Abstract

Sparse active illumination enables precise time-of-flight depth sensing as it maximizes signal-to-noise ratio for low power budgets. However, depth completion is required to produce dense depth maps for 3D perception. We address this task with realistic illumination and sensor resolution constraints by simulating ToF datasets for indoor 3D perception with challenging sparsity levels. We propose a quantized convolutional encoder-decoder network for this task. Our model achieves optimal depth map quality by means of input pre-processing and carefully tuned training with a geometry-preserving loss function. We also achieve low memory footprint for weights and activations by means of mixed precision quantization at training techniques. The resulting quantized models are comparable to the state of the art in terms of quality, but they require very low GPU times and achieve up to 14-fold memory size reduction for the weights w.r.t. their floating point counterpart with minimal impact on quality metrics.

1. Introduction

Time-of-flight (ToF) sensors are active depth sensing devices [53] with the potential to provide more reliable scene understanding by true 3D perception. Due to their low power consumption and accuracy at real-time frame rates, ToF sensors were recently integrated in mobile consumer devices [33]. However, ToF relies on active illumination [53] which accounts for a relevant part of its power consumption. To use the limited power budget of a mobile device more efficiently, the scene can be illuminated with a dot pattern light source so that its radiant intensity concentrates onto a small number of regions (dots). A low-power ToF sensor for indoors 3D perception typically captures 500 ~ 1500 dots per frame. Because of this sparsity level, sensor fusion techniques are necessary to obtain dense depth maps. Hereafter, we tackle this depth completion task by using a single RGB image frame together with a sparse depth map warped to the RGB camera view (at small baseline of a few mm to minimize occlusion issues w.r.t. the ToF sensor). We dub this the sparse ToF setting.

To obtain dense depth from a sparse depth map, deep learning-based approaches have gained significant attention, with several contributions showing competitive performances especially on autonomous driving datasets [17, 44, 47]. We instead focus on efficient indoor 3D perception with ToF. In order to attain overall low power budget for both ToF sensing and depth completion, neural networks must be designed and trained accordingly for efficient inference. In this paper we introduce a carefully designed encoder-decoder CNN for depth completion. We achieve a low memory footprint for weights and activations by means of mixed-precision quantization-aware training and show that the resulting quantized models achieve conspicuous memory size savings for weights and activations, while remaining competitive w.r.t. quality metrics. Our contributions are as follows:

- We propose an efficient depth completion model suitable for low-power sparse ToF sensing. We evaluate our model and other state-of-the-art models on two datasets: the public-domain dataset NYU-Depth v2 [43], and SDS-ST1k [6], a contributed dataset providing accurate sparse ToF and RGB camera simulations.
- We analyze the impact of input pre-processing providing fast initialization; a normals estimation block enabling supervision with ground truth normals; different loss functions and network configurations. Each of these elements contributes to achieving optimal depth map quality for the resulting model.
- We evaluate quantization-aware training strategies to quantize weights and activations of the previous model. To the best of our knowledge, mixed-precision quantization [46] has been used successfully to classification tasks. In this paper we present the first ap-
application of mixed precision quantization to the depth completion task. This technique allows us to minimize the weights and activations memory requirements with very limited quality degradation.

2. Related Works

2.1. Depth Completion

The main task we explore in this paper is popular in computer vision since the introduction of depth sensors [3, 11] and has been addressed by neural networks with convolutional encoder-decoder architectures [9, 24, 34, 35, 37, 39, 52]. Many other works define the state of the art in depth completion [12, 13, 20, 22, 28, 31, 40, 45]. We will compare our results against CNN solutions [8, 9, 34, 37] selected on the basis of their similar complexity (single encoder-decoder). These will be retrained on our sparse ToF datasets. There exist also higher-complexity networks leveraging, e.g., multiple decoder branches [39, 51] or separate encoder branches [16, 42, 48, 52] but in this work we aimed for a more compact and lightweight model. We will only focus on single-frame depth completion. Many contributions extend depth completion by either self-supervision or by providing as input multiple RGB-D frames with known or inferred poses [16, 34, 38, 49, 50]. By tackling only the single-frame case, we grant temporally-independent behavior for both dynamic and static cameras and remove additional computational effort required by pose estimation via neural networks (e.g., [25]) or traditional odometry. We now discuss the main works we compare against.

