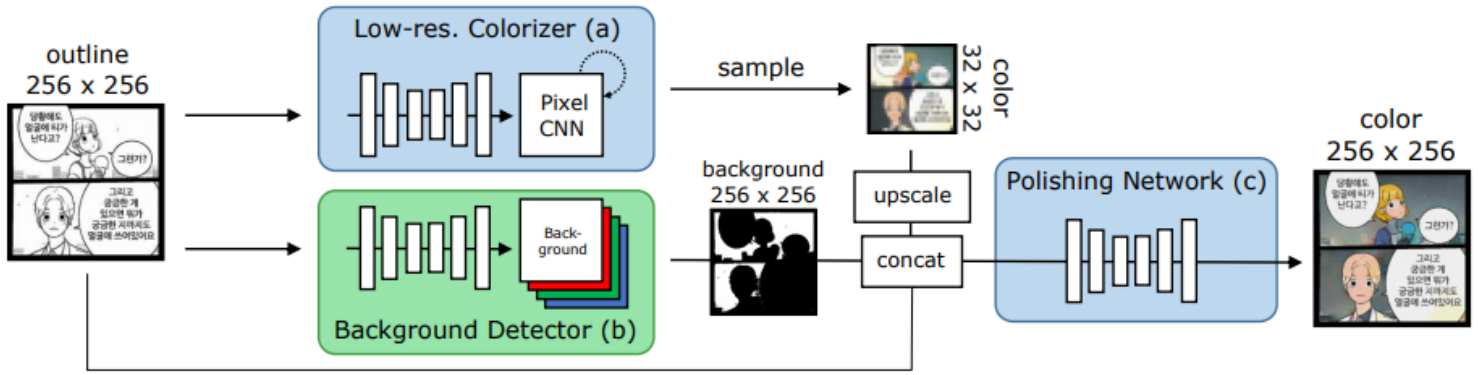
Consistent Comic Colorization with Pixel-wise Background Classification

논문 소개

- https://nips2017creativity.github.io/doc/Consistent_Comic_Colorization.pdf
- 만화/웹툰의 outline 이미지를 넣어주면 자동으로 채색된 결과를 반환하는 모델

모델 구조



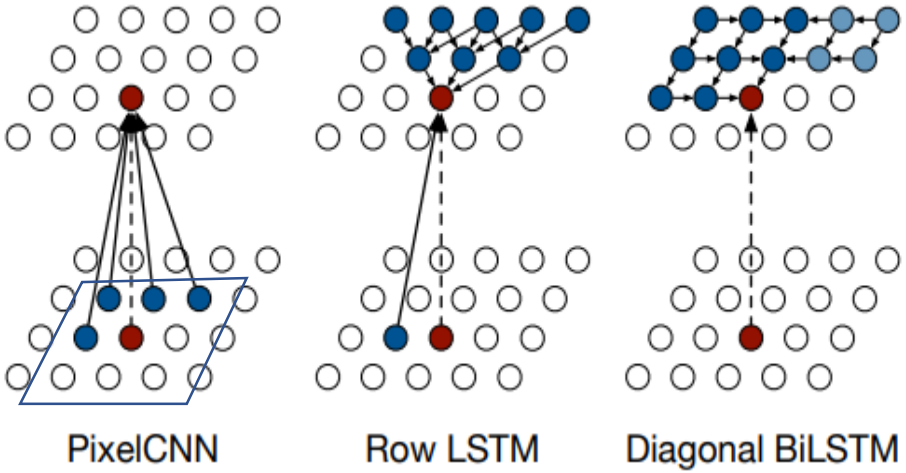
- (a) low-resolution colorizer: 이미지를 저해상도로 바꾸어 색상을 생성하는 부분
- (b) Background Detector: 색상을 detect하는 부분
- (c) Polishing Network: a, b 부분의 출력 값을 이용해 원본 이미지에 색상을 입힌다

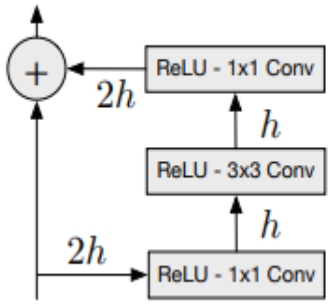
Low-Resolution Colorizer

PixelCNN

- Pixel Recurrent Neural Networks: https://arxiv.org/pdf/1601.06759.pdf
- RNN(LSTM) 기반으로 color 이미지를 생성하는 방법을 제시한 논문
- 논문 중간에 RNN 말고 CNN을 이용한 모델을 **간단히** 언급했는데, 이 것이 바로 PixelCNN이다.

