

When to use Convolutional Neural Networks for Inverse Problems

Nathaniel Chodosh¹ Simon Lucey^{1,2}
¹Carnegie Mellon University ²Argo AI
 {nchodosh, slucey}@andrew.cmu.edu

Abstract

Reconstruction tasks in computer vision aim fundamentally to recover an undetermined signal from a set of noisy measurements. Examples include super-resolution[7], image denoising[23], and non-rigid structure from motion[13], all of which have seen recent advancements through deep learning. However, earlier work made extensive use of sparse signal reconstruction frameworks (e.g. convolutional sparse coding). While this work was ultimately surpassed by deep learning, it rested on a much more developed theoretical framework. Recent work by Papyan et al. [16] provides a bridge between the two approaches by showing how a convolutional neural network (CNN) can be viewed as an approximate solution to a convolutional sparse coding (CSC) problem. In this work we argue that for some types of inverse problems the CNN approximation breaks down leading to poor performance. We argue that for these types of problems the CSC approach should be used instead and validate this argument with empirical evidence. Specifically we identify JPEG artifact reduction and non-rigid trajectory reconstruction as challenging inverse problems for CNNs and demonstrate state of the art performance on them using a CSC method. Furthermore, we offer some practical improvements to this model and its application, and also show how insights from the CSC model can be used to make CNNs effective in tasks where their naive application fails.

1. Introduction

In computer vision we often deal with signals which contain local spatial or temporal statistical dependencies. Convolution has been demonstrated to be an effective tool for modeling this type of inherent redundancy; notable examples include convolutional sparse coding (CSC) and convolutional neural networks (CNNs). In CSC in particular, one can say that a signal is convolutionally compressible if there exists a convolutional dictionary and a sparse vector such that,

$$\mathbf{z} \approx \mathbf{D}\mathbf{x} . \quad (1)$$

Here \mathbf{z} is the signal, $\mathbf{D} \in \mathbb{R}^{N \times K}$ is the convolutional dictionary, and \mathbf{x} is the sparse signal such that $\|\mathbf{x}\|_0 \ll K$. The convolutionally compressible assumption has been notably applied to many inverse reconstruction problem in vision such as image inpainting, super resolution, and trajectory reconstruction[12, 27, 30].

Inverse problems which use the convolutionally compressible assumption can be expressed in a general form;

$$\arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{M}\mathbf{D}\mathbf{x}\|_2^2 + \lambda\Omega(\mathbf{x}), \quad (2)$$

where \mathbf{y} is the degraded signal such that $\mathbf{y} = \mathbf{M}\mathbf{z}$, and Ω provides a prior on the sparsity of \mathbf{x} . Throughout this paper we shall assume we know the degradation matrix \mathbf{M} a priori.

Recently, however, CNNs have demonstrated their utility for such inverse problems[23, 7]. Unlike CSC methods, however, CNNs make no explicit assumptions about the signal (\mathbf{z}) being convolutionally compressible; but are yet considered state-of-the-art for most inversion problems in vision. In this paper we want to explore if CNNs are always the correct tool for inverse problems in computer vision. In particular, we hypothesize that for inverse problems with particular types of degradation matrices (\mathbf{M}), CNNs are substantially sub-optimal in comparison to CSC inspired approaches. Notable examples include JPEG artifact removal[11], and non-rigid trajectory reconstruction[30].

Bridge between CSC and CNNs: In this paper we shall heavily rely upon a recently established theoretical connection between CSC and CNNs by Papyan *et al.* [16]. Their work shows that a convolutional neural network (CNN) can be interpreted as approximately solving a multi-layer CSC objective. Specifically the authors state that Equation (2) can be approximated by a feed forward CNN when Ω represents a *hierarchical sparsity model*. That is, instead of just assuming that $\mathbf{z} = \mathbf{D}\mathbf{x}$, we further assume that the sparse code \mathbf{x} also has a convolutional structure with respect to a different dictionary, and so on. Given this observation, we can view CNNs as implicitly including the sparsity prior which is made explicit in Equation (2). In this work we use this connection to hypothesize that for certain types of degradation matrices, the implicit convolutional assumption

tions of CNNs fail and in these cases CSC based methods are expected to perform better.

When will CNNs fail? Traditional applications of CNNs to inverse problems involve feeding the corrupted signal \mathbf{y} into the network and predicting the original signal \mathbf{z} . As shown in the work of Pappyan *et al.* [16], this implicitly encodes the assumption that \mathbf{y} is convolutionally compressible. This is a valid assumption for tasks such as super resolution and inpainting. However, in this work, we make the observation that it is possible for \mathbf{M} to have a structure that destroys the convolutional compressibility assumption for \mathbf{y} , despite it still holding for the original signal \mathbf{z} . We predict that for inverse problems where \mathbf{M} has this structure, CSC based models will outperform CNNs through their ability to explicitly model the convolutional structure of the predicted signal separate from the non convolutional structure of the degradation.