Sparse-to-Dense (S2D [34]) is a popular approach using a single encoder-decoder network. Its core contribution is self-supervision by pose estimation (a feature we do not leverage) as well as its long-standing role in the KITTI application of mixed precision quantization to the depth completion task. This technique allows us to minimize the weights and activations memory requirements with very limited quality degradation.

2. Related Works

2.1. Depth Completion

The main task we explore in this paper is popular in computer vision since the introduction of depth sensors [3, 11] and has been addressed by neural networks with convolutional encoder-decoder architectures [9, 24, 34, 35, 37, 39, 52]. Many other works define the state of the art in depth completion [12, 13, 20, 22, 28, 31, 40, 45]. We will compare our results against CNN solutions [8, 9, 34, 37] selected on the basis of their similar complexity (single encoder-decoder). These will be retrained on our sparse ToF datasets. There exist also higher-complexity networks leveraging, e.g., multiple decoder branches [39, 51] or separate encoder branches [16, 42, 48, 52] but in this work we aimed for a more compact and lightweight model. We will only focus on single-frame depth completion. Many contributions extend depth completion by either self-supervision or by providing as input multiple RGB-D frames with known or inferred poses [16, 34, 38, 49, 50]. By tackling only the single-frame case, we grant temporally-independent behavior for both dynamic and static cameras and remove additional computational effort required by pose estimation via neural networks (e.g., [25]) or traditional odometry. We now discuss the main works we compare against.

Sparse-to-Dense (S2D [34]) is a popular approach using a single encoder-decoder network. Its core contribution is self-supervision by pose estimation (a feature we do not leverage) as well as its long-standing role in the KITTI leaderboard compared to its low complexity. The network architecture is similar to ours. As shown in [34, Sec. 6.4] and confirmed in our experiments, its robustness to sparsity is limited. We reproduced and retrained their model.

Convolutional Spatial Propagation Network (CSPN [9]) was among the first works [9, 37, 51] to propose a two-stage network architecture using spatial propagation layers. This entails: (i) generation of a “blurred” depth map and affinity matrix; (ii) iterative refinement using the learned affinity matrix as weights for anisotropic diffusion. We retrained the authors’ model via their code.

2.2. Neural Networks Quantization

Quantized Neural Networks (QNN) [18, 29, 54] are DNNs for which a smaller number of bits $b \leq 32$ is used to represent the weights and activations. QNNs are key to achieving efficient and low-power inference: they require considerably less memory and have a lower computational complexity, since quantized values can be stored, multiplied and accumulated efficiently.

Quantization-Aware Training (QAT) refers to a set of special algorithms to train QNNs, leading to superior results than post-training quantization. Recent QAT methods [2, 7, 10, 23, 30] proved effective on classification tasks. These methods often use uniform quantization and the same bit width in each network layer. To improve QAT, Uhlich et al. [46] and Nikolić et al. [36] proposed to learn the optimal bit width for each layer while achieving a target size budget. On classification tasks, these mixed precision approaches show superior results since the bit widths can be allocated optimally across the network. Much less evidence of successful QAT exists for regression tasks such as depth completion.

3. Sparse ToF Datasets

We target datasets that are critically challenging in terms of sparsity. We consider indoor scenes (range: $[0, 15]m$) with rich structure and detailed objects with diverse cam-
era orientations. Given sparse depth maps $D_S$, let us define the sparsity level $K := 100 \frac{|\{i \in [n_h] \times [n_v] : (D_S)_i \neq \text{Invalid}\}|}{n_h \times n_v}$ at target depth map resolution $n_h \times n_v$. We tackle very sparse settings where $K \approx 0.4 \sim 1.4\%$ (as a reference in the widely used KITTI depth completion benchmark $K \approx 4 \sim 5\%$). A visual overview is provided in Fig. 1.

Indeed, such higher values of $K$ simplify the depth completion task. In this work we will focus on indoor 3D perception datasets with very sparse depth maps, where traditional pipelines [27] will struggle to achieve good results.

Our main results are given on NYU-Depth v2 [43] as broadly adopted public-domain dataset. For the training set, we use a subset of $\approx 50K$ RGB-D images from the 249 standard training scenes. Each image is downsized to $320 \times 240$ then center-cropped to $304 \times 224$, identically to [35, 37]. The main difference we introduce is that we process the depth by subsampling it with a triangular tiling dot pattern generated from sparse illumination of commercial VCSELs [33] instead of choosing random indices. This results in sparse depth maps $D_S$ with on-average 943 active pixels ($K \approx 1.4\%$; see Fig. 1b) sampled from the full, dense depth map $D_{GT}$ which is used as supervision. The surface normals map $N_{GT}$ was also estimated from $D_{GT}$. The standard test set of 654 images is processed identically.