PixelCNN



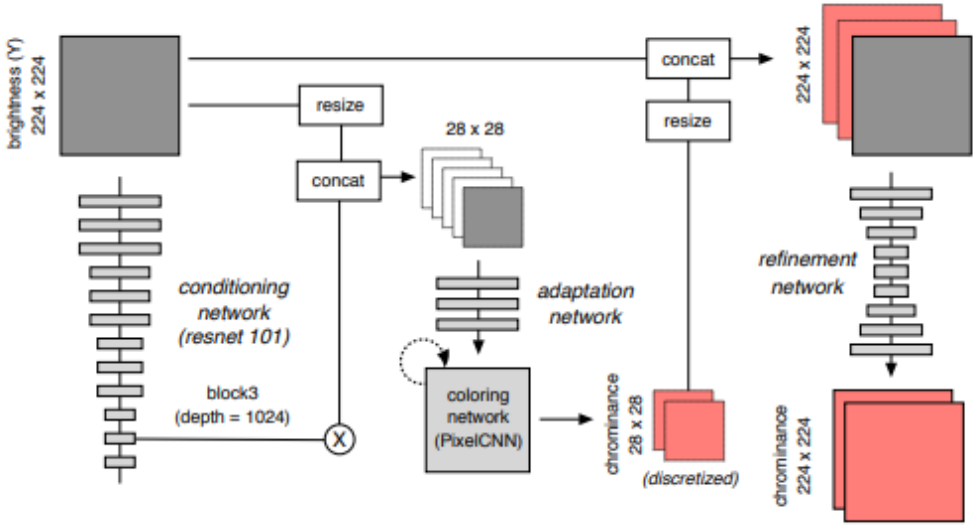


Residual block

1. 좌측이 Residual block을 여러 층 쌓는다. (Masked Convolution 이용)
2. 마지막 layer: 256-way softmax layer(Natural Images) / Sigmoid (MNIST)가 각각의 RGB 색상마다 존재
   즉, RGB값이 0일 확률, 1일 확률, .. 255일 확률을 각각 구해서 가장 높은 값 선택

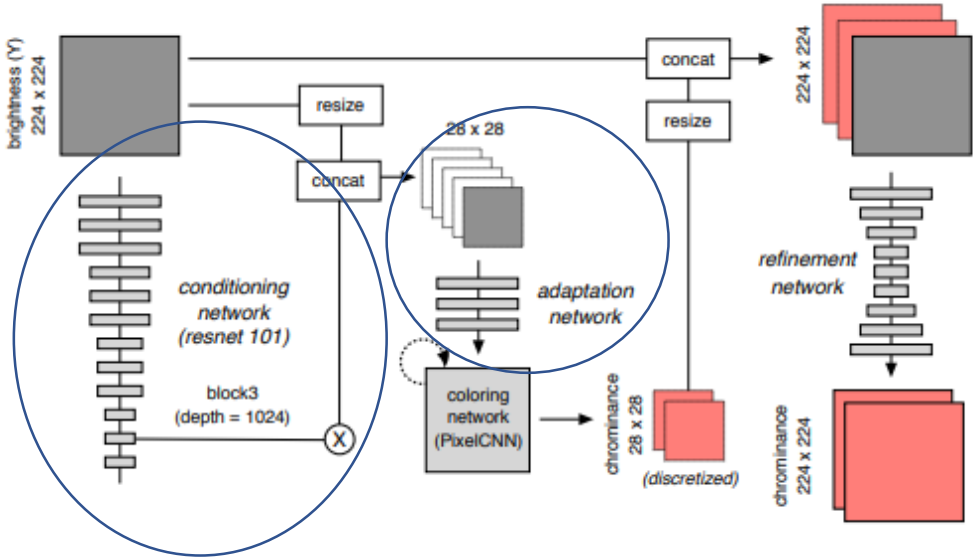
- Masked Convolution: 필터의 중심에서 우측과 아래의 가중치가 0인 필터

