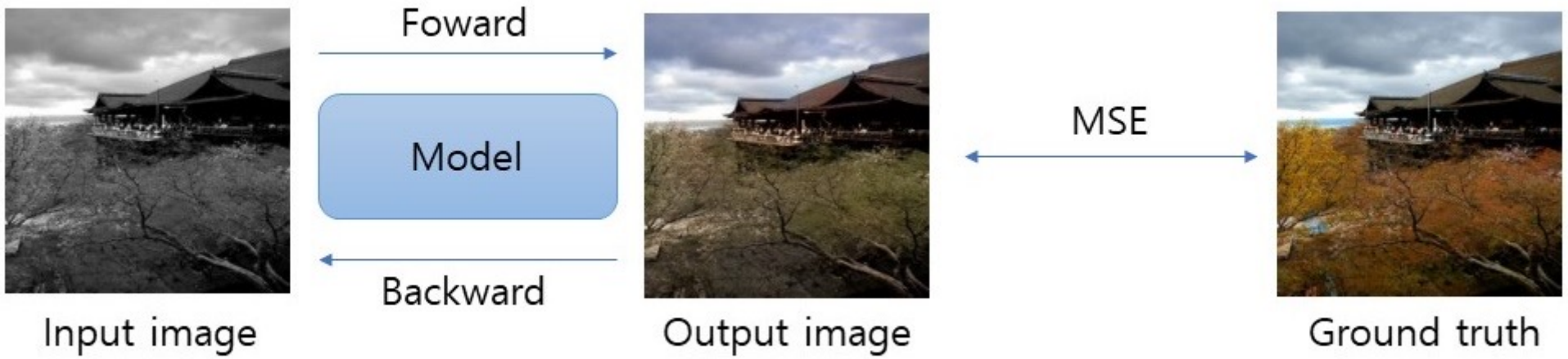
- Colorization network의 output은 **CIE L*a*b* color space의 a*, b* 값**
- CIE L*a*b*는 색상을 나타내는 map의 종류로, L*는 밝기, a* 와 b* 는 각각 색상의 방향을 나타냄

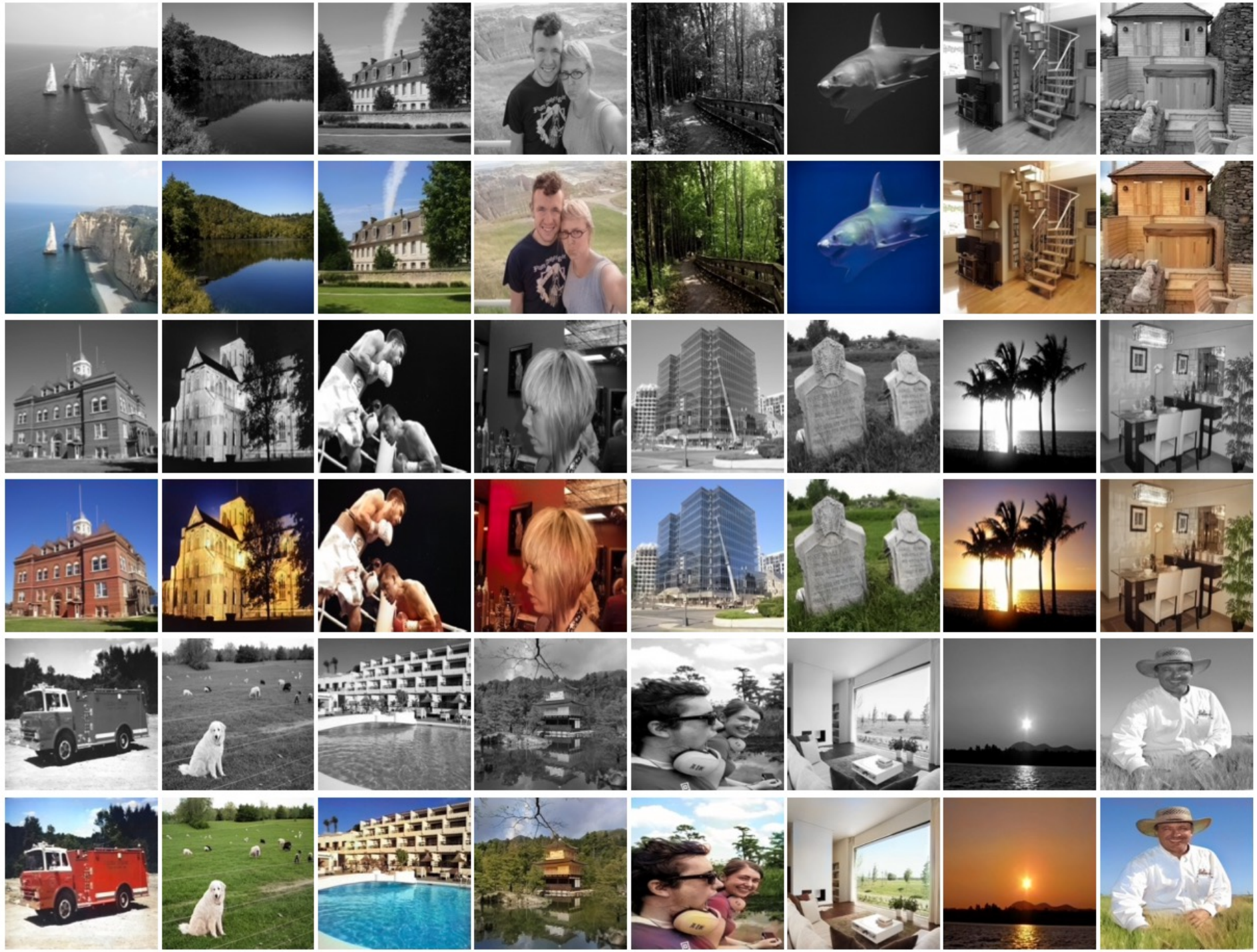
→ Colorization network는 a*, b*의 component를 예측하는 것을 목적


