Deep Convolutional Compressed Sensing for LiDAR Depth Completion

Nathaniel Chodosh, Chaoyang Wang, Simon Lucey

Robotics Institute, Carnegie Mellon University
{nchodosh,chaoyanw}@andrew.cmu.edu, slucey@cs.cmu.edu

Abstract. In this paper we consider the problem of estimating a dense depth map from a set of sparse LiDAR points. We use techniques from compressed sensing and the recently developed Alternating Direction Neural Networks (ADNNs) to create a deep recurrent auto-encoder for this task. Our architecture internally performs an algorithm for extracting multi-level convolutional sparse codes from the input which are then used to make a prediction. Our results demonstrate that with only two layers and 1800 parameters we are able to out perform all previously published results, including deep networks with orders of magnitude more parameters.

Keywords: Depth Completion, Super LiDAR, Compressed Sensing, Convolutional Sparse Coding

1 Introduction

In recent years 3D information has become an important component of robotic sensing. Usually this information is presented in 2.5D as a depth map, either measured directly using LiDAR or computed using stereo correspondence. Since LiDAR and stereo techniques yield few samples relative to modern image sensors, it has become desirable to convert sparse depth measurements into high resolution depth maps as shown in Figure 1.

Recent works [10, 14] have directly applied deep networks to depth completion from sparse measurements. However, common network architectures have two drawbacks when applied to this task: 1) They implicitly pose depth completion as finding a mapping from sparse depth maps to dense ones, instead of as finding a depth map that is consistent with the sparse input. This essentially throws away information and we observe that feed forward networks do not learn to propagate the input points through to the output. Qualitative evidence of this can be seen in Figure (1). 2) Common networks are sensitive to the sparsity of the input since they treat all pixels equally, regardless of whether or not they represent samples or missing input. Special CNN networks have been designed to address this problem, but they still do not express the constraints given by the input [14]. In this paper we address both of these issues with a novel deep recurrent autoencoder architecture, which internally optimizes its depth prediction with respect to both sparsity and input constraints. To do this, we have taken

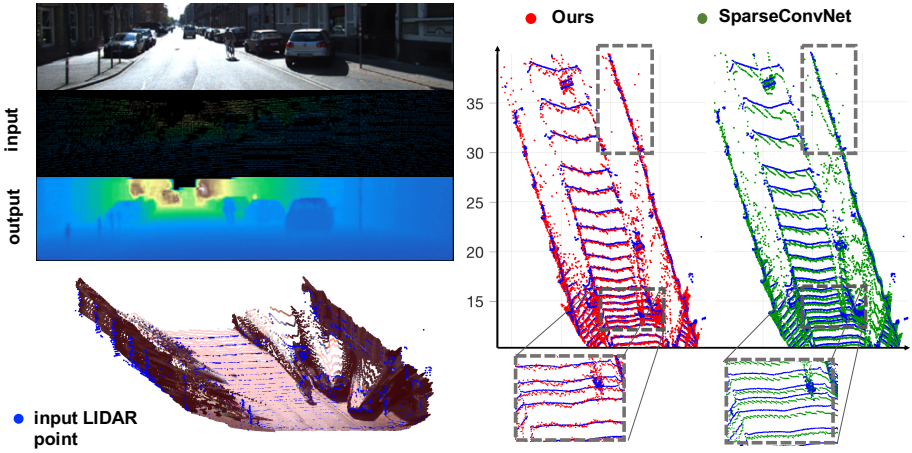

Fig. 1: The bicyclist and bollards can barely be seen in the input map but a clearly represented in the output. Our method also accurately reconstructs very thin objects such as the sign post. On the right it can be seen that our method enforces that its prediction should match the input points while the SparseConvNet systematically underestimates the depth.

inspiration from Compressed sensing (CS) which provides a natural framework for this problem. Formally, CS is concerned with recovering signals from incomplete measurements by enforcing that signals be sparse when measured in an appropriate basis. This basis takes the form of an overcomplete matrix which maps sparse representations to observed signals.

The choice of dictionary is crucial for recovering the signal efficiently, especially when the dimensionality is high. For high resolution imagery data, such as depth maps, multi-layer convolutional sparse coding (CSC) [12] is effective as it explicitly models local interactions through the convolution operator with tractable computational and model complexity. However, none of the existing multi-layer convolutional sparse coding algorithms are designed for learning from sparse ground truth data. This is reflected by the fact that recent works [7, 8] applying CS to depth completion are restricted to using single-level, hand crafted dictionaries. CS has also fallen out of fashion since the existing algorithms have difficult to interpret hyper-parameters, and often do not achieve good performance without careful tuning of these parameters.

Recent developments in the formal analysis of deep learning have shown that convolutional neural networks and convolutional sparse coding are closely related. Specifically it has been shown that CNNs with ReLU activation functions are carrying out a specific form of the layered thresholding algorithm for CSC. Layered thresholding is a simple algorithm for solving multi-layered convolutional sparse coding (ML-CSC) problems, which can be effective when there is little noise and the coherence of the dictionary is high. Motivated by the work of Murdock *et al.* [13], in this paper we propose a network architecture which

encodes a more sophisticated algorithm for ML-CSC. Encoding the ML-CSC objective in a deep network allows us to learn the dictionaries and parameters together in an end to end fashion. We show that by better approximating this objective, we can out perform all published results on the KITTI depth completion benchmark while using far fewer parameters and layers. Furthermore, this work builds on the Alternating Direction Neural Network (ADNN) framework of Murdock *et al.* which gives theoretical insight into deep learning and we believe is a promising new area of research.