PixColor



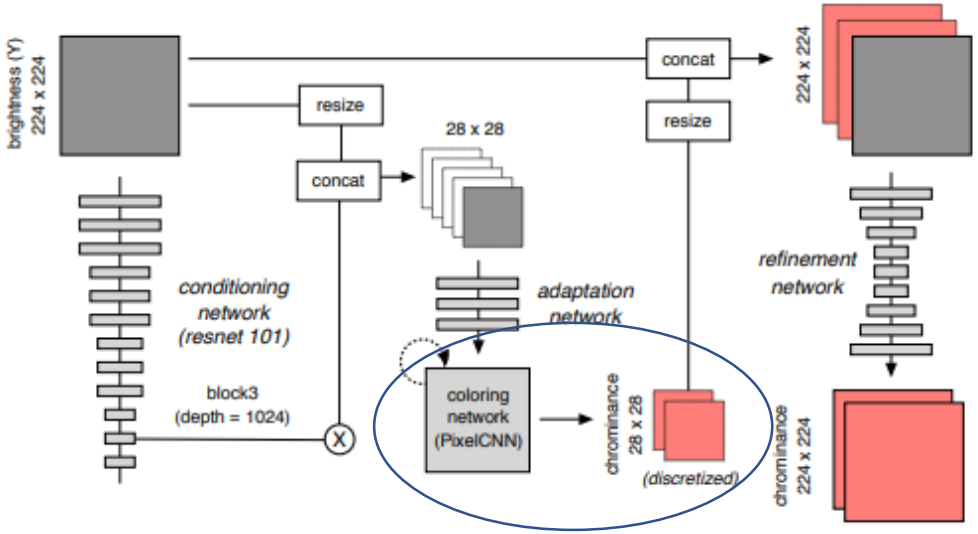
- https://arxiv.org/pdf/1705.07208.pdf
- 224*224 크기의 이미지를 받아 컬러값을 추출하는 모델
- 28*28 크기의 컬러 이미지를 생성하던 PixelCNN을 보완
- PixelCNN 출력을 224*224로 안 한 이유: 느려서

PixColor



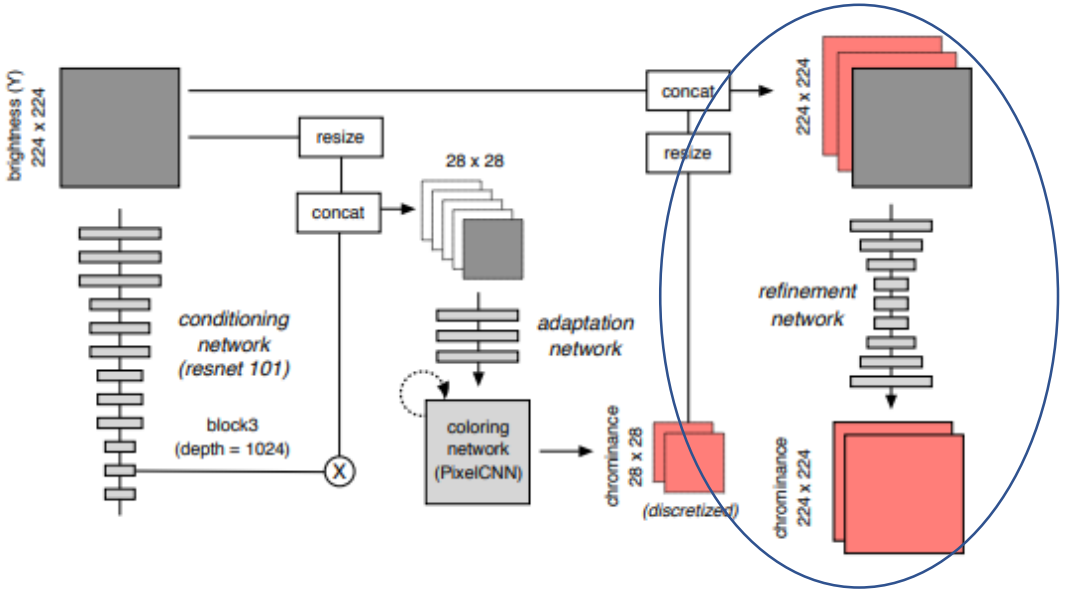
- Conditioning network: resnet 101의 block 3 값 이용 (이미지 feature 추출)
- Adaptation network: 명도값과 conditioning networ의 featur를 합쳐서 PixelCNN에 주기 위한 network
- 위와 같은 구조를 쓴 이유: PixelCNN의 출력 크기가 작지만, 더 잘 색상값을 추출하기 위해

