Standard GAN vs W-GAN for image colorization

Valentina Tonazzo
Elena Zoppellari

Università degli Studi di Padova

Outline

Introduction

Image colorization: The goal is to assign plausible colors to each pixel of a grayscale image.

In this project, we have implemented two generative models:

1. Based on the PatchGAN architecture developed by Isola et al. in 2016 [1]
2. Variation of the previous one introducing the Wasserstein distance as the objective function

Dataset

Tiny-ImageNet-200, composed by images with 64×64 pixels resolution.

[1] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," CoRR, vol. abs/1611.07004, 2016.

Data Pre-processing

- The images were transformed from the RGB to the *Lab* color space.
- Grayscale input images were created by isolating the Luminance channel, which was duplicated twice to preserve the original image structure of $64 \times 64 \times 3$.
- All images were normalized to the range of [-1,1].

Data Loading

- Training set: 100k images, divided into 500 per class.
- Testing set: 1k images, divided into 50 per class.
- Batch size: 128.

GANs and Conditional GANs

A Generative Adversarial Network (GAN) consists of two neural networks, a generator and a discriminator, which are trained one against the other to improve their performance.

- The generator produces fake samples from noise: $G : \mathbf{z} \rightarrow \mathbf{y}$.

Colorization task

To predict colored images from grayscale ones, not from noise!

Conditional GANs:

- The generator takes both noise and an observed grayscale image $\mathbf{x}$ as inputs: $G : \{\mathbf{z}, \mathbf{x}\} \rightarrow \mathbf{y}$.
- The discriminator distinguishes between real and fake samples, returning a probability value p for the candidate colored image $\mathbf{y'}$: $D : \{\mathbf{y'}, \mathbf{x}\} \rightarrow p$.

cGAN objective function (1)

The general loss equation of a cGAN model is:

$$L_{cGAN}(G, D) = \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}}[\log D(\mathbf{x}, \mathbf{y})] + \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{z}}[\log(1 - D(\mathbf{x}, G(\mathbf{x}, \mathbf{z}))] \quad (1)$$

$$\min_G \max_D L_{cGAN}(G, D) \quad (2)$$

- Adding also $L1$, the Generator produces images that are closer to the ground truth:

$$L_{cGAN,tot}(G, D) = L_{cGAN}(G, D) + \lambda \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y},\mathbf{z}}[\|\mathbf{y} - G(\mathbf{x}, \mathbf{z})\|_1] \quad (3)$$

- In Isola et al. study, it was observed that the model learned to simply ignore the noise $\rightarrow \mathbf{z} = 0$.

cGAN objective function (2)

UNIVERSITÀ DEGLI STUDI DI PADOVA

In practice, Binary Cross Entropy has been used:

$$BCE(\alpha, \beta) = \frac{1}{m} \sum_{i=1}^{m} [\alpha_i \times \log \beta_i + (1 - \alpha_i) \times \log (1 - \beta_i)] \quad (4)$$

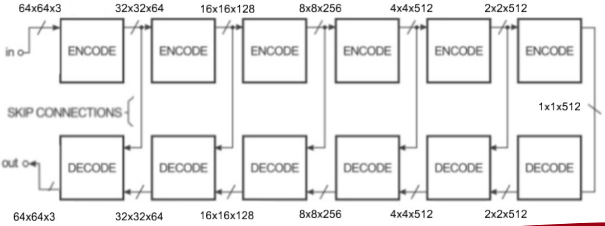
- For the Discriminator:
  * Real images: $\beta = D(\mathbf{x}, \mathbf{y})$, the desired label is $\alpha = 1$.
  * Fake images: $\beta = D(\mathbf{x}, G(\mathbf{x}, \mathbf{z} = 0))$, the desired label is $\alpha = 0$.
  * $D_{loss} = \frac{BCE_{fake} + BCE_{true}}{2}$
- For the Generator:
  * $\beta = D(x, G(\mathbf{x}, \mathbf{z} = 0))$ but $\alpha = 1$ because the Generator aims to fool the Discriminator.
  * $L1$ loss with $\lambda = 100$.

Generator

U-Net based architecture:

Skip connections between downsampling and upsampling layers.

Hyperbolic tangent as activation function in the last layer, predictions in the input range $[-1, 1]$.

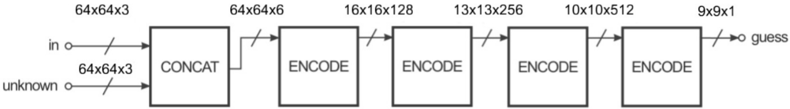


PatchGAN architecture:

Type of CNN which classify small $N \times N$ patches of the input image at time, rather than the full image.

$N \times N = 46 \times 46$.

Sigmoid activation function in the last layer, results in range of probabilities $[0, 1]$.



Why WGAN?