To summarize, the main contributions of this paper are:

1. We frame an end-to-end multi-layer dictionary learning algorithm as a neural network. This allows us to effectively learn dictionaries and hyper-parameters from a large dataset. In comparison, existing CS algorithms either use hand crafted dictionaries, separately learned multi-level dictionaries, or are inapplicable to incomplete training data [15], as is our case.
2. Our method allows for explicit encoding of the constraints from the input sparse depth. Current deep learning approaches [10] simply feed in a sparse depth map and rely solely on data to teach the network to identify which inputs represent missing data. Some recent models [14] explicitly include masks to achieve sparsity invariance, but none have a guaranteed way of encoding that the input is a noise corrupted subset of the desired output. In contrast our method directly optimizes the predicted map with respect to the input.
3. Our method demonstrates state-of-the-art performance with much fewer parameters compared to deep networks. In fact, using only two layers of dictionaries and 1600 parameters, our method already substantially outperforms modern deep networks which use more than 20 layers and over 3 million parameters [10]. As a result of having fewer parameters, our approach trains faster and requires less data.

2 Related Work

Since our proposed method is a fusion of deep learning and compressed sensing, we will review in this section previous work that has used either technique for depth estimation.

2.1 Compressed Sensing

Compressed sensing is a technique in signal processing for recovering signals from a small set of measurements. Naturally it has been applied to depth completion in previous work, but has been limited to single-level hand-crafted dictionaries. The earliest is Hawe *et al.* [7], who show that disparity maps can be represented sparsely using the wavelet basis. L.-K. Liu *et al.* [8] built on that by combining wavelets with contourlets and investigated the effect of different sampling patterns. Both methods were out performed by Ma & Karaman[9] who exploit the

simple structures of man-made indoor scenes to achieve full depth reconstruction. In contrast to all of these works, our approach learns multi-level convolutional dictionaries from a large dataset of incomplete ground truth depth maps.

2.2 Deep Learning

Depth estimation using deep learning has largely been restricted to single-shot, RGB to depth prediction. This line of inquiry started with Eigen *et al.* [2] who showed that a deep network could reasonably estimate depth using only an RGB image. Many variants of this method have since been explored[5, 6, 1]. Lania *et al.* [3] introduced up-projection blocks which allowed for very deep networks and several other works have proposed variants of their architecture. The most relevant of these variants is the Sparse-to-Dense network of Ma & Karaman[10], which they also apply to depth completion from LiDAR points. Uhrig *et al.* [14] introduced the KITTI depth completion dataset, and showed that CNNs which explicitly encode the sparsity of the input achieve much better performance. Riegler *et al.* [16] designed ATGV-Net, a deep network for depth map super resolution, but they assume a rectangular grid of inputs so it is not applicable to LiDAR completion. We will use the methods of Ma & Karaman and Urhig *et al.* as our baseline comparisons since they represent the state-of-the-art in LiDAR depth completion

Notations. We define our notations throughout as follows: lowercase boldface symbols (*e.g.* $\mathbf{x}$) denote vectors, uppercase boldface symbols (*e.g.* $\mathbf{W}$) denote matrices;

3 Preliminary

3.1 Compressed sensing

Compressed sensing concerns the problem of recovering a signal from a small set of measurements. In our case, we're interested in reconstructing the depth map $\mathbf{d}$ with full resolution from the sparse depth map $\mathbf{d}_s$ produced by LiDAR. To achieve this, certain prior knowledge of the signal is required. The most widely used prior assumption is that the signal can be reconstructed with a sparse linear combination of basis elements from an over-complete dictionary $\mathbf{W}$. This gives an optimization problem similar to sparse coding:

$$\min_{\mathbf{z}} \|\mathbf{MWz} - \mathbf{d}_s\| + b\|\mathbf{z}\|_0 , \tag{1}$$

where $\mathbf{z}$ is the code, $\mathbf{Wz}$ produces our predicted depth map, and $\mathbf{M}$ is a diagonal matrix with 0 and 1s on its diagonal. It's used to mask out the unmeasured portions of the signal, such that the reconstruction error is only applied to the pixels which have been measured.

The key question to apply CS in Eq. 1 is: 1) For high dimensional signals such as the depth map, how to design the dictionary such that it encourages

uniqueness of the code while still being computationally feasible; 2) How to learn the dictionary to get best reconstruction accuracy. In Sec. 4, we are going to show that the dictionary can be factored into a structure equivalent to performing multi-layer convolution, and that we can unroll the optimization of Eq. 1 into a network similar to a deep recurrent neural network. This allows us to learn the dictionary together with other hyper-parameters (*e.g.* b) through end-to-end training.