PixColor



- Coloring Network: PixelCNN을 그대로 이용.
- 출력: chrominance(크로마, 색차)
  - Y/Cb/Cr의 Cb/Cr 부분
  - Y: 휘도(밝기)
  - U(Cb) = 파랑 – 휘도
  - V(Cr) = 빨강 - 휘도

PixColor



- 28*28 크기의 색차값과 224*224 크기의 휘도값을 하나로 합침
- Refinement network
  - 보다 자연스런 색을 만들기 위한 network
  - 색차값의 크기가 휘도값 크기보다 작기 때문에 이를 보정하기 위함

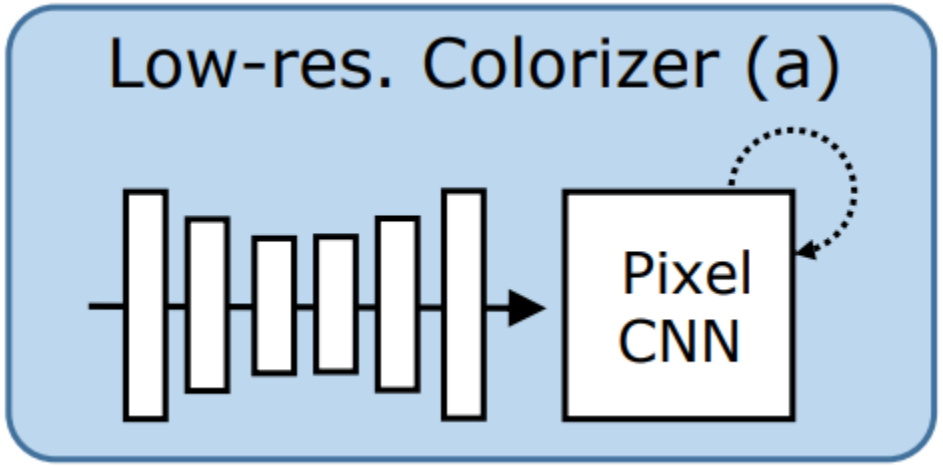
PixelCNN++

- https://arxiv.org/pdf/1701.05517.pdf
- PixelCNN에서 256-way softmax 대신 여러 logistic 함수를 조합하여 색상값 추정
  - 이유: 256-way softmax는 메모리를 너무 많이 써서 학습이 느리다
  - v: 색상값 밀도
  - μ: 평균
  - s: scale parameter(척도 모두)
  - σ(): logistic sigmoid function

$$\nu \sim \sum_{i=1}^{K} \pi_i \text{logistic}(\mu_i, s_i) \quad (1)$$

$$P(x|\pi, \mu, s) = \sum_{i=1}^{K} \pi_i \left[\sigma((x + 0.5 - \mu_i)/s_i) - \sigma((x - 0.5 - \mu_i)/s_i)\right], \quad (2)$$

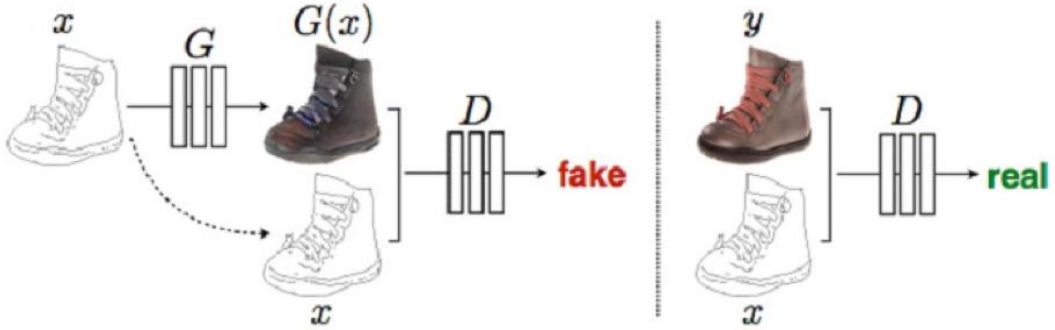
Low-Resolution Colorizer



- 저해상도(28*28) 색상 정보를 만드는 부분
- 모델 구조: PixColor의 colorization model 사용
- 추가로, PixelCNN++의 logistic mixture model 사용
  - PixColor에서는 사용하지 않음
  - Dataset 크기가 작아서 이 logistic mixture model을 사용하는 것이 더 나은 것 같음

