Image Restoration with Deep Generative Models

Raymond A. Yeh*, Teck-Yian Lim*, Chen Chen,
Alexander G. Schwing, Mark Hasegawa-Johnson, Minh N. Do

Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign

ICASSP 2018

Overview

Image restoration refers to the task of recoving an image from a corrupted sample

Overview

Image restoration refers to the task of recoving an image from a corrupted sample

Examples:

- Inpainting

- Denoising
- etc.

Overview

Image restoration refers to the task of recoving an image from a corrupted sample

Examples:

- Inpainting

- Denoising
- etc.

Task is generally ill-posed

Problem Formulation

Task:
Let y denote the observed image, x^* be the original unobserved image, A a known generative operator A, and noise ϵ.

$$y = A(x^*) + \epsilon,$$

We seek to recover $\hat{x}$ with an objective of the form

$$\hat{x} = \underset{x}{\operatorname{argmin}}\, d(y, A(x)) + \lambda R(x)$$

Where $R(\cdot)$ is some prior, and $d(\cdot)$ is some distance metric(*e.g.* p-norm).

Background I

Traditional Approach:

- Hand designed prior, R, (*e.g.* TV, Low-rank, sparsity, etc.)
- Solve the objective function with some solver
- **Disadvantage:** Priors tend to be simple, generally unable to capture complicated structures in data

Background II

Data-driven, direct:

- Train a deep network, $h(\cdot;\Theta)$ on clean and corrupted pairs in training set $\mathcal{D}$, that maps the corrupted measurements directly predict a clean version.

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \left\| x_i - h(y_i;\Theta) \right\|_p + \lambda \left\| \Theta \right\|, \qquad \forall (x_i, y_i) \in \mathcal{D}$$

Output image:

$$\hat{x} = h(y;\Theta^*)$$

- **Disadvantages:** New model needs to be trained for each new corruption

Overview of Generative Adversarial Nets I

Formulated as a 2-player minimax game between a Generator G and discriminator D with value function $V(G, D)$ where,

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - D(G(z))]$$

Overview of Generative Adversarial Nets I

Formulated as a 2-player minimax game between a Generator G and discriminator D with value function $V(G, D)$ where,

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - D(G(z))]$$

Intuitively,

- D is a classifier that predicts if the given input belongs to the training dataset
- G is a function that generate signals that are able to fool D from a random latent variable z

Overview of Generative Adversarial Nets I

Formulated as a 2-player minimax game between a Generator G and discriminator D with value function $V(G, D)$ where,

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - D(G(z))]$$

Intuitively,

- D is a classifier that predicts if the given input belongs to the training dataset
- G is a function that generate signals that are able to fool D from a random latent variable z

Note that GANs do not model p_x explicitly.

Credit: Goodfellow *et al.* NIPS 2014

Overview of Generative Adversarial Nets II

Convincing faces generated by fully convolutional GANs (DCGAN)

Credit: Radford *et al.* ICLR 2016

Our Proposed Method I

Leveraging the success of GANs, we combine the flexibility of traditional approaches together with the power of a data-driven prior.

Our Proposed Method I

Leveraging the success of GANs, we combine the flexibility of traditional approaches together with the power of a data-driven prior.

Ideally, we would like to solve the following MAP problem,

$$\underset{x}{\operatorname{argmin}} \|y - Ax\|_p + \lambda \log p_X(x)$$

Our Proposed Method I

Leveraging the success of GANs, we combine the flexibility of traditional approaches together with the power of a data-driven prior.

Ideally, we would like to solve the following MAP problem,

$$\underset{x}{\operatorname{argmin}} \|y - Ax\|_p + \lambda \log p_X(x)$$

However, this cannot be done naively with GANs as p_x is not modelled explicitly.

Our Proposed Method II

Objective function:

$$\hat{z} = \arg\min_{z} \|y - A(G(z))\|_p$$
$$+ \lambda\left(\log(1 - D(G(z))) - \log(D(G(z)) + \log(p_{\mathbf{z}}(z)) \right)$$

Our Proposed Method II

Objective function:

$$\hat{z} = \arg\min_{z} \|y - A(G(z))\|_p$$
$$+ \lambda\Big(\log(1 - D(G(z))) - \log(D(G(z)) + \log(p_{\mathbf{z}}(z))\Big)$$

- the first term is the reconstruction loss or the data fidelity term

Our Proposed Method II

Objective function:

$$\hat{z} = \arg\min_{z} \|y - A(G(z))\|_p$$
$$+ \lambda \left(\log(1 - D(G(z))) - \log(D(G(z)) + \log(p_{\mathbf{z}}(z)) \right)$$

- the first term is the reconstruction loss or the data fidelity term
- the second term is our proposed data-driven prior.