3.2 Deep Component Analysis

Equation (1) can be generalized to multi-layered sparse coding in which one seeks a very high level sparse representation $\mathbf{z}_\ell$ such that $\mathbf{d} = \mathbf{W}_1\mathbf{W}_2 \dots \mathbf{W}_{\ell-1}\mathbf{z}_\ell$ and each intermediate product $\mathbf{z}_i = \mathbf{W}_i\mathbf{W}_{i+1} \dots \mathbf{W}_{\ell-1}\mathbf{z}_\ell$ is also sparse. This formulation makes using a large effective dictionary computationally tractable, and when the dictionaries have a convolutional structure it allows for increased receptive fields while keeping the number of parameters manageable. This is further generalized to Deep Component Analysis (DeepCA) by the recent work of Murdock *et al.* which replaces the ℓ_0 loss with arbitrary sparsity-encouraging penalties. The DeepCA objective function is stated in [13] as:

$$\min_{\{\mathbf{z}_i\}} \sum_{i=1}^{\ell} \frac{1}{2} \|\mathbf{z}_{i-1} - \mathbf{W}_i\mathbf{z}_i\|_2^2 + \Phi_i(\mathbf{z}_i), \tag{2}$$

where the Φ_j are sparsity encouraging regularizers. Previous work has shown that the specific choice of $\Phi(\mathbf{x}) = I(\mathbf{x} > 0) + b\|\mathbf{x}\|_1$ yields optimization algorithms very similar to a feed-forward neural network with Relu activation functions. By using the ADMM algorithm to solve equation (2), Murdock *et al.* create Alternating Direction Neural Networks, a generalization of feed forward neural networks which internally solve optimization problems with the form of (2). Alternating Direction Neural Networks (ADNNs) perform the optimization in a fully differentiable manner and cast the activation functions of each layer as the proximal operators of penalty function Φ_i of that layer. This allows for learning the dictionaries W_i and parameters b through gradient descent and back propagation with respect to an arbitrary loss function on the sparse codes. To mirror neural networks, Murdock *et al.* apply various loss functions to the highest level of codes, which take the place of the output layer in traditional NNs. In the following sections we will show how ADNNs can be adapted to the depth completion problem within the framework of compressed sensing.

4 Deep Convolutional Compressed Sensing

4.1 Inference

Directly applying compressed sensing to the DeepCA objective gives

$$\min_{\{\mathbf{z}_i\}} \frac{1}{2} \|\mathbf{d}_s - \mathbf{M}\mathbf{W}_1\mathbf{z}_1\|_2^2 + \sum_{i=2}^{\ell} \frac{1}{2} \|\mathbf{z}_{i-1} - \mathbf{W}_i\mathbf{z}_i\|_2^2 + \sum_{i=1}^{\ell} \Phi_i(\mathbf{z}_i), \tag{3}$$

where $\mathbf{d}_s$ is the input sparse depth map. However, if we take the $\mathbf{W}_i$ to have a convolutional structure then an element $\mathbf{z}_1$ will not be recovered if its spatial support contains no valid depth samples. Thus, extracting the higher level codes is itself a missing data problem and can be written the same way. This gives the full Deep Convolutional Compressed Sensing objective:

$$\min_{\{\mathbf{z}_i | i>0\}} \sum_{i=1}^{\ell} \frac{1}{2} \|\mathbf{M}_{i-1}\mathbf{z}_{i-1} - \mathbf{M}_{i-1}\mathbf{W}_i\mathbf{z}_i\|_2^2 + \Phi_i(\mathbf{z}_i). \tag{4}$$

Here, to simplify notation, we merge the depth reconstruction cost (left term in Eq. 3) and the reconstruction cost of the codes together, with $\mathbf{z}_0 = \mathbf{d}_s$ and $\mathbf{M}_0$ denotes the mask $\mathbf{M}$ used in (3) . Each $\mathbf{M}_i$ is a mask encoding which elements of $\mathbf{z}_i$ had any valid inputs in their spatial support. In practice computing $\mathbf{M}_i$ is done with a maxpooling operation with the same stride and kernel size as the convolution represented by $\mathbf{W}_{i+1}^T$.

We solve (4) using the ADMM algorithm, which introduces auxiliary variables $\mathbf{y}_i$ that we constrain to be equal to the codes $\mathbf{z}_i$ as below:

$$\begin{aligned} \min_{\{\mathbf{y}_i, \mathbf{z}_i | i>0\}} & \sum_{i=1}^{\ell} \frac{1}{2} \|\mathbf{M}_{i-1}\mathbf{y}_{i-1} - \mathbf{M}_{i-1}\mathbf{W}_i\mathbf{z}_i\|_2^2 + \Phi_i(\mathbf{y}_i) \\ \text{s.t.} \quad & \mathbf{z}_i = \mathbf{y}_i. \end{aligned} \tag{5}$$