Background Detector

cGAN (Conditional GAN)



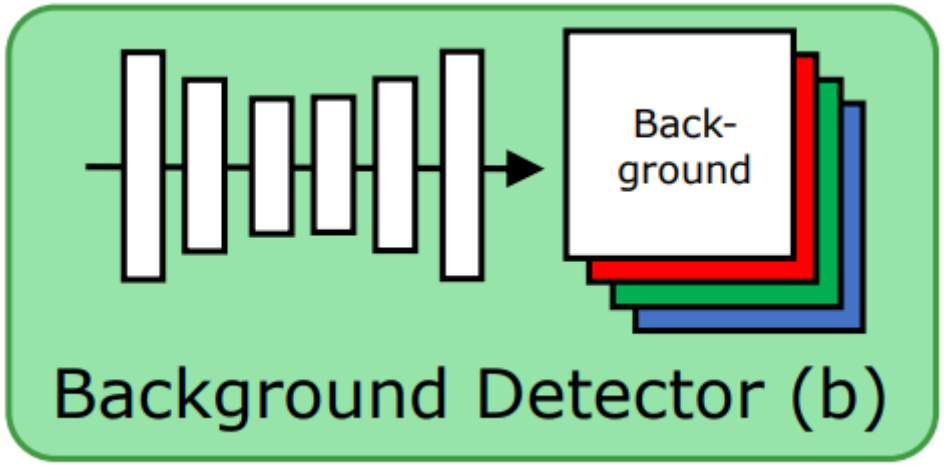
https://dreamgonfly.github.io/2018/03/17/gan-explained.html

- 랜덤 노이즈와 condition을 주면, 그에 맞는 출력값을 생성하는 모델
- 주어진 이미지를 다른 이미지로 변형시키는 것을 하나의 모델로 해결
- 기존 GAN과의 차이점
  - Random noise 뿐만 아니라 condition도 input으로 받는다

Image-to-Image translation with Conditional Adversarial Networks

- https://arxiv.org/pdf/1611.07004.pdf
- cGAN을 이용해 한 이미지를 다른 종류의 이미지로 변환시키는 모델 제안
  - 예: 위성 사진을 지도 이미지로, 스케치 사진을 실제 사진으로
- 모델 구조: cGAN 변형
  - Generator: U-net처럼 skip connection 추가
  - Markovian Discriminator(PatchGAN) 이용

Background Detector



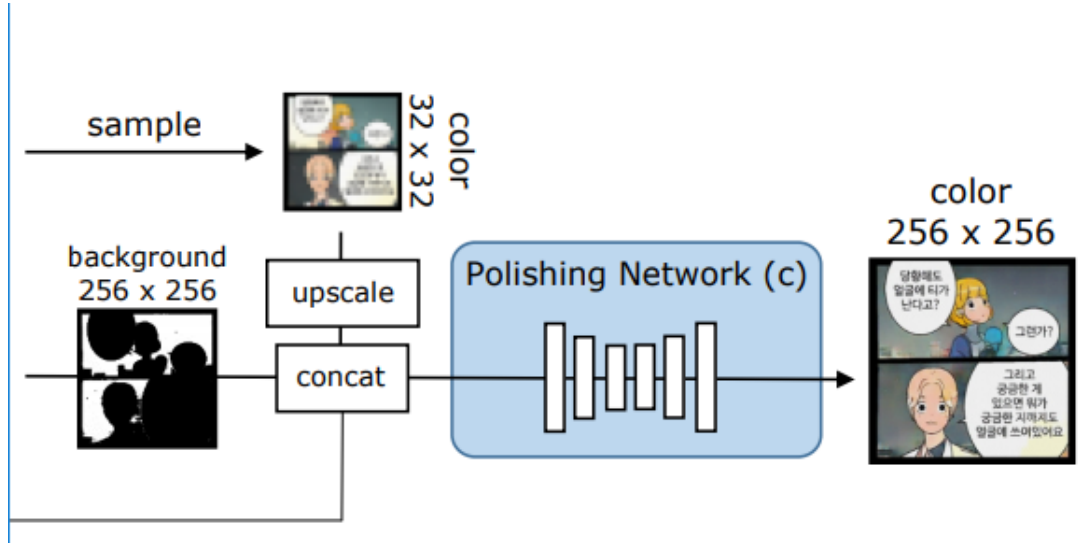
- 배경인 부분/배경 아닌 부분을 다른 방법으로 train/inference 하기 위해 배경을 분리할 필요가 있음
- 구현: Image-to-Image의 Generator Architecture를 그대로 사용
- CNN layer를 여러 겹 쌓은 encoder-decoder 모델

