SOLVING LINEAR INVERSE PROBLEMS USING GAN PRIORS

Kartik Gokhale, Pulkit Agarwal

May 2022

Contents

1 Introduction

Linear inverse problems aim to solve the following equation

$$y = Ax$$

where A is the sensing matrix. When y has fewer measurements than the number of data elements, it becomes a compressed sensing problem. Under these conditions, obtaining a solution for x becomes an underspecified problem. However, the theory behind compressed sensing allows us to recover x accurately when A satisfies some properties and x is assumed to be sparse in some basis, which is a fair assumption for natural images in DCT basis.

However sparsity is not the best we can do. In particular, there might exist sparse representations that are nowhere close to a natural image (take any random sparse matrix). We seek to improve this prior using a Projected Gradient Descent Generative Adversarial Network (PGDGAN). GAN generated priors are able to capture more (further restrict the set of feasible images) than sparsity alone. However, GAN-based methods suffer from convergence problems, and require random restarts from various initial points. This is where PGDGAN provides better results with added theoretical guarantee

Thus, the PGDGAN method serves to improve upon the existing GAN methods by providing provable guarantees.

1.1 PGDGAN algorithm

This is a 2-step algorithm. It primarily operates on a generative model, which is assumed to be a differentiable function from : $R^k \rightarrow R^n$ that maps a standard normal vector $z \in R^k$ to the high dimensional sample space $G(z) \in R^n$.

The two steps are

1. The first step is simply an application of a gradient descent update rule on the loss function
2. In the projection step, we find the closest approximation from the range of our generator to the current estimate (projection of the estimate onto the span of the Generator). Here we assume that the desired set of natural images can be closely approximated by the generator span.

2 Experimental Results

2.1 PGDGAN Reconstruction

We have implemented the PGDGAN algorithm for the celebA dataset. The GAN used is a pre-trained deep convolutional GAN, which has been trained on nearly 20k images. 64 of the images that have not been seen by the GAN model during training are being used for reconstruction using the PGD algorithm. Our code can be found in the file "pgdgan.py", and it requires tensorflow version 1.0.1 to run successfully.

The outer gradient descent step of the PGD algorithm has been implemented for T = 10 iterations. To implement the inner gradient descent, which searches for the optimal reconstruction in the span of the GAN generator, we have invoked a call to the file used by the author for the same. Unfortunately, the file contains an error, due to which the code is only able to work for one iteration of the outer gradient descent loop.

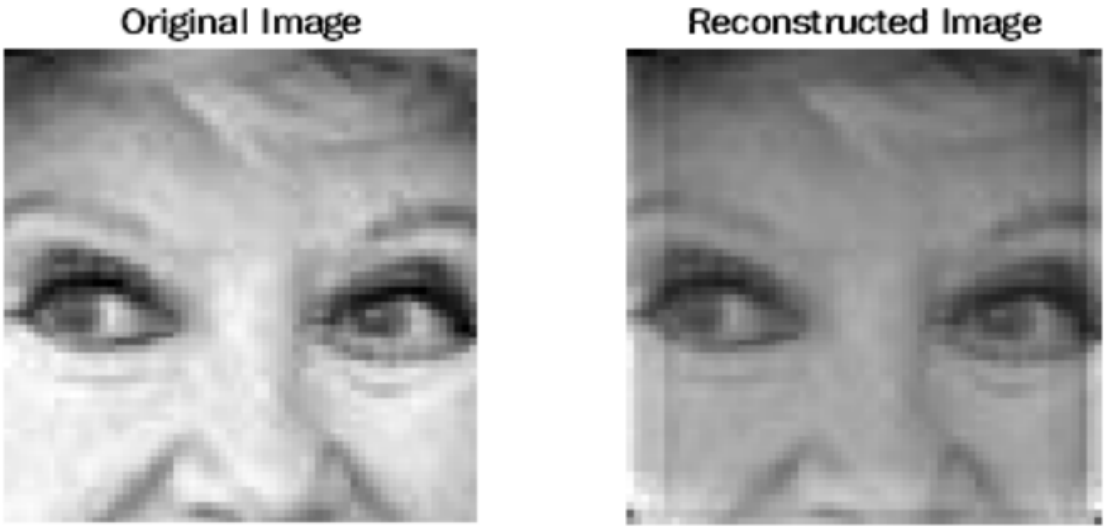
We have also implemented a vanilla GAN on MNIST, but we decided against training it since the author has not given any way to implement the inner gradient descent step for the same.