Here, we again refer the input sparse depth $\mathbf{d}_s$ as $\mathbf{y}_0$. With this, the augmented Lagrangian of (5) with dual variables $\boldsymbol{\lambda}$ and a quadratic penalty weight ρ is:

$$L_\rho(\mathbf{z}, \mathbf{y}, \boldsymbol{\lambda}) = \sum_{i=1}^{\ell} \frac{1}{2} \|\mathbf{M}_{i-1}\mathbf{y}_{i-1} - \mathbf{M}_{i-1}\mathbf{W}_i\mathbf{z}_i\|_2^2 + \Phi_i(\mathbf{y}_i) + \boldsymbol{\lambda}_i^T(\mathbf{z}_i - \mathbf{y}_i) + \frac{\rho}{2} \|\mathbf{z}_i - \mathbf{y}_i\|_2^2. \tag{6}$$

The ADMM algorithm then minimizes L_ρ over each variable in turn, while keeping all others fixed. Following Murdock *et al.* we will incrementally update each layer instead of first solving for all $\mathbf{z}_i$ followed by all $\mathbf{y}_i$. They show this order leads to faster convergence. The ADMM updates for each variable are as follows:

1. At each iteration $t+1$, $\mathbf{z}_i$ is first updated by minimizing L_ρ with the associated auxiliary variable $\mathbf{y}_i$ from the previous iteration, and $\mathbf{z}_{i-1}$ from the

current iteration fixed:

$$\begin{aligned}\mathbf{z}_i^{[t+1]} &= \underset{\mathbf{z}_i}{\operatorname{argmin}}\ L_\rho(\mathbf{z}_i, \mathbf{y}_{i-1}^{[t+1]}, \mathbf{y}_i^{[t]}, \boldsymbol{\lambda}_i^{[t]}) \\ &= (\mathbf{W}_i^T\mathbf{M}_{i-1}^T\mathbf{M}_{i-1}\mathbf{W}_i + \rho\mathbf{I})^{-1}(\mathbf{W}_i^T\mathbf{M}_{i-1}^T\mathbf{M}_{i-1}\mathbf{W}\mathbf{y}_{i-1}^{[t+1]} + \rho\mathbf{y}_i^{[t]} - \boldsymbol{\lambda}^{[t]}).\end{aligned} \tag{7}$$

This gives a fully differentiable update of $\mathbf{z}_i$ but the matrix inversion is computationally expensive, especially since W_i is in practice very large. To deal with this problem we make the approximation that $\mathbf{W}_i$ is a Parseval tight frame [13], that is we assume $\mathbf{W}_i\mathbf{W}_i^T = \mathbf{I}$. In addition to being common practice in autoencoders with tied weights, this assumption is also made by Murdock *et al.* and has previously been explicitly enforced in deep neural networks [23]. We can then use the binomial matrix identity to rewrite the $\mathbf{z}_i$ update as:

$$\mathbf{z}^{[t+1]} = \tilde{\mathbf{y}}_i^{[t]} + \frac{1}{1+\rho}\mathbf{W}_i^T\mathbf{M}_{i-1}^T(\mathbf{M}_{i-1}\mathbf{y}_{i-1}^{[t+1]} - \mathbf{M}_{i-1}\mathbf{W}_i\tilde{\mathbf{y}}_i^{[t]}), \tag{8}$$

where $\tilde{\mathbf{y}}_i^{[t]} \triangleq \mathbf{y}_i^{[t]} - \frac{1}{\rho}\cdot$

2. Similarly, the update rule for the auxiliary variables $\mathbf{y}_i$ is:

$$\begin{aligned}\mathbf{y}_i^{[t+1]} &= \underset{\mathbf{y}_i}{\operatorname{argmin}}\ L_\rho(\mathbf{z}_i^{[t+1]}, \mathbf{z}_{i+1}^{[t]}, \mathbf{y}_i, \boldsymbol{\lambda}_i^{[t]}) \\ &= \phi_i\left(\frac{1}{1+\rho}\mathbf{W}_{i+1}\mathbf{z}_{i+1}^{[t]} + \frac{\rho}{1+\rho}(\mathbf{z}_i^{[t+1]} + \frac{\boldsymbol{\lambda}_i^{[t]}}{\rho})\right) \\ \mathbf{y}_\ell &= \phi_i\left(\mathbf{z}_i^{[t]} + \frac{\boldsymbol{\lambda}_i^{[t]}}{\rho}\right).\end{aligned} \tag{9}$$

Here ϕ_i is the proximal operator associated with the penalty function Φ_i. For appropriate choices of Φ_i, ϕ_i is differentiable and can be computed efficiently. With this in mind, we choose $\Phi_i(\mathbf{x}) = I(\mathbf{x} > 0) + b\|\mathbf{x}\|_1$ so that $\phi_i(\mathbf{x}) = \mathrm{ReLU}(\mathbf{x} - \frac{b}{\rho})$.