Background Detector

- 학습 방법
  1. Low-res. Colorizer의 출력값, 그리고 Ground Truth Image를 아래와 같이 변경
     1. 배경: 전체 범위 내 출력 RGB값을 평균낸다.
     2. 배경 아닌 부분: 출력 RGB 값 그대로 사용
  2. 위에서 생성한 이미지, 그리고 Ground Truth 이미지의 L1 distance를 loss function으로 사용
- 위와 같이 한 이유
  - 캐릭터와 말풍선은 색이 비슷비슷해서 상대적으로 색상 예측이 쉽다
  - 배경은 색을 마음대로 정해서 예측하기 어렵다.

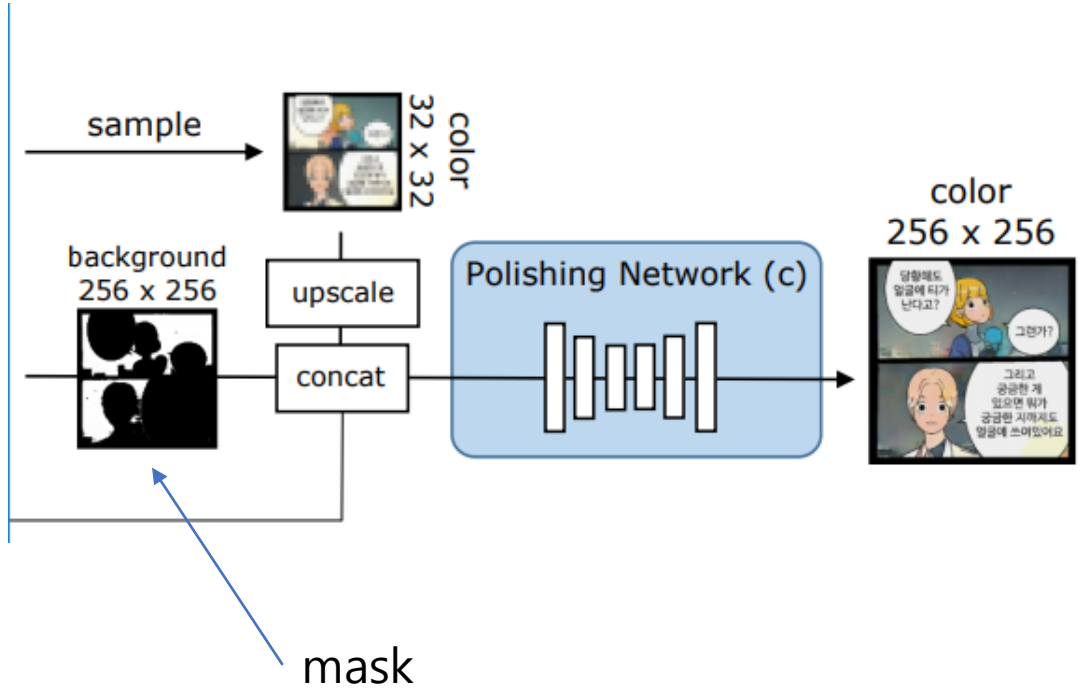
Polishing Network

Polishing Network



- 모델 구조: Image-to-Image의 Generator Architecture
- Polishing Network가 필요한 이유
  - 저해상도의 색상 이미지와 outline을 이용해 outline에 맞춰서 색을 입히려고.

Polishing Network



- Mask 범위
  - Train시: 랜덤 선택
  - Inference: 배경 부분 선택
- Mask하는 이유: 모델이 low-res. Colorization 출력값을 그대로 내보내는 것을 막기 위해
- Mask된 부분은 low-res. Colorization 출력값을 평균 내어 사용
  - 이렇게 해서 배경의 색을 하나로 통일시킨다.