Objective function:

$$\hat{z} = \arg\min_{z} \|y - A(G(z))\|_p + \lambda\Big(\log(1 - D(G(z))) - \log(D(G(z)) + \log(p_{\mathbf{z}}(z)) \Big)$$

- the first term is the reconstruction loss or the data fidelity term
- the second term is our proposed data-driven prior.
- We solve for $\hat{z}$, initialized randomly, using gradient descent variants (*e.g.* ADAM).

Our Proposed Method II

Objective function:

$$\hat{z} = \arg\min_{z} \|y - A(G(z))\|_p$$
$$+ \lambda\left(\log(1 - D(G(z))) - \log(D(G(z)) + \log(p_{\mathbf{z}}(z))\right)$$

- the first term is the reconstruction loss or the data fidelity term
- the second term is our proposed data-driven prior.
- We solve for $\hat{z}$, initialized randomly, using gradient descent variants (*e.g.* ADAM).
- Finally $\hat{x} = G(\hat{z})$, and optional blending step can also be applied if desired.

Our Proposed Method - Assumptions

Assumptions:

- we know the class of images we are restoring
- we have a corresponding well-trained generator G and discriminator D for this class of images

Justification of Regularizer

Ideally we would like to use $p_X(x)$ as the prior. However, this is not available for GANs. For a fixed G, the optimal discriminator D for a given generator G is

$$D^*(x) = \frac{p_{\mathbf{X}}(x)}{p_{\mathbf{X}}(x) + p_{\mathbf{G}}(x)},$$

Justification of Regularizer

Ideally we would like to use $p_X(x)$ as the prior. However, this is not available for GANs. For a fixed G, the optimal discriminator D for a given generator G is

$$D^*(x) = \frac{p_{\mathbf{X}}(x)}{p_{\mathbf{X}}(x) + p_{\mathbf{G}}(x)},$$

Rearranging terms,

$$\begin{aligned}\log(p_{\mathbf{X}}(x)) &= \log(D(x)) - \log(1 - D(x)) \\ &\quad + \log(p_{\mathbf{Z}}(z)) + \log\left(\left|\frac{\partial z}{\partial x}\right|\right),\end{aligned}$$

where $p_{\mathbf{G}}(x) = p_{\mathbf{Z}}(z)\left|\frac{\partial z}{\partial x}\right|$. Since $\left|\frac{\partial z}{\partial x}\right|$ is intractable to compute, we assume it to be constant.

Choice of A

Finally we need to choose an A for the restoration task

A **should:**

- reflect the *forward* operation that generates the corruption
- sub-differentiable

Choice of A

Finally we need to choose an A for the restoration task

A **should:**

- reflect the *forward* operation that generates the corruption
- sub-differentiable

For specific tasks:

- **Image Inpainting:** (weighted) masking function
- **Image Colorization:** RGB to HSV conversion, using only V (RGB to grayscale)
- **Image Super Resolution:** Down sampling operation
- **Image Denoising:** Identity
- **Image Quantization:** Identity. Ideally, a step function might make sense but it produces no meaningful gradients

Datasets and Corruption Process

Dataset:

- GAN trained on CelebA dataset
- Faces were aligned and cropped to 64×64

Datasets and Corruption Process

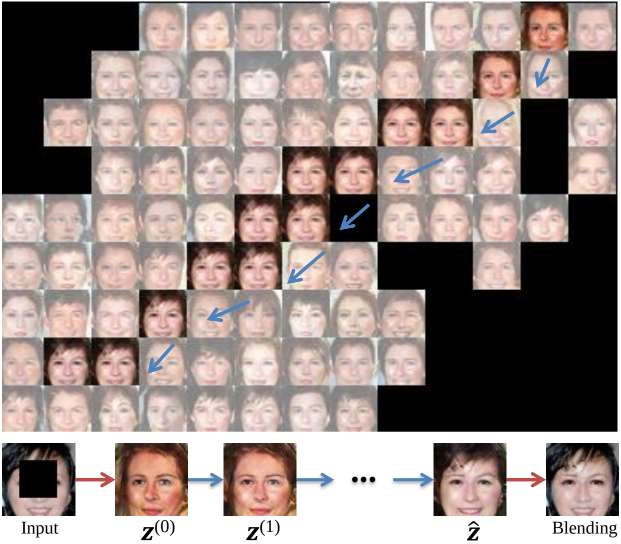
Dataset:

- GAN trained on CelebA dataset
- Faces were aligned and cropped to 64×64

Corruption process:

- **Semantic Inpainting:** The corruption method is a missing center patch of 32×32;
- **Colorization:** The corruption is the standard grayscale conversion;
- **Super Resolution:** The corruption corresponds to downsampling by a factor of 4;
- **Denoising:** The corruption applies additive Gaussian noise, with standard deviation of 0.1 (pixel intensities from 0 to 1);
- **Quantization:** The corruption quantizes with 5 discrete levels per channel.