3. Finally the dual variable $\boldsymbol{\lambda}_i$ is updated by:

$$\boldsymbol{\lambda}_i^{[t+1]} = \boldsymbol{\lambda}_i^{[t]} + \rho(\mathbf{z}_i^{[t+1]} - \mathbf{y}_i^{[t+1]}). \tag{10}$$

The full procedure is detailed in algorithm (1). As shown in above, all the operations used in the ADMM iteration are differentiable, and can be implemented with deep learning layers *e.g.* convolution, convolution transpose, and ReLU. We unroll the ADMM iteration for a constant number of iterations T, and output our optimized code $\mathbf{z}_\ell$ for the last layer. We can then extract our prediction of the depth map by applying the effective dictionary to the high level code z_ℓ as shown in equation (11). This is different from the standard decoder portion of a deep autoencoder, where the nonlinear activations are applied in between each convolution. Our approach does not require this since the internal optimization of z_ℓ enforces equality constraints between layers, which is not the

Algorithm 1: Deep Convolutional Compressed Sensing

Input : model parameters $\mathbf{W}_i, b_i$, iterations T, sparse depth $\mathbf{y}_0 = \mathbf{d}_s$, mask $\mathbf{M}$
Output: sparse codes $\mathbf{z}_i^{[T]}$, predicted depth $\mathbf{d}_{\text{pred}}$
for $i \leftarrow 1$ **to** ℓ **do**
  $\mathbf{z}_i^{[0]} \leftarrow \mathbf{W}_i^T \mathbf{M}_{i-1}^T \mathbf{y}_{i-1}$;
  $\mathbf{y}_i^{[0]} \leftarrow ReLU(\mathbf{z}_i^{[0]} - b/\rho)$;
  $\boldsymbol{\lambda}_i^{[0]} \leftarrow 0$;
for $t \leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** ℓ **do**
    Update $\mathbf{z}_i^{[t]}$ using equation (8);
    Update $\mathbf{y}_i^{[t]}$ using equation (9);
    Update $\boldsymbol{\lambda}_i^{[t]}$ using equation (10);
  Predict $\mathbf{d}_{\text{pred}}$ using equation (11);

case for conventional autoencoders. We choose to reconstruct the depth from z_ℓ instead of a lower layer because its elements have the largest receptive field and therefore z_ℓ will have the fewest number of missing entries.

$$\mathbf{d}_{\text{pred}} = \mathbf{W}_1 \mathbf{W}_2 \dots \mathbf{W}_\ell \mathbf{z}_\ell \tag{11}$$

4.2 Learning

With the ADMM update unrolled to T iterations as described above, the entire inference procedure can be thought of as a single differentiable function:

$$\mathbf{d}_{\text{pred}} = f_{\text{DCCS}}^{[T]}(\mathbf{M}, \mathbf{d}_s; \{\mathbf{W}_i, b_i\}) \tag{12}$$

Thus the dictionaries $\mathbf{W}_i$ and the bias term b_i which are the parameters for $f_{\text{DCCS}}^{[T]}$ can be learned through stochastic gradient descent over a suitable loss function. Using the standard sum of squared loss error, dictionary learning is formed as minimizing the depth reconstruction error $\mathcal{L}_{\text{reconstruct}}$:

$$\min_{\{\mathbf{W}_i, b_i\}} \sum_{n=1}^{N} \left\| \mathbf{d}_{\text{gt}}^{(n)} - \mathbf{M}'^{(n)} \mathbf{d}_{\text{pred}}^{(n)}) \right\| \tag{13}$$

Where $\mathbf{d}_{\text{gt}}^{(n)}$ is the ground truth depth map of the n_{th} training example. We allow the ground truth depth map to have missing value by using mask $\mathbf{M}'^{(n)}$ to segment out the invalid pixels in the ground truth depth map.

In practice we found that due to the sparsity of the training data, the depth maps our method predicted were rather noisy. To fix this issue we included the well known anisotropic total variation loss (TV-L1) when training to encourage smoothness of the predicted depth map. Note that this change has no significant impact on the quantitative error metrics, but produces more visually pleasing

outputs. The total loss is then given by summation of the depth reconstruction loss and the TV-L1 smoothness loss, with hyper-parameter α to control the weighting for the smoothness penalty:

$$\mathcal{L} = \mathcal{L}_{reconstruct} + \alpha \mathcal{L}_{\text{TV-L1}}. \tag{14}$$

We empirically determined that $\alpha = 0.1$ produces the best results.

5 Experiments

5.1 Implementation Details

We implemented three variants of algorithm (1) for the cases $\ell = 1, 2, 3$. For the single layer case we let $\mathbf{W}_1^T$ be a 11x11 convolution with striding of 2 and 8 filters. For $\ell = 2$ we let $\mathbf{W}_1^T$ be an 11x11 convolution with 8 filters and W_1^T be a 7x7 convolution with 16 filters. Finally for the $\ell = 3$ case: $\mathbf{W}_1^T$ is an 11x11 convolution with 8 filters, $\mathbf{W}_2^T$ is a 5x5 convolution with 16 filters, and $\mathbf{W}_3^T$ is a 3x3 convolution with 32 filters. For both $\ell = 2$ and $\ell = 3$, all convolutions have striding of 2. For the single layer case we learned the dictionaries with the number of iterations set to 5 and then at test time increased the number of iterations to 20. For the two and three layer cases the number of iterations was fixed at train and test time to 10 except in section 5.4 where the number of test and training iterations is varied. All training was done with the ADAM optimizer with the standard parameters: learning rate = 0.001,$\beta_1 = 0.9$, β_2 =0.999, $\epsilon = 10^{-8}$.