In addition, we provide results on SDS-ST1k [6], a contributed dataset using 8 diverse environments from Unreal Engine [14] marketplace assets yielding $\approx 18K$ images from random camera poses at resolution 640 $\times$ 480. To obtain sparse ToF data, we apply a light source with dot pattern light shading in our raytracing ToF simulator; this is received on a simulated ToF sensor compatible with low power designs and camera specifications (e.g., sensor integration time, camera intrinsics). This provides both sparse ToF depth maps $D_s$ with realistic parallax and pattern geometry, and ground truth $D_{GT}$, $N_{GT}$. In this setting we obtain on-average 1239 active pixels, yielding a challenging $K \approx 0.4\%$ (see Fig. 1c). The recommended training/testing split yields $\approx 5K$ images as test set; this will be provided at https://github.com/sony/ai-research-code. We shall train our models with $160 \times 160$ non-overlapping patches (i.e., 12 per image) to meet training device memory limitations.

4. Network Design

The input of all considered depth completion methods is an RGB-D frame comprised of a color image and a sparse depth map, suitably projected to RGB camera space by assuming known intrinsics and RGB-ToF extrinsics, and with negligible RGB-ToF baseline to avoid occlusions.

4.1. Input Pre-Processing

We leverage the input pre-processing method of Chen et al. [8]: given the sparse depth $D_S$, we compute an initial guess $D_{NNI}$ of the depth map by Nearest Neighbors Interpolation (NNI), i.e., mapping each coordinate to the closest
sparse depth pixel in the Euclidean sense. We also compute the Euclidean Distance Transform (EDT) $E$ with [15] (via OpenCV) given the 2-D valid coordinates of $D_S$. NNI is obtained by the same call, which takes median CPU time $t_{NNI} = 4.99$ ms over SDS-ST1k and $t_{NNI} = 0.99$ ms over NYU-Depth v2. We normalize $D_{NNI}$ by the maximum range $D_{max} := 15m$ and $E$ by $E_{max} := 40$, which we find empirically to be an appropriate normalization value for our datasets. The EDT feeds the CNN an uncertainty map of $D_{NNI}$, i.e., it is low where sparse depth is known and progressively higher when it is not. We tried replacing NNI with linear interpolation\(^3\) in preliminary tests, but this did not show any benefits while increasing CPU time.

### 4.2. Network Architecture

A summary of our supervised approach is given in Fig. 2. The core CNN element (Fig. 2b) is a simple “UNet-like” [41] template, dubbed UNet\(_{θ,n_f,n_s}\), of which we tune the number of scales $n_s$ and feature maps $n_f$ relative to the highest scale. Our reference is $n_s = 5$, $n_f = 64$. The number of feature maps doubles at each scale as their spatial resolution is reduced, up to $2^{n_s-1}$ at the lowest scale. The basic building blocks (Fig. 2c) are standard 2-D convolutions with $3 \times 3$ filters, with padding to same resolution as the layer input and stride 1. We chose to replace transpose convolutions in the decoder with upsampling and convolution layers, as preliminary tests indicated better performance on our datasets. All skip connections are by concatenation except for the last one, which is a tensor addition w.r.t. $D_{NNI}$. Thus, we have the (normal) interpolated depth $\hat{D} := D_{NNI} + \text{UNet}_{θ,n_f,n_s}(D_{NNI}, E, C)$, i.e., UNet\(_{θ,n_f,n_s}\) computes a residual w.r.t. the initial guess.

### 4.3. Normals Estimation Block

As shown in Fig. 2a, we estimate the surface normals map $\hat{N}$ from the interpolated depth map $\hat{D}$ rather than using costly dedicated decoders [39]. We resort to a well-known approximation that is directly applicable to depth maps; the computation graph is reported in Fig. 2d. Given a depth map tensor $D$ at the input, we approximate horizontal and vertical image axes derivatives by centered differences using two fixed-kernel 1-D convolutions yielding the tensors $(\hat{D}/\hat{x}, \hat{D}/\hat{y})$. We then have at the $i$-th pixel the normal vector

$$ (\hat{N})_i := \left[\frac{\hat{D}/\hat{x}}{\sqrt{(\hat{D}/\hat{x})^2 + (\hat{D}/\hat{y})^2}}\right]_i - 1. \quad (1) $$

We found this small-kernel method sufficiently accurate to enforce similarity\(^4\) between $(\hat{N})_i$, $(N_{GT})_i$ at training.