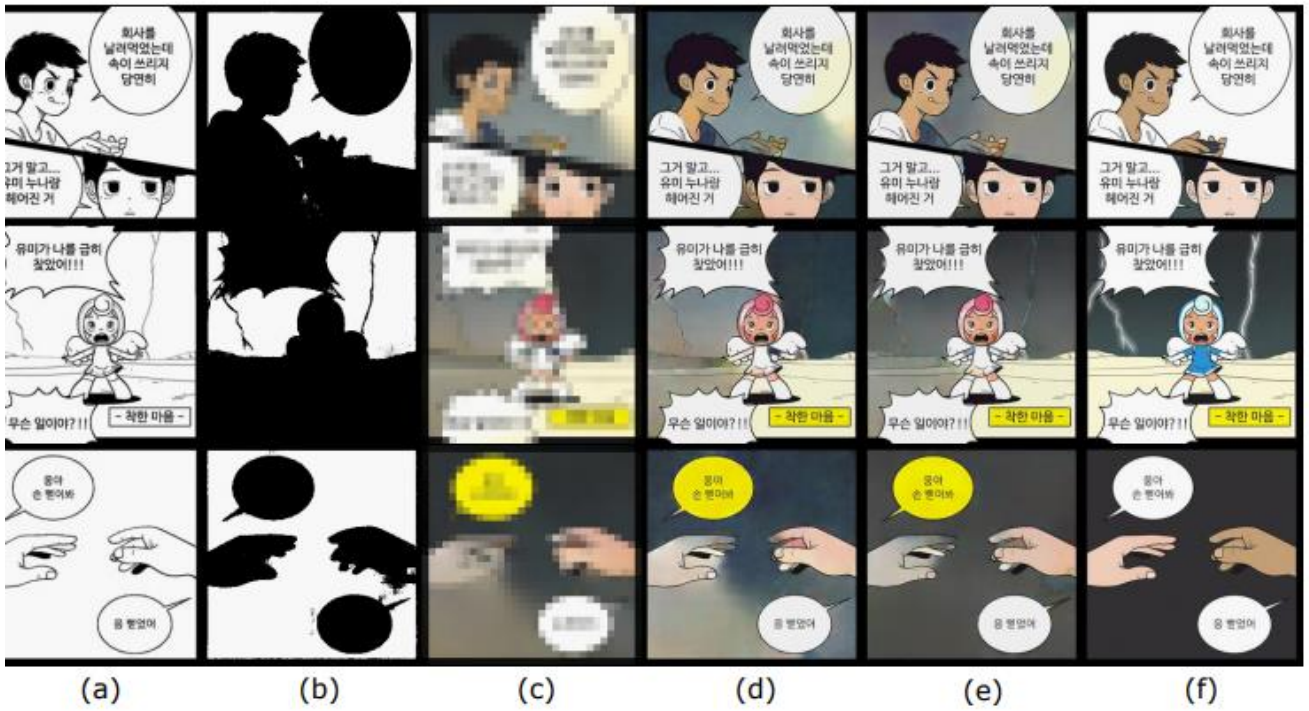
Train&Result

Train

- 데이터셋: 유미의 세포들 1화~238화 (7394개 이미지)
  - Training set: 7014개
  - Validation set: 380개
  - 이동건 작가님의 허락을 받고 논문에 사용하였다.
- Outline 이미지
  - OpenCV의 canny edge detection 결과
  - 여기에 이미지의 검정색 부분을 조합하여 생성
- 유미의 세포들
  - 유미의 감정과 생각을 의인화된 세포들로 표현하는 작품.
  - 2019. 05. 24. 기준 382화(국내) / 375화(영어) 연재중

Results

- 수치화된 결과는 없지만, 우측과 같이 Example 제시
- 배경 색상이 고르지 않은(d) 것을, background detection을 이용해 색상을 고르게 만든 것이 이 논문의 핵심



gure 2: From left, outline (a), background detection (b), low-resolution colorization (c), colorization ithout background information (d), colorization with background information (e), and ground truth olorization (f). Best seen in color. © Donggeon Lee.

Reference

- https://nips2017creativity.github.io/doc/Consistent_Comic_Colorization.pdf
- https://arxiv.org/pdf/1601.06759.pdf
- https://arxiv.org/pdf/1705.07208.pdf
- https://arxiv.org/pdf/1611.07004.pdf
- https://tensorflow.blog/2016/11/29/pixelcnn-1601-06759-summary/
- https://dreamgonfly.github.io/2018/03/17/gan-explained.html

Q&A