Contributions: Our main contribution is bridging the gap between the sparse coding based analysis of CNNs and its practical application, by predicting which types of inverse problems CNNs struggle with and showing how these issues can be overcome using CSC. To validate our claim we provide four pieces of experimental evidence. First, we perform a synthetic inverse problem experiment which highlights how the structure of \mathbf{M} affects CNN performance. Second, we show how JPEG artifact removal (AR) can be expressed as an inverse problem with non-convolutional \mathbf{M} and achieve state of the art performance with our method. In particular we greatly outperform a recent optimization based approach [11] to JPEG AR by explicitly modeling the non-convolutional nature of \mathbf{M} . Third, we identify non-rigid trajectory reconstruction as another inverse problem with non-convolutional \mathbf{M} , again demonstrate state of the art performance on it. With this task we also show how the CSC model can be used to design CNNs which elegantly include non-convolutional \mathbf{M} . Finally, we propose several practical contributions to the application of multi layer CSC models which improve performance in all tasks.

2. Background and Related Work

Traditionally, sparse coding is concerned with solving the optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \|\mathbf{x}\|_0 \\ \text{s.t. } \mathbf{D}\mathbf{x} = \mathbf{y}, \end{aligned} \quad (3)$$

where \mathbf{y} is the signal of interest, \mathbf{D} is known as the dictionary, and \mathbf{x} is the sparse code. Put another way, we are searching for the most parsimonious representation of \mathbf{y} with respect to \mathbf{D} . Unfortunately the problem as stated in (3) is NP-Hard in general and does not allow for noise in

the signal. As a result it is normally relaxed to

$$\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1. \quad (4)$$

which is also the Lagrangian form of LASSO regression[20]. Various algorithms have been developed for this problem [9, 6] and have demonstrated remarkable empirical success in denoising [10], deblurring [8], and other image processing tasks [14]. One of the algorithms traditionally used to solve problem (4) is the Iterative Shrinking and Thresholding Algorithm (ISTA). ISTA works by repeatedly applying the formula:

$$\mathbf{x}^{[k+1]} = \mathcal{S}_\lambda(\mathbf{x}^{[k]} - \mathbf{D}^T(\mathbf{D}\mathbf{x}^{[k]} - \mathbf{y})), \quad (5)$$

where \mathcal{S}_λ is the soft thresholding operator which is the proximal operator of the ℓ_1 regularizer. \mathcal{S}_λ is given by:

$$\mathcal{S}_b(x) = \begin{cases} x - b & x > b \\ 0 & b \leq x \leq -b \\ x + b & x < -b \end{cases}. \quad (6)$$

Recently, Pappyan *et al.* [16] noticed that this operator is identical to the ReLU operator except for the negative case. They then showed that if one adds a non-negativity constraint to equation (4) the proximal operator becomes the ReLU function and the first iteration of the corresponding thresholding algorithm becomes:

$$\text{ReLU}(\mathbf{D}^T \mathbf{y} - \lambda), \quad (7)$$

i.e. the first layer of a neural network. The authors extend this idea to a hierarchy of sparsity constraints in order to model a deep CNN. That is, they require that the sparse code \mathbf{x} itself be sparse with respect to another convolutional dictionary, and so on. Relaxing these constraints leads to the optimization problem

$$\min_{\{\mathbf{x}_i > 0\}} \sum_{i=0}^N \alpha_i \|\mathbf{x}_i - \mathbf{D}_{i+1} \mathbf{x}_{i+1}\|_2^2 + \lambda_i \|\mathbf{x}_i\|_1, \quad (8)$$

where we have introduced weights for the now multiple ℓ_2 terms and let $\mathbf{y} = \mathbf{x}_0$ for simplicity.

Applying the ideas behind ISTA to this new objective gives rise to a layered thresholding algorithm whose first pass is equivalent to a deep neural network. The authors have developed the theory of this model in several publications[19, 18, 1], but have not applied it to the kind of large scale learning problems encountered in vision.

Related Work: Other authors such as Murdock *et al.* [15] have used the hierarchical sparsity model, but were primarily interested in its relation to component analysis and enforcing output constraints. The work of Chodosh *et al.* [4] is most similar to ours, but limited themselves to simply

applying the objective to LiDAR super resolution. In contrast, this work makes a broader claim about CNNs, uses the model to predict where the CSC solution will be most effective, and demonstrates how two new tasks can be formulated as CSC problems. Furthermore we use a more theoretically sound optimization method than Chodosh *et al.*, and in the supplement give evidence that this leads to better performance.

Many authors have used the sparse coding model for image processing tasks. A full summary of this body of work is outside the scope of this section, but we will note some important works such as Elad *et al.* which was the first to show the applicability of sparsity models for denoising [10], Dong *et al.* who introduced adaptive dictionary selection for deblurring [8], and Xu *et al.* who extended the denoising model to better deal with real world image statistics and introduced a prior-learning scheme for non-local self similar models [24, 25].

There has also been some work on using CNNs for inverse problems which does not rely on the CSC framework. For example, Deep Image Prior by Ulyanov *et al.* [22] uses the implicit prior of a CNN to solve inverse problems, however they focus on a no-learning approach which does not use any data. Another notable work is that of Chang *et al.* [17] which learns a projection operator for use in inverse problem optimizations. Their focus is on creating a single network which can be used in multiple inverse problems, rather than the specialized networks we are interested in here.