Error Metrics For evaluation on the KITTI benchmark we use the conventional error metrics [14, 4], *e.g.* root mean square error (RMSE), mean absolute error (MAE), mean absolute relative error (MRE). We also use the percentage of inliers metric, δ_i which counts the percent of predictions whose relative error is within a threshold raised to the power i. Here, we use smaller thresholds (1.01^i) compared to the more widely used ones (1.5^i) in oder to compare differences in performance under tighter metrics.

5.2 KITTI Depth Completion Benchmark

We evaluate our method on the new KITTI Depth Completion Benchmark [14] instead of directly comparing against the LiDAR measurements from the raw KITTI dataset. The raw LiDAR points given in KITTI are corrupted by noise, motion of the vehicle during sampling, image rectification artifacts, and accounts to only 4% of the total number of pixels in the image. Thus it's not ideal for evaluating depth completion systems. Instead, the benchmark proposed in [14] resolved these issues by accumulating LiDAR measurements from nearby frames in the video sequences, and automatically removing accumulated LiDAR points that deviate too far from the points reconstructed by semi-global matching. This provides quality ground truth and effectively simulates the main application of interest: recovering dense depth from a single LiDAR sweep.

	RMSE (m)	MAE (m)	MRE	$\delta_1 < 1.01$	$\delta_2 < 1.01^2$	$\delta_3 < 1.01^3$
Bilateral NN[17]	4.19	1.09	-	-	-	-
SGDU[18]	2.5	0.72	-	-	-	-
Fast Bilateral Solver[19]	1.98	0.65	-	-	-	-
TGVL[20]	4.85	0.59	-	-	-	-
Closest Depth Pooling	2.77	0.94	-	-	-	-
Nadaraya Watson[21, 22]	2.99	0.74	-	-	-	-
ConvNet	2.97	0.78	-	-	-	-
ConvNet + mask	2.24	0.79	-	-	-	-
SparseConvNet[14]	1.82	0.58	0.035	0.33	0.65	0.82
Ma & Karaman[10]	1.68	0.70	0.039	0.21	0.41	0.59
Ours 1 Layer	2.77	0.83	0.054	0.3	0.47	0.59
Ours 2 Layers	1.45	0.47	0.028	0.41	0.68	0.8
Ours 3 Layers	**1.35**	**0.43**	**0.024**	**0.48**	**0.73**	**0.83**

Table 1: Validation error of various methods on the KITTI Depth Completion benchmark. All results except for SparseConvNet and Ma's are taken as reported from [14]. Our method outperforms all previous state-of-the-art depth only completion methods (Middle) as well as those that use RGB images for guidance (Top).

In Table 1, we form a close comparison against the very deep Sparse-to-Dense network (Ma & Karaman [10]) and the Sparsity Invariant CNN (SparseConvNet [14]) which are the current sate-of-the-art deep learning-based method.

The Sparse-to-Dense network uses a similar deep network architecture as those used for single shot depth prediction – with Resnet-18 as the encoder and up-projection blocks for the decoder. While the Sparse-to-Dense network is able to achieve good RMSE, it falls behind the SparseConvNet on MAE. We believe that this is because the deeper network can better estimate the average depth of a region but is unable to predict fine detail, leading to a higher MAE. By comparison, our method is able to both estimate the correct average depth and reconstruct fine detail due to its ability to directly optimize the prediction with respect to the input. Most notably our method outperforms all of the existing methods by a wide margin, including those that use RGB images and those that use orders of magnitude more parameters than our method.

Varying Sparsity Levels Uhrig *et al.* show that their Sparsity Invariant CNNs are very robust to a mismatch between the training and testing levels of sparsity. While we do not see a practical use for disparities as large as those tested in [14], we do believe that depth completion systems should perform well under reasonable sparsity changes. To this end we adjusted the level of input sparsity in the KITTI benchmark by dropping input samples with probability p, for various values of p. The results of this experiment are shown in Figure (2). While it is clear that our method does not achieve the level of sparsity invariance of the SparseConvNet, it still outperforms both baseline results even when the only 50% of the input samples are kept.

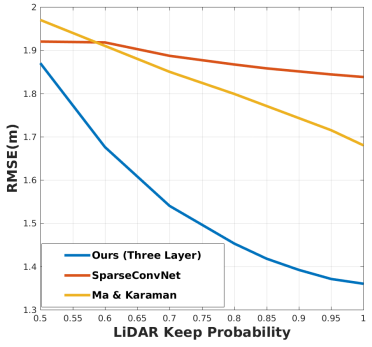

Fig. 2: Results on the KITTI benchmark for varying levels of input sparsity. The keep probability represents the probability that any particular LiDAR sample is retained. We demonstrate robustness to reasonable changes in input sparsity, outperforming both baselines up to a 50% reduction in the number of input points.

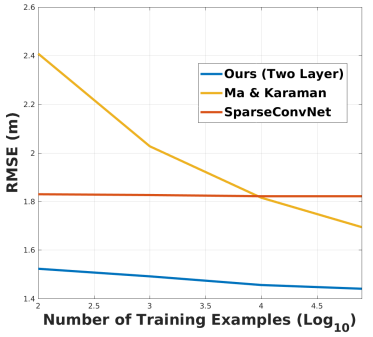

Fig. 3: Results of selected methods on the KITTI benchmarks for varying training set sizes. Our method performs well with training sizes ranging from 100-86k but still benefits from larger training sizes.