입력
Low 레벨
특징 추출기
mid 레벨
특징 추출기
mid 레벨
특징 추출기
특징 합치기
컬러라이제이션
출력

학습 과정

Input으로 흑백의 이미지가 입력되고, output으로 색차 이미지가 출력됨
출력된 output은 ground truth와 MSE loss계산하고, backpropagate 되면서 학습함





- 한계 또한 존재함

-data-driven task이기 때문에 train data와 유사한 image만 정확히 채색할 수 있음

- 남성의 셔츠처럼 어느 색이어도 가능한 모호한 범주의 경우 train data에 단순히 의존

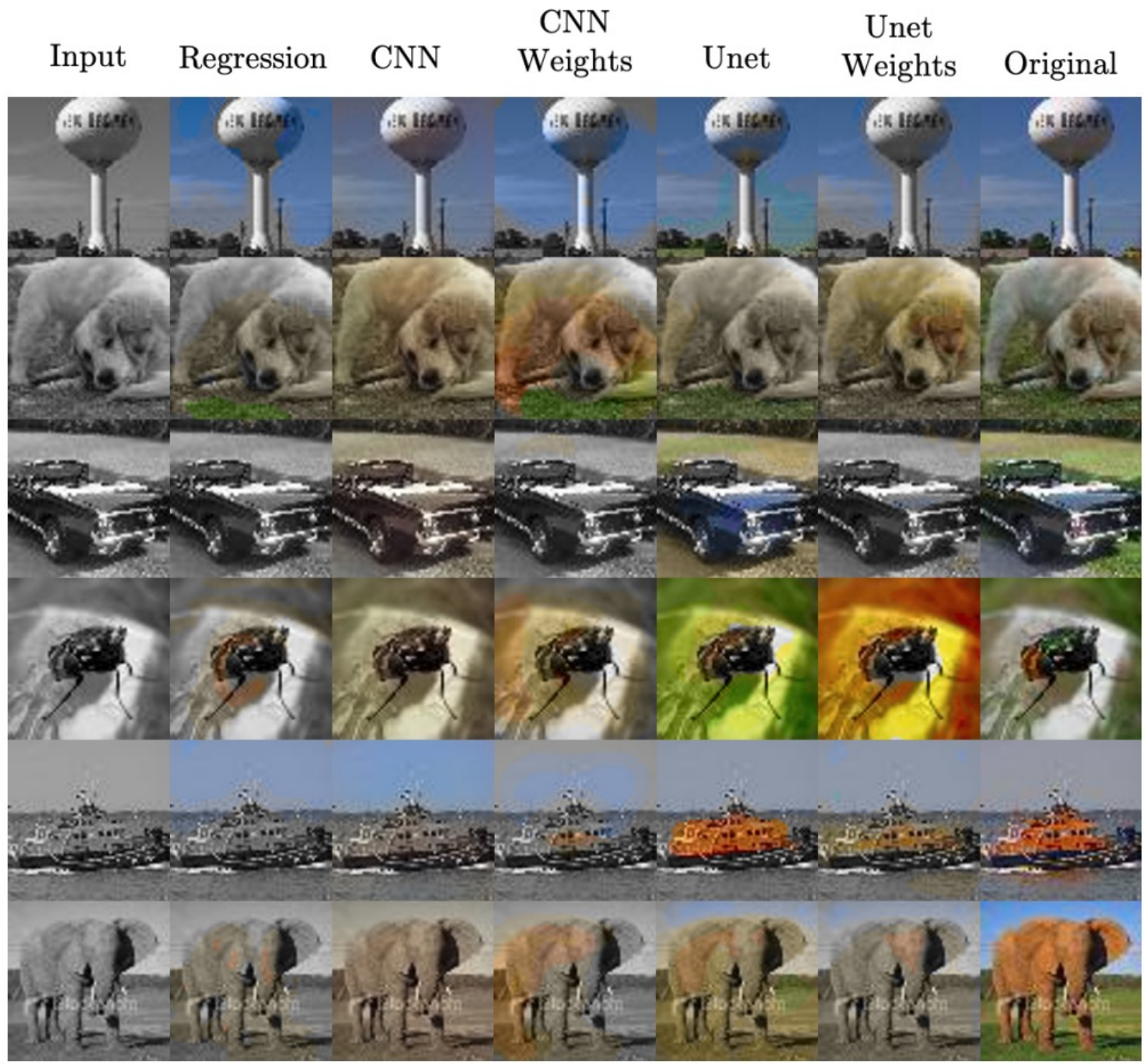


Figure 3. Results on Tiny ImageNet Dataset