3. Method

3.1. Applying Sparse Coding to Inverse Problems

Before describing our method in detail, we will first explain how sparse coding is used to solve inverse problems. Recall from the introduction that we are interested in problems of the form

$$\operatorname{argmin}_{\mathbf{x}} \|\mathbf{y} - \mathbf{M}\mathbf{D}\mathbf{x}\| + \lambda\Omega(\mathbf{x}). \quad (9)$$

One can see that by making the substitution $\mathbf{D}_1 = \mathbf{M}\mathbf{D}$ and collecting the remaining terms into $\Omega(\mathbf{x})$ we can transform equation (8) into the form of (9). Therefore once we have recovered the sparse codes $\{\mathbf{x}_i\}$ we can predict the original signal with $\hat{\mathbf{z}} = \mathbf{D}_1\mathbf{x}_1$. However, other authors [13] have noted that more stable recovery can be had by instead predicting

$$\hat{\mathbf{z}} = \mathbf{D}_1\operatorname{ReLU}(\mathbf{D}_2\operatorname{ReLU}(\dots\mathbf{D}_N\operatorname{ReLU}(\mathbf{x}_N))), \quad (10)$$

which enforces the intermediate non-negativity constraints and produces better results. This is the form of prediction we use for the rest of the paper.

3.2. Modifying the Objective

In Section 2 we claimed that $\operatorname{ReLU}(\mathbf{D}\mathbf{y} - \lambda)$ is equivalent to a neural network layer, when in fact the traditional bias vector has been replaced by the scalar λ . We will now provide a modification of the model to remedy this. Consider the following new objective function:

$$\min_{\mathbf{x}_i > 0} \sum_{i=1}^N \frac{\alpha_i}{2} \|\mathbf{x}_{i-1} - \mathbf{D}_i\mathbf{x}_i\|_2^2 + \|\mathbf{b}_i \circ \mathbf{x}_i\|_1, \quad (11)$$

where we penalize each element of \mathbf{x}_i individually by replacing λ with the vector \mathbf{b} , and \circ represents point-wise multiplication. We can now see that the thresholding algorithm has become the standard deep learning $\operatorname{ReLU}(\mathbf{D}^\top\mathbf{y} - \mathbf{b})$. We note that this does *not* change the expressiveness of the model, as there is a one to one correspondence between the two forms. Please refer to the supplement for full details on this and experiments showing its utility.

3.3. Unrolling the Optimization

Algorithm 1: Our method for solving multi-layer CSC problems, see Equation (14) for a definition of $\hat{\mathbf{g}}_i^{[t]}$

```

for  $i \leftarrow 1$  to  $\ell$  do
   $\mathbf{x}_i^{[0]} \leftarrow 0$ ;
for  $t \leftarrow 1$  to  $T$  do
  for  $i \leftarrow 1$  to  $N$  do
     $\mathbf{x}_i^{[t]} \leftarrow \operatorname{ReLU}(\hat{\mathbf{x}}_i^{[t-1]} - \frac{1}{L_i}(\hat{\mathbf{g}}_{i-1}^{[t]} + \mathbf{b}_i))$ 

```

In their original work Papyan *et al.* present the LIST (layered iterative soft thresholding) algorithm for solving multi-layer CSC problems [16]. This algorithm is elegant, but for practical application we are interested in fast convergence, since each iteration can be quite expensive. To this end, Murdock *et al.* also proposed an algorithm in [15] based on the Alternating Direction Method of Multipliers (ADMM). This was effective but relied on the assumption that $\mathbf{D}^\top\mathbf{D} = \mathbf{I}$ for all dictionaries, which was not satisfied in practice. This formulation and assumption was also used by Chodosh *et al.* [4]. We avoid this issue by replacing ADMM with a recently developed convex optimization algorithm from Xu *et al.* [26].

Applying the algorithm of Xu *et al.* to Equation (11) gives us the iterative update:

$$\mathbf{x}_i^{[t]} = \operatorname{argmin}_{\mathbf{x}_i} (\hat{\mathbf{g}}_i^{[t]})^\top (\mathbf{x}_i - \hat{\mathbf{x}}_i^{[t-1]}) + \frac{L_i}{2} \|\mathbf{x}_i - \hat{\mathbf{x}}_i^{[t-1]}\|^2 + \|\mathbf{b}_i \circ \mathbf{x}_i\|_1, \quad (12)$$

where $\hat{\mathbf{x}}_i^{[t-1]} = \mathbf{x}_i^{[t-1]} + w_i(\mathbf{x}_i^{[t-1]} - \mathbf{x}_i^{[t-2]})$ is an extrapolation of the previous estimate of \mathbf{x}_i , $\hat{\mathbf{g}}_i^{[t]} = \nabla f_i^k(\hat{\mathbf{x}}_i^{[k-1]})$ is the gradient of the smooth part of the objective w.r.t to the i th block evaluated at $\hat{\mathbf{x}}_i^{[k-1]}$ and L_i is a step size parameter.

Equation (12) admits a closed form solution which can be derived by examining the Karush-Kuhn-Tucker (KKT) conditions for optimality. This leads to

$$\mathbf{x}_i^{[t]} = \text{ReLU}(\hat{\mathbf{x}}_{i-1}^{[t-1]} - \frac{1}{L_i}(\hat{\mathbf{g}}_{i-1}^{[t]} + \mathbf{b}_i)), \quad (13)$$

and the gradient expression

$$\hat{\mathbf{g}}_i^{[t-1]} = \begin{cases} \alpha_i \mathbf{D}_i^\top (\mathbf{D}_i \hat{\mathbf{x}}_i^{[t-1]} - \mathbf{x}_{i-1}^{[t]}) \\ + \alpha_{i+1} (\hat{\mathbf{x}}_i^{[t-1]} - \mathbf{D}_{i+1} \mathbf{x}_{i+1}^{[t-1]}), i < N \\ \alpha_i \mathbf{D}_i^\top (\mathbf{D}_i \hat{\mathbf{x}}_i^{[t-1]} - \mathbf{x}_{i-1}^{[t]}), i = N \end{cases} \quad (14)$$

We leave the full derivation of the above results to the supplementary material. Putting both of these equations together gives our full method for solving the multi-layer CSC objective which is summarized in Algorithm 1.