Visualization of Optimization for Inpainting



Credit: Yeh *et al.* CVPR 2017

Results

Table: Quantitative comparison on image restoration tasks using SSIM and PSNR(dB).

Applications	Inpainting		Colorization		Super Res		Denoising		Quantization	
Metric	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
TV[a]	0.7647	23.10	-	-	0.6648	21.05	0.7373	21.97	0.6312	20.77
LR[b]	0.6644	16.98	-	-	**0.6754**	**21.45**	0.6178	18.69	0.6754	20.65
Sparse[c]	0.7528	20.67	-	-	0.6075	20.82	**0.8092**	**23.63**	**0.7869**	**22.67**
Ours	**0.8121**	**23.60**	0.8876	20.85	0.5626	19.58	0.6161	19.31	0.6061	19.77

Results

Table: Quantitative comparison on image restoration tasks using SSIM and PSNR(dB).

Applications	Inpainting		Colorization		Super Res		Denoising		Quantization	
Metric	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
TV[a]	0.7647	23.10	-	-	0.6648	21.05	0.7373	21.97	0.6312	20.77
LR[b]	0.6644	16.98	-	-	**0.6754**	**21.45**	0.6178	18.69	0.6754	20.65
Sparse[c]	0.7528	20.67	-	-	0.6075	20.82	**0.8092**	**23.63**	**0.7869**	**22.67**
Ours	**0.8121**	**23.60**	0.8876	20.85	0.5626	19.58	0.6161	19.31	0.6061	19.77

Other than inpainting, our method seems to perform poorly under these metrics.

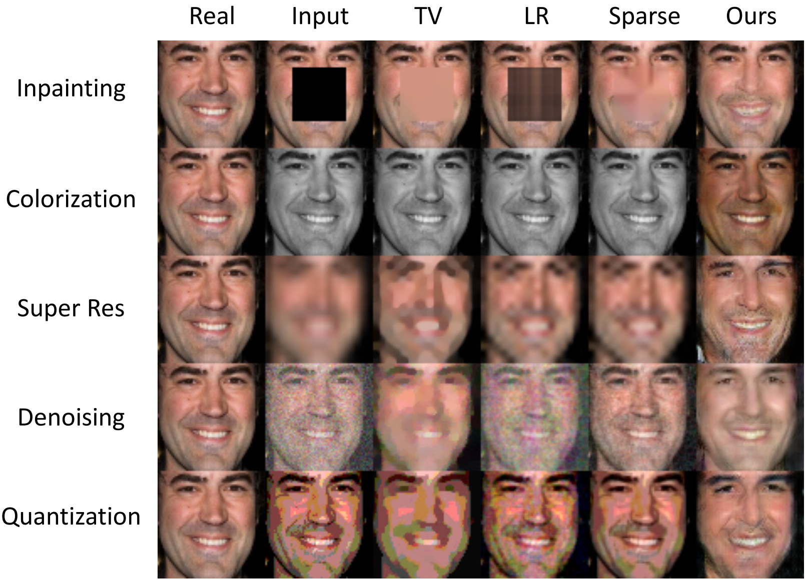
But is that the full story?

[a] Afonso *et al*. TIP 2011

[b] Hu *et al*. PAMI 2013

[c] Elad *et al*. CVPR 2006, Yang *et al*. TIP 2010

Qualitative Results I



Qualitative Results II

Real Input TV LR Sparse Ours

Inpainting

Colorization

Super Res

Denoising

Quantization

Conclusion

Contributions:

- Using GANs as a data-driven prior
- Same model can be used for different problems (no re-training!)
- Not restricted to a specific generative network

Conclusion

Contributions:

- Using GANs as a data-driven prior
- Same model can be used for different problems (no re-training!)
- Not restricted to a specific generative network

Limitations and potential improvements:

- Current GANs are not yet able to handle general images
- Better initial z, perhaps with a LUT or another deep network?

Questions?

Code and more examples at:

https://goo.gl/vNokXj