5.3 Effect of Amount of Training Data

Modern deep learning models typically have tens of thousands to millions of parameters and therefore require enormous training sets to achieve good performance. This is in fact the motivation for the KITTI depth completion dataset, since previous benchmarks did not have enough data to train deep networks. In this section we investigate the dependence on the amount of training data on the performance of our method in comparison with a standard deep network and the sparsity invariant variety.

Figure (3) shows the results of evaluating these models on the 1k manually selected validation depth maps after training on varying subsets of the 86k training maps. Our method outperforms both baselines for all training sizes. As expected Ma & Karaman's method fails to generalize well when trained on a small dataset since the model has 3.4M parameters but performs well once trained on the full dataset. It is interesting to observe that the method of Uhrig *et al.* does not gain any performance from training on more data. As a result it is ultimately out performed by the deep network which does not take sparsity into account. Our method is able to perform comparably to the sparsity invariant network with only 100 training examples but does increase in performance when given more data, validating the need for learning layered sparse coding dictionaries from large training sets.

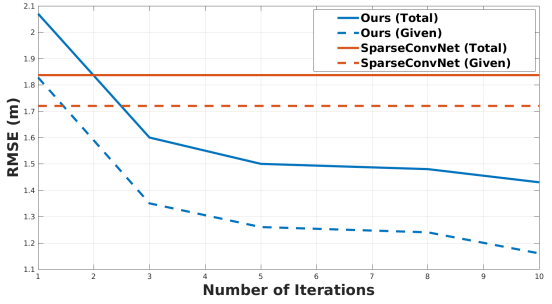

Fig. 4: Results on the depth completion benchmark for different numbers of ADMM iterations. The total error is shown in blue while the red line shows the error on just those points given as input. The dotted lines show the same metrics but for the SparseConvNet of Uhrig *et al.* [14].

5.4 Effect of Iterative Optimization

In this section we demonstrate that the success of our approach comes from its ability to refine depth estimates over multiple iterations. Applying a feed forward neural network to this problem frames it as finding a mapping from sparse LiDAR points to true depth maps. This is a reasonable approach but it doesn't utilize all of the available information, specifically it doesn't encode the relationship that input samples are a subset of the output that has been corrupted by noise. In contrast, our approach of phrasing depth completion as a compressed sensing missing data problem directly expresses that relationship. By solving this problem in an iterative fashion our network that is able to find depth maps that are both consistent with the input constraints and have sparse representations.

The importance of iterative optimization is shown in Figure (4) where we examine the performance of our method as a function of the number of ADMM iterations it uses. It is clear that with few iterations our network fails to enforce the constraints and performs comparably to the SparseConvNet. This is also consistent with Murdock *et al.* 's observation that a feed forward network resembles a single iteration of an ADNN. As we increase the number of iterations our method is able to better optimize its prediction and gains a substantial performance boost.

6 Conclusion

In this work we have proposed a novel deep recurrent autoencoder for depth completion. Our architecture builds on the work of Murdock *et al.* on Deep Component Analysis and further establishes the link between sparse coding and deep learning. We demonstrate that our model outperforms existing methods

for depth completion, including those that leverage RGB information. We also show that the success of our method is fundamentally a product of the internal optimization it performs, and that due to its small number of parameters it is able to perform well even without a large training set.