3.4. Learning Dictionaries and Parameters

In this section, we discuss how the dictionaries \mathbf{D} are obtained in our multi-layered sparse coding pipeline. Early work in sparse coding used handcrafted dictionaries such as the discrete cosine transform or wavelet packers [5], but these were ultimately outperformed by dictionaries learned by optimizing over real world datasets using algorithms such as KSVD [2]. Extensions of these algorithms to the multi-layer case have recently been proposed by Sulam and Elad [19], which we compare against in section 4.2.

Let the function $\mathcal{H}^{(T)}(\mathbf{y})$ represent our algorithm truncated to T iterations. For simplicity we will let the output of \mathcal{H} be the highest level code \mathbf{x}_N . As described in Section 3.1, we can write our predicted signal as

$$\hat{\mathbf{z}} \approx \mathbf{D}_1 \text{ReLU}(\dots \mathbf{D}_N \text{ReLU}(\mathcal{H}^{(T)}(\mathbf{y}))). \quad (15)$$

Given a reconstruction loss function $\mathcal{L}(\hat{\mathbf{z}}, \mathbf{z})$ and M training examples, we can express the goal of dictionary learning as:

$$\min_{\{\mathbf{D}_i, \mathbf{b}_i, L_i, w_i\}_{i=1}^N} \sum_j^M \mathcal{L}(\hat{\mathbf{z}}_j, \mathbf{z}_j). \quad (16)$$

Inspired by the success of deep learning we propose to solve this objective by treating the entire inference algorithm as a black box, and optimizing the parameters with stochastic gradient descent (SGD) or one of its variants.

4. Experiments

4.1. Diagonal vs Block Diagonal Inverse Problems

In this section we perform a synthetic experiment to show how the structure of the measurement matrix greatly

affects the performance of a naive CNN approach to solving inverse problems. First we provide some intuition as to why this might be the case. The structure of a CNN encodes the belief that the input signal has a convolutional structure, that is, an individual sample is highly correlated with its neighbors. When an image is degraded by missing data or blurring, these correlations are modified, but still persist. In these cases one would expect a CNN to perform well at reconstructing the original signal and many results confirm this [3, 28, 21, 22]. In these tasks the corresponding measurement matrix is diagonal or convolutional.

Now consider an inverse problem where the measurement matrix is block diagonal. For simplicity we will assume that the blocks are arranged such that each 4x4 patch of the image is multiplied by an arbitrary linear transformation. We will refer to this problem as block recovery. From the perspective of solving an inverse problem, this situation is identical to inpainting or deblurring, but we would expect it to be much more difficult for a CNN.

To test this hypothesis we performed a synthetic experiment where we interpolate smoothly between inpainting and block recovery and show that the performance of a CNN degrades as we move closer to block recovery. Furthermore we will show how our proposed CSC based method is relatively unaffected by this difference in measurement matrix structure, even without retraining. Specifically we define a set of learning tasks parameterized by the scalar α . In each task the input is a measurement matrix $\mathbf{M}_i(\alpha)$ and the measured image $\mathbf{y}_i = \mathbf{M}_i(\alpha)\mathbf{z}_i$, and the goal is to recover the original image \mathbf{z}_i . We let $\mathbf{M}_i(\alpha) = \mathbf{M}_{inpainting}(1 - \alpha) + \mathbf{M}_{block}\alpha$ where $\mathbf{M}_{inpainting}$ is a diagonal matrix which zeros out some pixels, and \mathbf{M}_{block} is the previously described block diagonal transform. We use as data the Berkeley Segmentation Dataset (BSD-500) which provides a canonical set of high resolution images. Our baseline CNN is shown in figure 1, and to compare we use a three layer CSC model. We note that for the CNN we have to retrain for different values of α but for the CSC model we simply train once at $\alpha = 0.5$. The results can be found in figure (2), validating our hypothesis. While interesting, this experiment wouldn't be useful if it didn't lead us to some practical application. The following experiments aim to do this by showing how two different computer vision tasks have this block diagonal structure.

4.2. JPEG Artifact Removal

In the previous section we showed how when the measurement matrix is block diagonal, we expect our CSC based method to perform better than a CNN. The synthetic experiment was contrived to demonstrate this, but we will now see how removing artifacts created by the original JPEG algorithm in fact has a similar form.

