COLORIZING BLACK&WHITE IMAGES USING GAN

PMLDL Assignment 2

Authors
Alimzhan Kaiyrly — Valihan Ilyasov
l.kayirly@innopolis.university — V.ilyasov@innopolis.university

Contents

1 Project Description

Github link: https://github.com/AlimKAH/Black-White-photo-colorization

1.1 Why colorizing of an Image?

Our team chose the topic of ”Colorizing Black and White Images using GAN” for our assignment due to its relevance and potential applications in various fields. We believe most of us have some old photos in black and white which would be great to convert to RGB schema. Or you are a manga reader and want to see your favorite manga colored?

This is a small possibilities that are opened to us by GAN’s. So why not to create a method that solves problems of humanity?

1.2 How we approached the problem?

Our solution starts from exploring following scientific paper: *Image-to-Image Translation with Conditional Adversarial Networks by Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros.*

After we got acknowledged with the paper we started implementing our own solution and after that tried some pre-trained models with transfer learning.

We implemented following models:

1. Our own model with U-net as Generator
2. ResNet 18
3. ResNet 34

1.3 Dataset Used:

We took a slice(10.000 images) from COCO dataset. We find it sufficient enough for our purposes

2 Main Results

Section covers final hyper-parameters for all the models, their results, comparison between them and final best solution.

2.1 *Dataset description*

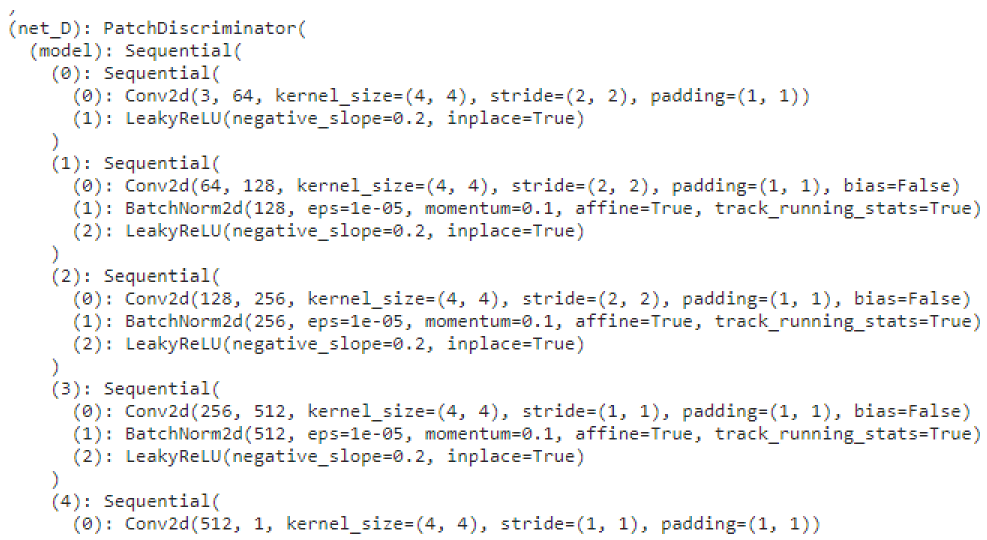
Dataset used: COCO 2017. We took 10000 samples where 8000 of them are used for training and 2000 for validation. Images from the dataset were resized to 256*256, and converted from RGB to L*a*b. Also some augmentation was performed.

2.2 Our model

As in the original paper our model revolves around generator based on U-Net. As the size for input image is 256*256 we need 8 layers of down-sampling to transform it to 1x1 image and 8 up-sampling layers to scale it back to norm. So our Generator consists from the following layers:

- 8 Sequential Layers, activation function LeakyReLU
- Downsampling for every layer
- BatchNorm2D layer
- RelU in last layer
- Upsampling layers with RelU and BatchNorm2d
- Tanh

For Discriminator we took Patch Discriminator. The idea behind that is we want to compare image by chunks as we generating a colored picture. It's architecture is quite simple:



```
,
(net_D): PatchDiscriminator(
  (model): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2, inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2, inplace=True)
    )
    (2): Sequential(
      (0): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2, inplace=True)
    )
    (3): Sequential(
      (0): Conv2d(256, 512, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1), bias=False)
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2, inplace=True)
    )
    (4): Sequential(
      (0): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
```

Figure 1: Discriminator Architecture

Model Hyper-Parameters

- Epochs = 50
- Optimizer = ADAM
- learning rate 1e-4
- Number of Layers: 16 for Generator and 5 for Patch Discriminator

