Machine Learning Image Restoration with PaddlePaddle

Hao Liu hliu2021@gwu.edu
Department of Computer Science, George Washington University

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

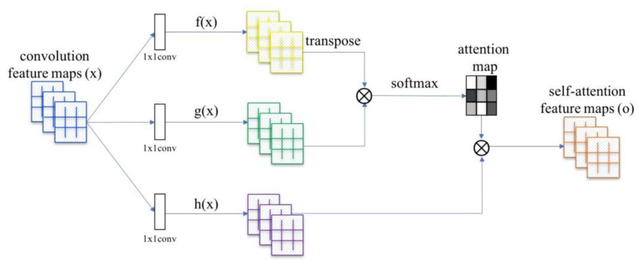
Abstract

As a specific image record of a period, old photos and historical images carry valuable memories, which have irreplaceable significance. However, in the past photography technology wasn't as advanced as in the present, and due to improper storage mechanisms, old photos faded, creased wearied, and torn. Hence, it is necessary to recover old photos. Photo restoration is meant to use edge detection and techniques to recover photos in which areas are broken or missing —— the traditional way of photo recovering, which does not perform well if the repair area contains complex, non-repetitive structures. In recent years deep learning continues to make advances in computer vision and image processing, using large numbers of images to train networks, allowing trained models to have a large amount of prior knowledge of images, giving another way to restore photos.

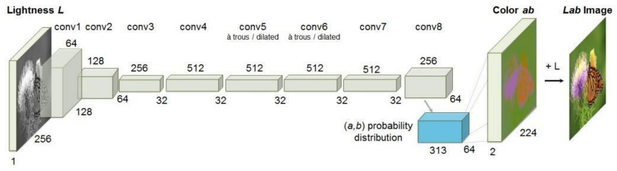
Using Self-Attention GAN as the principal solution while also using some other traditional image processing techniques to cover the small holes and missing pieces of the image. Then we will base on Colorful Image Colorization to restore image colors

Introduction

The Self-Attention Generative Adversarial Network, or SAGAN, allows for attention-driven, long-range dependency modeling for image generation tasks. Traditional convolutional GANs generate high-resolution details as a function of only spatially local points in lower-resolution feature maps. In SAGAN, details can be generated using cues from all feature locations. Moreover, the discriminator can check that highly detailed features in distant portions of the image are consistent with each other.

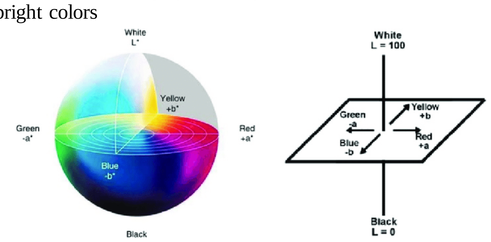


Colorization can be a powerful pretext task for self-supervised feature learning, acting as a cross-channel encoder. This approach results in state-of-the-art performance on several feature learning benchmarks. Low-level and semantic representations are utilized in the coloring process. Since many scene elements are currently generated naturally based on multi-peaked color distributions, our model is trained to predict pixel color histograms. This intermediate output can be used to automatically generate color images or to perform further manipulations before the images are formed.

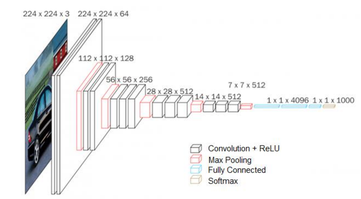


Methodology

Choose Lab formatted data, Lab mode is formed by three channels, they are Lightness, A, B. The A channel contains a color range from dark to green-gray to light pink. B channel is formed with colors from dark blue to grey and to yellow. Thus this color mixing forms bright colors



Input layer:
Since historic photos are mostly in black and white, the index converts the photo to greyscale upon input. This makes the photo a single channel instead of three. We are then able to use [C, H, W] to represent photos dimension, height, weight respectively.
To separate greyscale and color information from RGB photos, we can convert RGB to LAB as needed, thus the lightness in LAB represents outline and textures, A and B being the color channel. By inputting the L channel, we can make the model predict its A and B channels.
Output Layer:
In case insufficient color is generated, A and B color channels (0~255) are compressed, so inputting 256*256 pixels forms 313 groups of high-frequency color classification. Meanwhile downgrading [B,H,W,C] format to [H,W,313]. 313 represents 313 color combination probabilities.



Results

Model Training Result

```
W0423 20:17:06.295714 10992 device_context.cc:447] Please NOTE: device: 0, GPU Compute Capability: 8.6, Driver API Version: 11.5,
  Runtime API Version: 11.2
W0423 20:17:06.300696 10992 device_context.cc:465] device: 0, cuDNN Version: 8.2.
length of val dataset : 3923
iter:0   time:1  avg_loss5.7889
iter:100  time:34  avg_loss4.7249
iter:200  time:35  avg_loss4.3359
iter:300  time:34  avg_loss4.1676
iter:400  time:35  avg_loss4.2459
iter:500  time:35  avg_loss4.1400
iter:600  time:35  avg_loss4.1643
iter:700  time:34  avg_loss3.9914
```

Model Evaluation

iteration	Iter Accuracy	Orig Accuracy
100	50.5%	51.5%
200	50.2%	50.7%
300	50.2%	49.9%
400	52.4%	51.9%

Predicting Results

Original

Output(eccv16)

Output(siggraph17)

Original

Output(eccv16)

Output(siggraph17)

Conclusion

Upon step-by-step implementation of the project, I realized that, although photo saturation has improved by suppressing the averaging and increasing the dispersion, the results are not as appealing as realistic images. Only in some sense will it have better results (landscape etc.).
The project needs a better-designed and optimized loss function to make coloring results realistic enough. The project was also restricted by the small sample size, I used the ImageNet-mini database as a training dataset to compensate for my low spec hardware.
The result can be further improved if utilizing a larger dataset, more fine-tuning with the parameters, and more training time.

References

ZMpursue. (2020, October 31). ZMpursue/Paddlecolorization. GitHub. Retrieved from https://github.com/ZMpursue/PaddleColorization

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019, May). Self-attention generative adversarial networks. In International conference on machine learning (pp. 7354-7363). PMLR.

Zhang, R., Isola, P., & Efros, A. A. (2016, October). Colorful image colorization. In European conference on computer vision (pp. 649-666). Springer, Cham.

Larsson, G., Maire, M., & Shakhnarovich, G. (2016, October). Learning representations for automatic colorization. In European conference on computer vision. Springer, Cham.