The compression algorithm operates on 8x8 blocks of

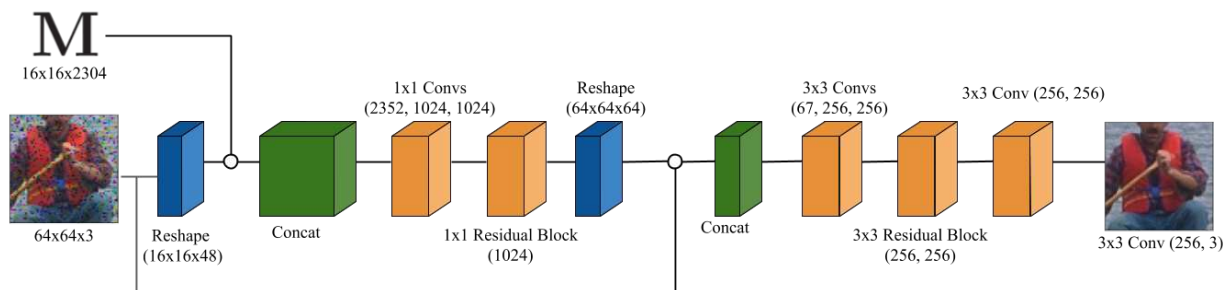


Figure 1: Our baseline CNN architecture. It is based on the JPEG Artifact Removal architecture of Zheng *et al.* Due to the high dimensionality of the block-diagonal transformation we first do a patch wise encoding step. The patch-wise encoder operates on a single 4x4 patch and corresponding block of the transformation, it encodes it to a 64 channel 4x4 image which concatenated with the original image and then fed through a standard CNN. The residual blocks are of the same form as Zheng *et al.*

Dataset	Quality	Fuet <i>al.</i> [11]			Zhenget <i>al.</i> [29]			Ours		
BSD500	5	0.0466	26.8	0.768	0.0381	28.5	0.853	0.0360	29.05	0.860
	10	0.0336	29.7	0.845	0.0269	32.1	0.912	<u>0.0273</u>	<u>31.6</u>	0.892
	20	0.0276	31.5	0.879	0.0249	32.4	0.906	0.0218	33.5	0.918
Parameters ($\times 10^5$)		3.74			106.6			.782		

Table 1: Results on BSD500 for reconstruction trained and tested at a specific quality factor, the numbers in each cell are RMSE, PSNR, and SSIM. Best results are in bold face, and results which are close to the best are underlined

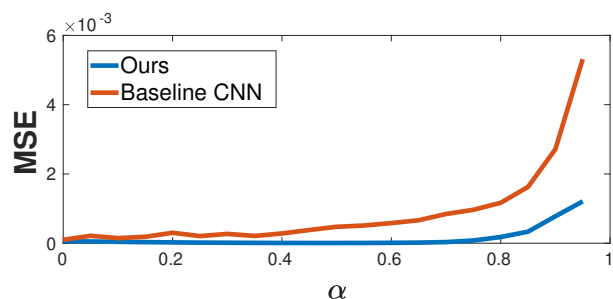


Figure 2: The results of the synthetic experiment on BSD500. For each value of α we train our CNN 5 times to convergence and then report the average of the top three validation results. In contrast we only train our model one time at $\alpha = 0.5$ which explains why the performance of our model peaks there since the hyper-parameters have been learned for that value.

the image independently. It is comprised of five steps: conversion to $YCbCr$ space, down-sampling of chroma channels, DCT-II transformation, rounding of DCT-II coefficients, and finally lossless compression of the remaining coefficients. The amount of rounding is controlled by a

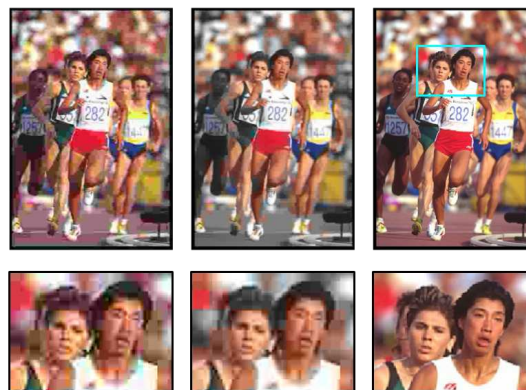


Figure 3: Top row, from left to right: an image from the Berkeley Segmentation Database compressed using JPEG, the same image degrade by our linear approximation, the original image. Bottom row, a zoom of the top images to show the blocking artifacts.

scalar parameter known as the quality factor, which varies from 1 to 100. The first three of these steps are linear transformations, and the final step is lossless and can

therefore be discarded for our purposes. The fourth step creates the well known blocking artifacts, but is not linear. However as can be seen in Figure 3 the majority of the artifacts are created by the frequency components which are rounded to zero. Therefore we can make an approximation to the JPEG algorithm by ignoring the rounding on frequency components which are not zeroed. This approximation is *locally linear* and can be expressed as $M_{JPEG} = M_{zeroing} M_{DCT} M_{downsample}$, where we have opted to perform the experiment in YC_bC_r space.

Since the algorithm processes each block separately their collective action is very similar to the previous synthetic experiment. As such a CNN would be expected to perform very poorly if applied directly to the output of the algorithm, the DCT-II coefficients. To avoid this issue, other authors choose to manually undo the the DCT and down-sampling operations, effectively applying the CNN to the decompressed image. This is the approach taken by Zheng *et al.* [29] which we will use as a comparison CNN architecture. We also compare against the recent work of Fu *et al.* [11], who also chose to apply convolutional sparse coding to this problem. However, there are several key differences between our approaches: First Fu *et al.* bases their architecture on *single layer* CSC. Second, Fu *et al.* applies LISTA to the objective, resulting in a network which is inspired by an optimization instead of explicitly performing it. Finally and most importantly, Fu *et al.* do not model the JPEG degradation and instead use the decompressed image as their input. As a result of these differences we find that our method uses far fewer parameters and achieves significantly better results. These findings are consistent with our hypothesis that explicit modeling of the measurement matrix is key when dealing with non-convolutional corruption.