2.2 Lasso Reconstruction

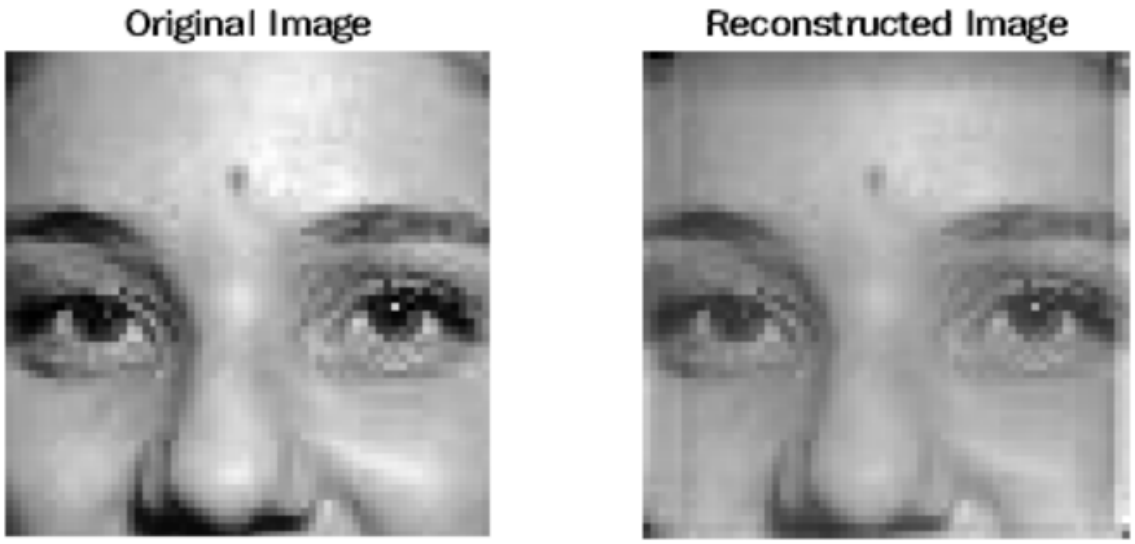
The following results are those obtained by solving the classic LASSO regression problem assuming sparsity prior on x. We have centrally cropped the image into a 64×64 matrix, and have applied reconstruction on grayscale 8×8 patches of the same. Merging the reconstructed patches is done by taking average over the overlapping pixels.

To reproduce the results, run the file "ISTA.m" present inside the folder "Lasso (ISTA)". The image being reconstructed has been hardcoded in the file, so the name of the image needs to be changed appropriately to get the corresponding reconstruction. All 64 test images are present in the folder "Lasso (ISTA)/celebAtest/".