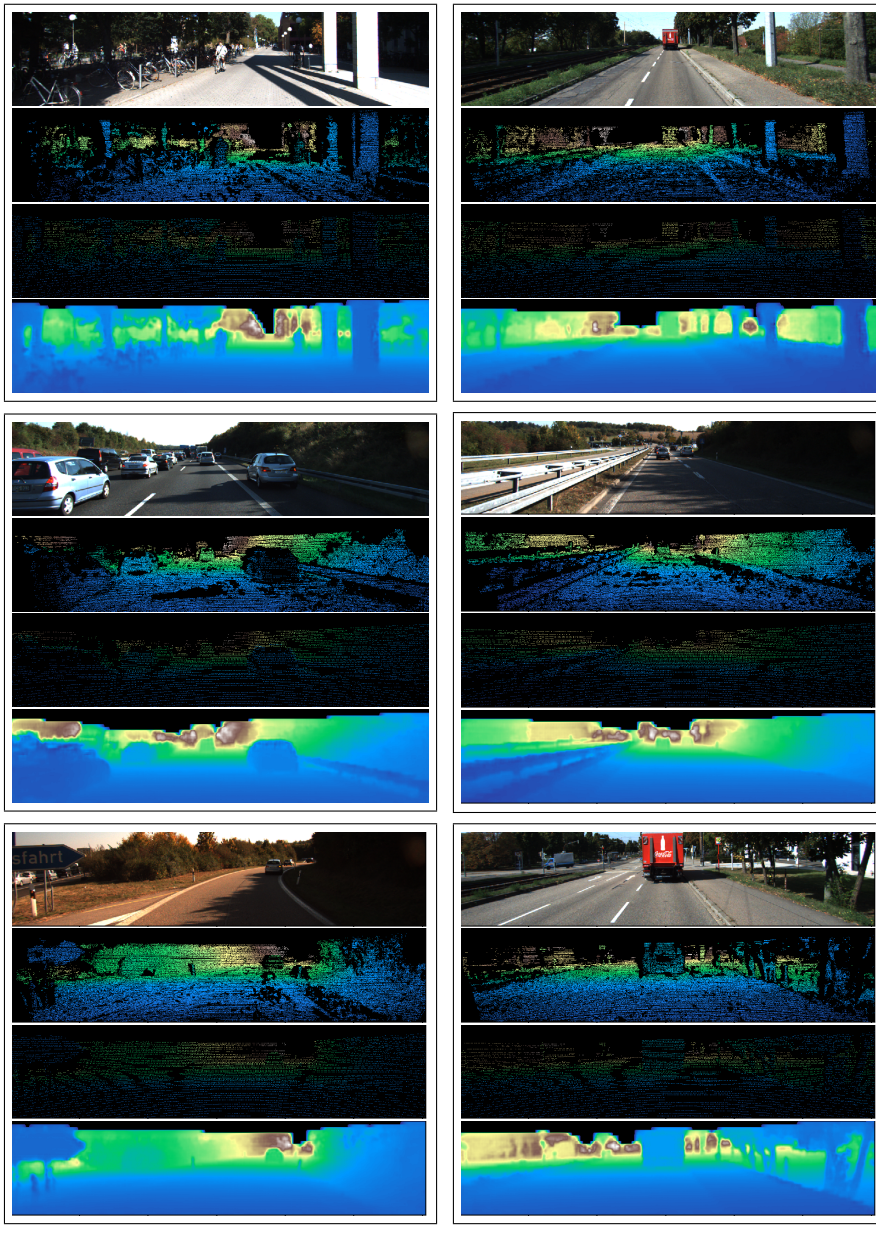

Fig. 5: Selected visual results form the KITTI benchmark. From top to bottom: RGB Image, Ground truth, input LiDAR points, Predicted depth.

References

1. F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170.
2. D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.
3. I. Laina, C. Rupprecht *et al.*, "Deeper depth prediction with fully convolutional residual networks," in *3D Vision (3DV), 2016 Fourth International Conference on.* IEEE, 2016, pp. 239–248.
4. A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
5. Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," *arXiv preprint arXiv:1702.02706*, 2017.
6. C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," *arXiv preprint arXiv:1609.03677*, 2016.
7. S. Hawe, M. Kleinsteuber, and K. Diepold, "Dense disparity maps from sparse disparity measurements," in *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2126–2133.
8. L.-K. Liu, S. H. Chan, and T. Q. Nguyen, "Depth reconstruction from sparse samples: Representation, algorithm, and sampling," *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1983–1996, 2015.
9. F. Ma, L. Carlone *et al.*, "Sparse sensing for resource-constrained depth reconstruction," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on.* IEEE, 2016, pp. 96–103.
10. F.Ma, S. Karaman, "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image", *arXiv preprint arXiv:1709.07492*, 2017
11. M. Elad. *Sparse and Redundant Representations.* New York, NY: Springer, 2010
12. Vardan, Papyan & Romano, Yaniv & Elad, Michael. "Convolutional Neural Networks Analyzed via Convolutional Sparse Coding". *Journal of Machine Learning Research.* 18. 2016
13. C. Murdock. "Deep Component Analysis via Alternating Direction Neural Networks", *arXiv preprint*, 2018
14. J. Uhrig, N. Schneider, L. Schneider, U. Franke, A. Geiger. "Sparsity Invariant CNNs", *arXiv preprint arXiv:1708.06500*, 2017
15. J. Sulam, V. Papyan, Y. Romano, M. Elad. "Multi-Layer Convolutional Sparse Modeling: Pursuit and Dictionary Learning". *arXiv preprint arXiv:1708.08705*
16. G. Riegler, M. Ruther, H. Bischof. "ATGV-Net: Accurate Depth Super-Resolution.". in *European Conference on Computer Vision*, 2016
17. V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
18. N. Schneider, L. Schneider, P. Pinggera, U. Franke, M. Pollefeys, and C. Stiller. Semantically guided depth upsampling. In *Proc. of the German Conference on Pattern Recognition (GCPR)*, pages 37–48. Springer, 2016.
19. J. T. Barron and B. Poole. The fast bilateral solver. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 617–632. Springer, 2016.

20. D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof. Image Guided Depth Upsampling Using Anisotropic Total Generalized Variation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2013.
21. E. Nadaraya. On estimating regression. In *Theory of Probability and its Applications*, 1964.
22. G. Watson. Smooth regression analysis. In *Sankhy: The Indian Journal of Statistics*, 1964.
23. C. Moustapha, B. Piotr, G. Edouard, D. Yann, and U. Nicolas. "Parseval networks: Imporoving robustness to adversarial examples". *arXiv preprint arXiv:1704.08847*