We perform all experiments on the BSD-500 dataset for a range of quality factors. We also perform all experiments on the full YC_bC_r image instead of on only the luminance channel. Since the method of Fu *et al.* [11] was designed for only a single channel input, we double the number of features in each convolution to compensate for the larger input.

In the first experiment we fix a quality factor and train each method to predict the original image from the compressed one. The results are show in table 1, which demonstrates that when the measurement matrix is closer to block diagonal our method achieves state of the art performance at quality factors 5 and 20 all while using much fewer parameters than the next closest model. Furthermore we are able to outperform both the deep sparse coding method and traditional multi-layer CSC by a wide margin at all quality factors. This is shown in the table as well as in the results of figure 6. That figure shows the results of a similar experiment except that we use identical model parameters to Sulam *et al.*, to demonstrate the effectiveness of learning

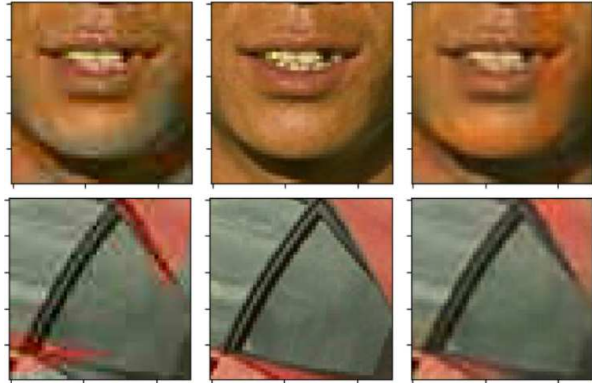


Figure 4: Some example artifact removal results from BSD-500, zoomed in to highlight the detail. From left to right: Degraded input image, target image, our result

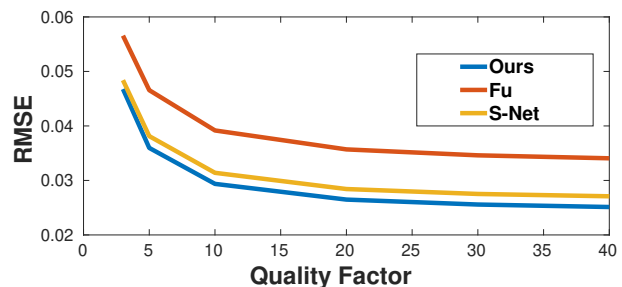


Figure 5: A comparison of our method and that of Zheng *et al.* on the JPEG artifact removal task. Note that as the degradation becomes more severe, that is the blocks of the measurement matrix become less diagonal, we see the same kind of fall off in performance as we did in the synthetic experiment.

the hyper-parameters directly from the data. Finally we performed an experiment where we mismatch the training and testing quality factors in order to measure the generalization of each model. Figure 5 shows the results of this experiment which demonstrate that due to our method specifically modeling the degradation, it outperforms all others by a wide margin when there is a mismatch between testing and training degradation.

4.3. Non-Rigid Trajectory Reconstruction

In non-rigid trajectory reconstruction, one seeks to recover the 3D trajectories of points from their 2D projections. Zhu *et al.* first proposed convolutional sparse coding for this problem in [30]. They demonstrate large performance gains through learning the filters directly from motion-capture data. However this requires manual tuning of the sparsity parameter which can be difficult to get right. In contrast, our method learns this parameter without tun-



Quality Factor	Ours	Sulam <i>et al.</i>
5	0.039	0.080
25	0.069	0.11

Figure 6: From top to bottom: image degraded at QF 5, our method’s output, the method of Sulam *et al.*, the target image. The poor performance of the method of Sulam *et al.* can be explained by the fact that these methods are very sensitive to the choice of parameters. The dictionary is learned with one set of parameters but there is no clear way to modify them once we introduce the JPEG degradation. The table reports RMSE for our method and Sulam *et al.* at quality factors 5 and 25

ing. Before presenting these results we will first give a brief description of the problem.

We refer the reader to Zhu *et al.* [30] for a description of how the trajectory reconstruction problem can be phrased as a sparse coding one, and present here just the equations for the input signal and measurement matrix.

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & & \\ & \ddots & \\ & & \mathbf{M}_F \end{bmatrix}, \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_F \end{bmatrix} \quad (17)$$

$$(18)$$

where the $\mathbf{M}_i, \mathbf{y}_i$ encode the weak perspective cameras and 2D points. They demonstrate that learning the dictionary \mathbf{D} on real motion capture data yields very impressive results which are still the state of the art on this task. Using the 3D ground truth points as supervision, we can train our end to end multi-layer CSC method for this task using the same formulation as described in section 3.4, except now our convolutions are in time instead of space.

In our experiments we replicated the setup of Zhu *et al.* on the CMU motion capture dataset (mocap.cs.cmu.edu).

	NRMSE	RMSE
Zhu <i>et al.</i>	0.025	35.91
14 Layer Trajectory CNN	0.0254	39.98
Ours 3 Layer	0.0179	27.16

Table 2: Results of multi-layer CSC (ML-CSC) on the trajectory reconstruction problem. We report Normalized RMSE in addition to RMSE, to better compare sequences of different scales. Please see [30] for a description of this metric.