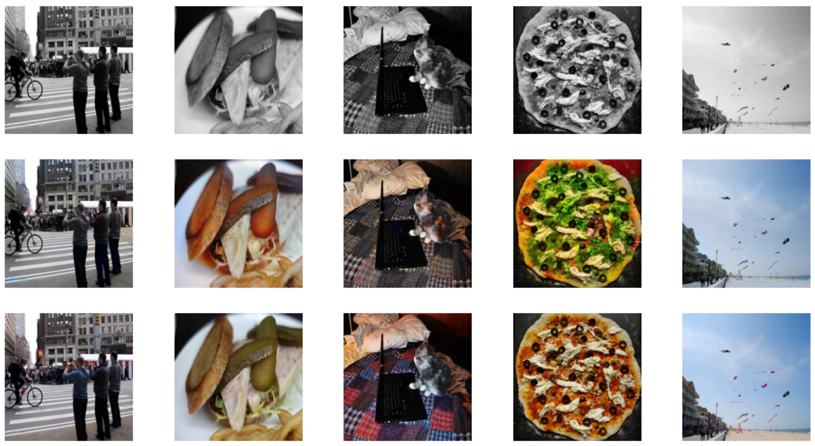


Figure 2: GAN Results

2.2.1 Results

As we can see our results weren't ideal. So we decided to continue our investigation and use pre-trained models.
Possible improvements: Increase the number of Epochs, so the model can improve its loss. Optimize the hyper parameters

2.3 ResNet18

We pulled the ResNet18 Model from fastai library cut-off last 2 layers and pre-trained our generator on the dataset. After that pass the weights we got to our main model, which resulted in significant decrease in training time. As except for generator, nothing changed previous description should cover this model too

Model Hyper-Parameters

- Epochs = 10
- Optimizer = ADAM
- learning rate 1e-4
- Generator - pretrained ResNet18

2.3.1 Results

ResNet18 although still having troubles colorizing image, performs much better, also considering quite low training time, only 10 epoch could provide much better results!

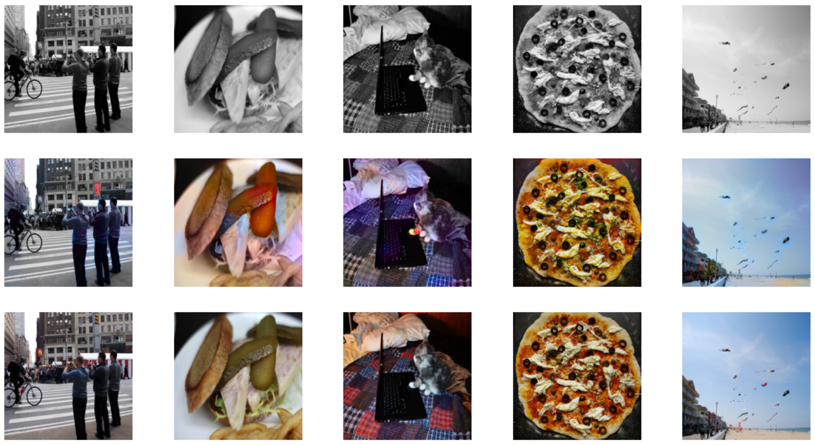


Figure 3: ResNet18 Results

2.4 ResNet34

The same applies here, although model is bigger.

Model Hyper-Parameters

- Epochs = 10
- Optimizer = ADAM
- learning rate 1e-4
- Generator - pretrained ResNet34

2.4.1 Results

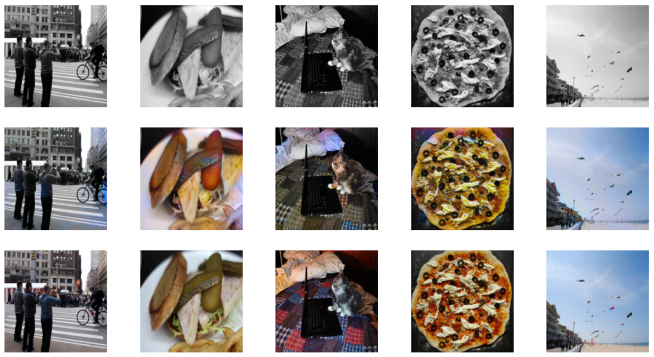


Figure 4: ResNet34 Results

And for our final decision tehre is a pre-trained ResNet34. The images it produced were in fact similar to ResNet18, however under close inspection it is obvious that ResNet34 performed better than other 2 options.