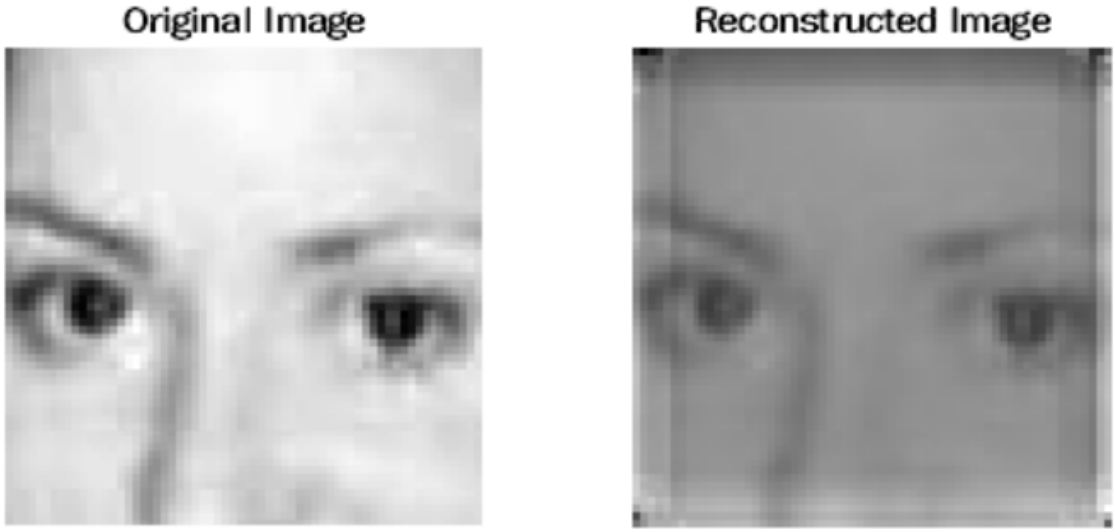
2.2.1 Picture 1



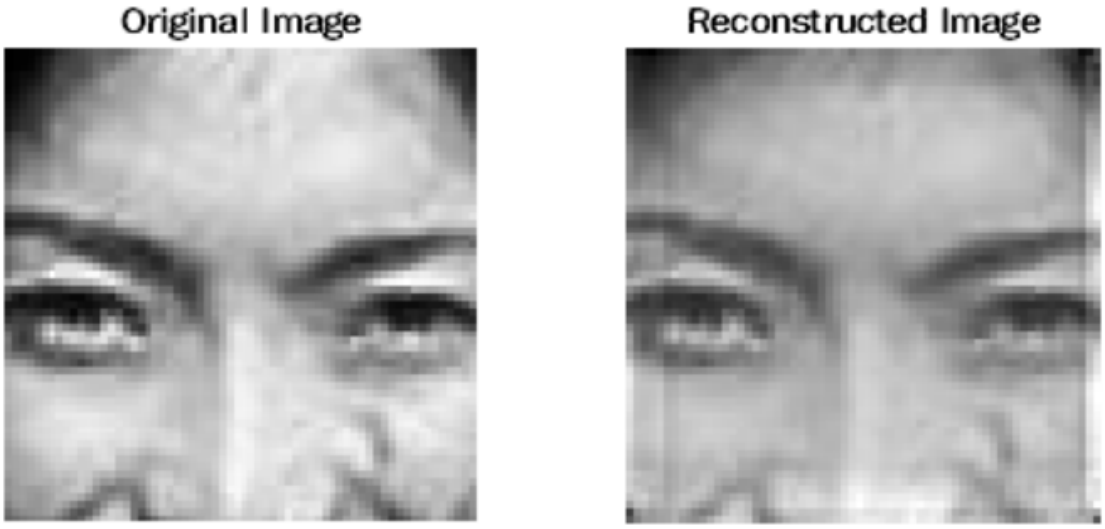
2.2.2 Picture 2



2.2.3 Picture 3



2.2.4 Picture 4



3 Conclusions

Based on our work, we came up with the following conclusions:

- To perform LASSO reconstruction, using ISTA patch-wise gives good results. In particular, if we wish to use some package like l1_ls also, we should do patch-wise reconstruction. This is helpful because of the large size of the vector to be reconstructed, which will make creation of the corresponding 2D DCT matrix slow and infeasible.
- The main implementation hazard was faced in implementing the inner gradient descent, used to find the projection of the vector on the generator space. Our initial idea was to use a pre-trained GAN and apply direct back-propagation. However, this suffers from 2 major problems. Firstly, there are no decent pre-trained DCGANs available on the celebA dataset, which use tensorflow version 2.x and are also saved in the form of checkpoints. Secondly, even if we had been able to get such a GAN, the back-propagation step would have involved deep calculations.
- Our current implementation, which uses the author's implementation to find the projection, mainly faces problems due to either slightly incorrect code on the author's part, or the use of outdated versions of tensorflow. We conclude that there is a need to upgrade the code to a later version of tensorflow (which would require a step similar to the one suggested above).