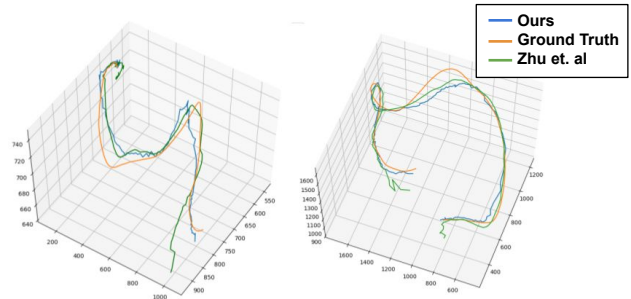


Figure 7: Some example trajectories and our reconstruction results

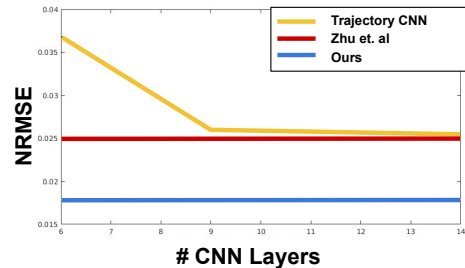


Figure 8: Performance vs Number of layers for our proposed Trajectory CNN

We took each sequence, resampled them all to 30fps, created a synthetic orthographic camera which orbits the center of mass of the tracked points at a rate of π/s , and then cut the tracks into 150 frame sequences to form individual training examples. For validation, we hold out the sequences of subjects 14, 94, and 114, and use the rest for training. We can see from the results in Figure 2 that we are able to outperform those results by a wide margin.

We do not show results for a naive CNN on this task since they fail to produce reasonable predictions. This is in line with our hypothesis that inverse problems such as these are not amenable to simple CNN approaches. Instead we will show how we can derive an modification of the input from the multi layer CSC model that allows a simple CNN to achieve reasonable results on this task.

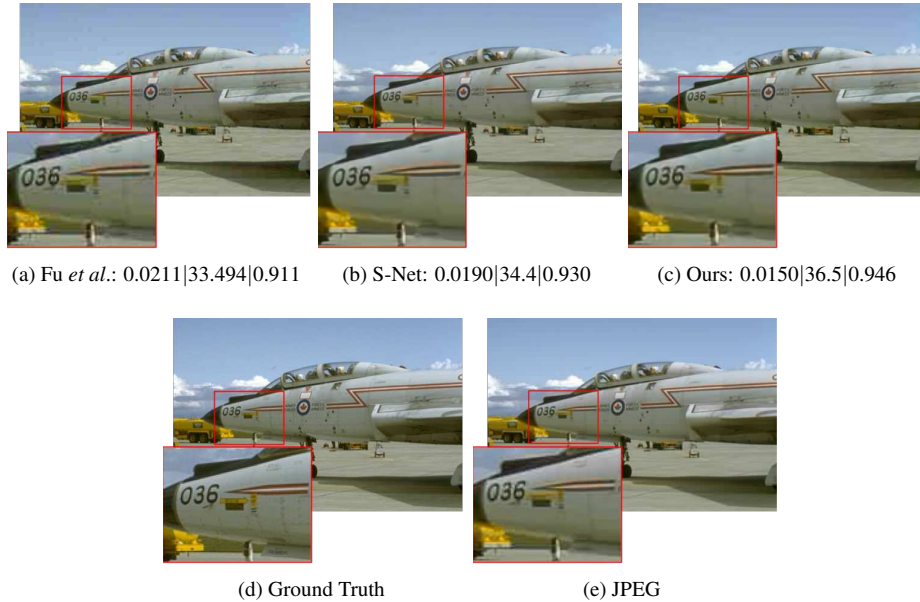


Figure 9: Qualitative results on BSD500. The captions contain RMSE, PSNR, and SSIM measures for each image

Recall from section 3.3 that the first pass of our multi-layer CSC algorithm is equivalent to a feed forward CNN. Specifically we have that on the first iteration:

$$\mathbf{x}_i^{[0]} = \text{ReLU}\left(\frac{1}{L_i} \mathbf{D}_i^\top \mathbf{x}_{i-1}^{[0]} + \mathbf{b}_i\right). \quad (19)$$

However, we also have from section 3.1 that when we have a measurement matrix \mathbf{M} then the first dictionary \mathbf{D}_1 should be replaced by $\mathbf{M}\mathbf{D}_i$. This means that the first layer’s first iteration is given by:

$$\mathbf{x}_1^{[0]} = \text{ReLU}\left(\frac{1}{L_i} \mathbf{D}_i^\top \mathbf{M}^\top \mathbf{y} + \mathbf{b}_1\right). \quad (20)$$

This leads to the observation that the input to the network is effectively $\mathbf{M}^\top \mathbf{y}$ instead of simply \mathbf{y} . We note that this is similar to how in the JPEG AR task, the input to the CNN was not the DCT coefficients but instead the decompressed image. Since the DCT transform is orthogonal, performing the inverse compression is very close to multiplying \mathbf{y} by \mathbf{M}^\top . In general the product $\mathbf{M}^\top \mathbf{y}$ does not contain all of the information required to reconstruct \mathbf{z} , as we saw in the JPEG task as well. But, we will now see that in the non-rigid trajectory reconstruction task enough of the information is preserved that it allows a CNN to perform reasonably.