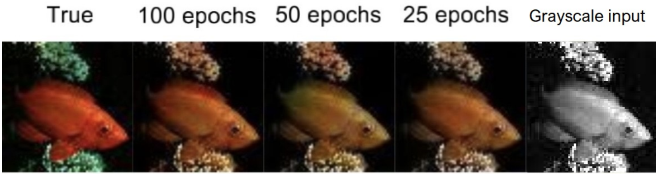
Università degli Studi di Padova

Wasserstein GANs

Generative model implemented to improve original GANs.

- P_r: probability distribution of real images.
- P_g: probability distribution computed by the generator,

GOAL: make these distributions as similar as possible.

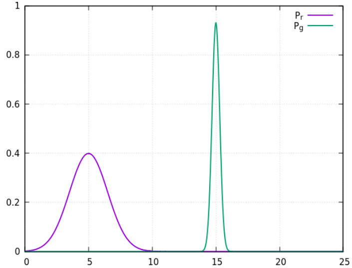


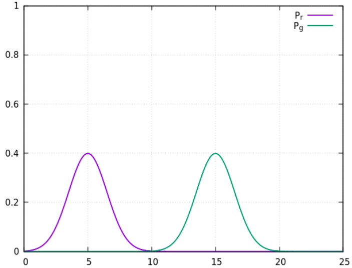
Generic GANs

Jensen-Shannon distance:

$$JS(P_r, P_g) = \int \log \frac{P_r(x)}{P_m(x)} P_r(x) d\mu(x) + \int \log \frac{P_g(x)}{P_m(x)} P_g(x) d\mu(x), \tag{5}$$

where $P_m = \frac{P_r + P_g}{2}$.

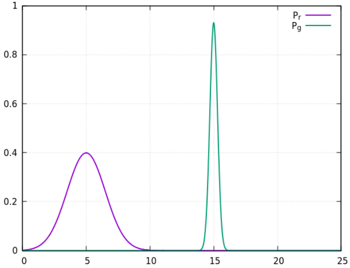


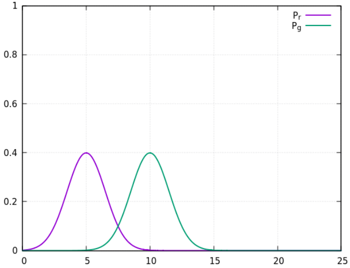


Wasserstain's distance:

$$W(P_r, P_g) = \max_{||f||<1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)]; \qquad (6)$$

- **Discriminator**: aims to *maximize* the separation between these two terms
- **Generator:** tries to *minimize* their difference





Implementation:

- Discriminator $\Rightarrow$ Critic: no sigmoid function,
- Lipschitz continuous $f \Rightarrow$ weight clipping $C = 0.01$,
- more training of Critic function.

PROS

1. meaningful loss: termination criteria,
2. training stability,
3. prevent mode collaps.

CONS

1. longer training time.

Results for patchGAN

Università degli Studi di Padova

Experiment settings:

- Train for 100 epochs
- NVIDIA T4 Tensor Core GPU

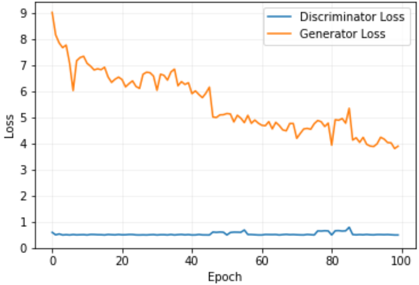


Figure: Loss values for different epochs of training for patchGAN model

PatchGAN image comparison

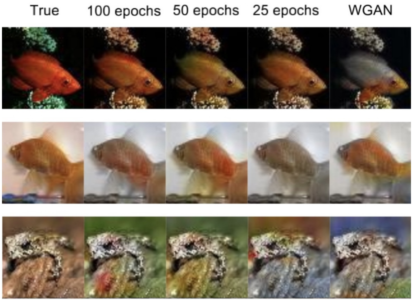


Figure: Good predictions results from the patchGAN model on the test set, according to the epoch of training

Figure: Wrong predictions for patchGAN

Results for WGAN

Experiment settings:

- Train for 100 epochs
- NVIDIA T4 Tensor Core GPU

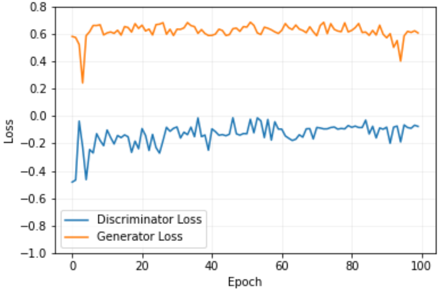


Figure: Loss values for different epochs of training for WGAN model

Figure: Good predictions results from the WGAN model on the test set, according to the epoch of training

Figure: Wrong predictions results for WGAN

Conclusions

What has been achieved:

1. implementation of a full *Generative Adversarial Network* for image colorization,
2. patchGAN vs WGAN:
   - mode collapse prevent, ✓
   - lack of meaning. ✗
3. Simplicity of code, easy to train.

Further future enhancements:

- tuning of patch-sizes,
- tuning of other hyperparameters,
- incorporation of a pretrained model for image reconition.