2.5 Side-to-side Comparison

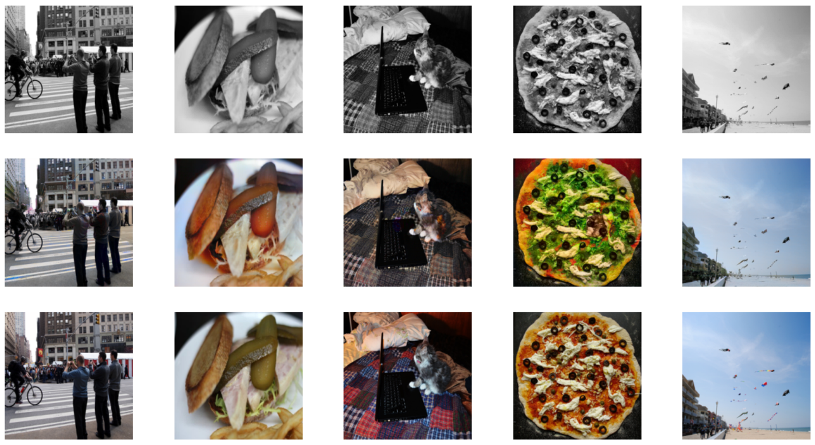


Figure 5: GAN Results

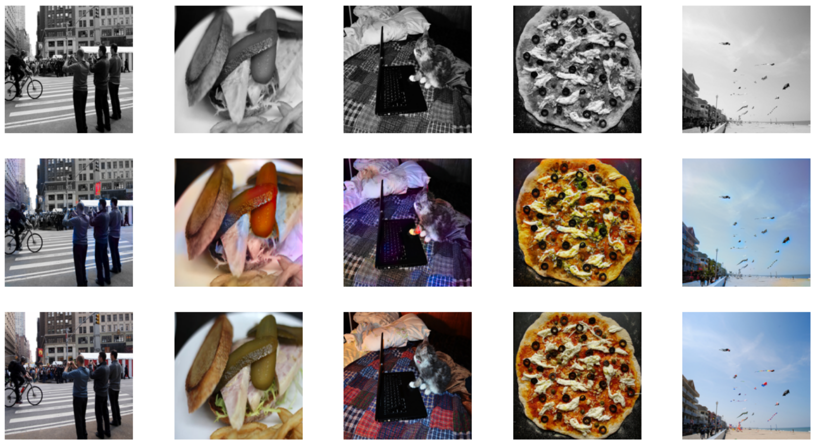


Figure 6: ResNet18 Results

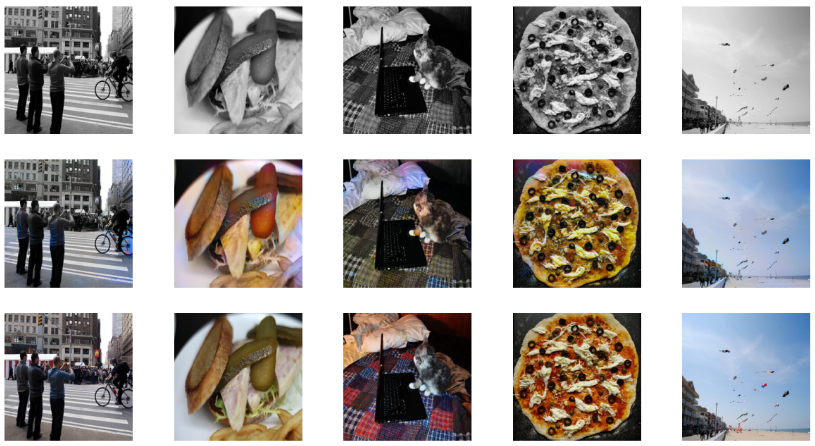


Figure 7: ResNet34 Results

2.6 Winner

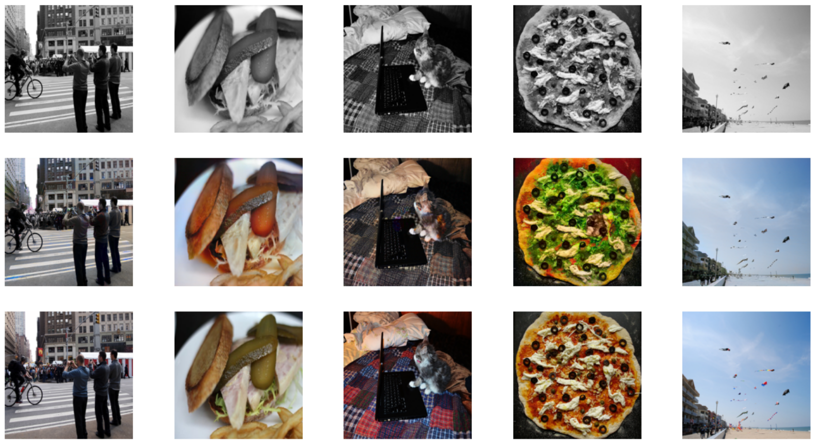


Figure 8: ResNet34 Results

We think that ***ResNet34*** performed the best!

3 Timeline of the project

Project Stage	**Project Plan**		
	Time	***Details***	***Results***
Exploring Gan's	1 week	Read the paper	Started to implement our solution
Gaining First Results	1 week	Train our GAN	Gained first impressions
New approach(ResNet18)	1 week	Train ResNet18	Results for ResNet18
Evaluation	1 week	Train ResNet34	Collect everything and compare
Report construction	1 week	Writing a report	Report

3.1 Individual Contribution

Alimzhan Kaiyrly: Found initial data, and constructed the plan for completing the project. Performed data exploration. Trained ResNets and collected results also wrote a report. Finalized the presentation
Valihan Ilyasov: Implemented a GAN and trained it. Found references and explored them. Created a presentation structure. Made a telegram bot

4 References and Acknowledgements

We extend our sincere gratitude to the authors of ***Image-to-Image Translation with Conditional Adversarial Networks***, ***Colorful Image Colorization*** and ***Colorizing black & white images with U-Net and conditional GAN*** for their exceptional contributions to the field. Their work has not only advanced the boundaries of our knowledge but has also been a significant source of inspiration for our research.