To give some intuition of why we expect this to work, let \mathbf{p}_i be the original 3D point which is projected to \mathbf{y}_i by orthogonal camera \mathbf{M}_i . Since the trajectories are smooth in time, the \mathbf{p}_i have a convolutional structure appropriate for a CNN. This structure is destroyed when multiplied by \mathbf{M}_i . However, we note that $\mathbf{M}_i^\top \mathbf{M}_i$ has the effect of back

projecting the points to the 3D space, but with the depth set to 1. Since the cameras are smooth in time as well, the resulting trajectory is a distorted version of the original which is smooth. Thus we expect a CNN to perform well on this modified input. Our prediction is confirmed by the results in table 2. This network is able to achieve similar results to the CSC approach but requires many more layers to make up for the lack of iterations as shown in Figure 8.

5. Conclusion

In this work we have used multi-layer CSC and its relationship with CNNs to predict when CNNs will perform poorly on linear inverse problems. We tested this hypothesis on a synthetic task and two real world ones and in all cases we found that when faced with measurement matrices which do not possess a convolutional structure, a naive CNN approach fails, and modification of the input is required for good performance. Specifically we achieve state of the art performance in both JPEG artifact reduction and non-rigid trajectory reconstruction. Furthermore, we used this observation to develop a CNN which can effectively solve the non-rigid trajectory reconstruction task, which previously had not seen a deep learning based approach. Additionally we made practical contributions in the form of 1) modifying the MLSC objective into a weighted version 2) finding an optimization method which converges quickly and does not require any assumptions about the dictionary.

Acknowledgments: This work was supported by the CMU Argo AI Center for Autonomous Vehicle Research.

References

- [1] Aviad. Aberdam, Jeremias. Sulam, and Michael. Elad. Multi-layer sparse coding: The holistic way. *SIAM Journal on Mathematics of Data Science*, 1(1):46–77, 2019.
- [2] Michal Aharon, Michael Elad, Alfred Bruckstein, et al. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311, 2006.
- [3] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. *CoRR*, abs/1605.00075, 2016.
- [4] Nathaniel Chodosh, Chaoyang Wang, and Simon Lucey. Deep convolutional compressed sensing for lidar depth completion. *CoRR*, abs/1803.08949, 2018.
- [5] Ronald R. Coifman, Yves Meyer, Steven Quake, and M. Victor Wickerhauser. Signal processing and compression with wavelet packets. *Wavelets and Their Applications*, page 363379, 1994.
- [6] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.
- [7] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016.
- [8] Weisheng Dong, Lei Zhang, Guangming Shi, and Xiaolin Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *IEEE Trans. Image Processing*, 20(7):1838–1857, 2011.
- [9] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition, 2010.
- [10] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Processing*, 15(12):3736–3745, 2006.
- [11] Xueyang Fu, Zheng-Jun Zha, Feng Wu, Xinghao Ding, and John Paisley. Jpeg artifacts reduction via deep convolutional sparse coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2501–2510, 2019.
- [12] Huiyi Hu, Brendt Wohlberg, and Rick Chartrand. Task-driven dictionary learning for inpainting. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page nil, 5 2014.
- [13] Chen Kong and Simon Lucey. Deep non-rigid structure from motion. *arXiv preprint arXiv:1907.13123*, 2019.
- [14] Julien Mairal, Francis R. Bach, and Jean Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014.
- [15] Calvin Murdock, Ming-Fang Chang, and Simon Lucey. Deep component analysis via alternating direction neural networks. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, pages 851–867, 2018.
- [16] Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *Journal of Machine Learning Research*, 18:83:1–83:52, 2017.
- [17] JH Rick Chang, Chun-Liang Li, Barnabas Póczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all—solving linear inverse problems using deep projection models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5888–5897, 2017.
- [18] J. Sulam, A. Aberdam, A. Beck, and M. Elad. On multi-layer basis pursuit, efficient algorithms and convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019.
- [19] Jeremias Sulam, Vardan Papyan, Yaniv Romano, and Michael Elad. Multilayer convolutional sparse modeling: Pursuit and dictionary learning. *IEEE Trans. Signal Processing*, 66(15):4090–4104, 2018.
- [20] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.
- [21] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 11–20, 2017.
- [22] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9446–9454, 2018.
- [23] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 350–358, 2012.
- [24] Jun Xu, Lei Zhang, and David Zhang. A trilateral weighted sparse coding scheme for real-world image denoising. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2018.
- [25] Jun Xu, Lei Zhang, Wangmeng Zuo, David Zhang, and Xi-angchu Feng. Patch group based nonlocal self-similarity prior learning for image denoising. In *Proceedings of the IEEE international conference on computer vision*, pages 244–252, 2015.
- [26] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.
- [27] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.
- [28] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. Real-time user-guided image colorization with learned deep priors. *CoRR*, abs/1705.02999, 2017.
- [29] Bolun Zheng, Rui Sun, Xiang Tian, and Yaowu Chen. S-net: a scalable convolutional neural network for jpeg com-

pression artifact reduction. *Journal of Electronic Imaging*, 27(4):043037, 2018.

- [30] Yingying Zhu and Simon Lucey. Convolutional sparse coding for trajectory reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):529–540, 2015.