### 4.4. Loss Function

Our supervised training follows the flow in Fig. 2a. The supervision is comprised of two terms. The first term measures the $ℓ_p$-norm distance between (normalized) ground truth $D_{GT}$ and $\hat{D}$, i.e., for dataset $D$ with $J$ samples,

$$ L_{\ell_p}(D) := \frac{1}{J} \sum_{j=1}^{J} \|\text{vec}(D_j - \hat{D}_{GT,j})\|_p, \quad p = 1, 2. \quad (2) $$

The second term measures similarity between the dense ground truth normals map $N_{GT}$ and $\hat{N}$ from the block in Fig. 2d: for each $i$-th pixel normal we measure the cosine similarity $\cos \alpha = \langle (N_{GT})_i, (\hat{N})_i \rangle$. We then take its average over the dataset and define the normals loss

$$ L_{\ell}(D) := -\frac{1}{J_{N} \cdot n_v} \sum_{j=1}^{J} \sum_{i=1}^{n_h n_v} \langle (N_{GT,j})_i, (\hat{N})_i \rangle \quad (3) $$

with range $[-1, 1]$ (at $-1$ all normals are identical). In the presence of invalid ground truth depth or normals map (e.g., for invalid pixels in NYU-Depth v2) we exclude the corresponding pixel from the loss computations.

Only part of the literature uses normals supervision\(^5\) [39, 51], while all depth completion approaches use depth supervision providing scale. Normals supervision alone is not

---

\(^3\)Linear interpolation on a non-uniform grid due to the dot pattern requires Delaunay triangulation, which is costly and cannot be reused as the pattern can change at every frame.

\(^4\)We ensure the surface normals’ orientation is always consistently pointing towards the camera both in $N_{GT}$ and $\hat{N}$.

\(^5\)Smoothness losses on the gradients of $D$ are also common [34, 42] but not considered here as the normals loss implements similar behavior in a supervised and more geometrically sound fashion.

<table>
<thead>
<tr>
<th>$L$</th>
<th>$\lambda_n$</th>
<th>RMSE (mm)</th>
<th>MNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{\ell_2}$</td>
<td>n/a</td>
<td>105.2</td>
<td>0.538</td>
</tr>
<tr>
<td>$L_{\ell_1}$</td>
<td>n/a</td>
<td>97.8</td>
<td>0.754</td>
</tr>
<tr>
<td>$L_{\ell_1} + \lambda_n L_n$</td>
<td>1</td>
<td>259.0</td>
<td>0.864</td>
</tr>
<tr>
<td>$L_{\ell_1} + \lambda_n L_n$</td>
<td>$10^{-1}$</td>
<td>227.0</td>
<td>0.868</td>
</tr>
<tr>
<td>$L_{\ell_1} + \lambda_n L_n$</td>
<td>$10^{-2}$</td>
<td>101.5</td>
<td>0.838</td>
</tr>
<tr>
<td>$L_{\ell_1} + \lambda_n L_n$</td>
<td>$10^{-3}$</td>
<td>98.7</td>
<td>0.783</td>
</tr>
<tr>
<td>$L_{\ell_1} + \lambda_n L_n$</td>
<td>$10^{-4}$</td>
<td>109.3</td>
<td>0.764</td>
</tr>
</tbody>
</table>

(a)

![Color](image1.png)

![$L_{\ell_2}$](image2.png)

![$L_{\ell_1}$ + $\lambda_n L_n$](image3.png)

(b)

Figure 3. Loss function tuning on SDS-ST1k (resolution $640 \times 480$): (a) Quantitative analysis; (b) Qualitative analysis of error maps (range: $[-500, 500]$ mm). Figure best viewed in color.
Table 1. Network tuning on SDS-ST1k (resolution 640 × 480). We confirm a steady decrease of quality metrics as the feature maps \( n_f \) relative to the highest level and the scales \( n_s \) are reduced.

\[
\begin{array}{|c|c|c|c|c|}
\hline
n_f & n_s & \text{MPrams} & \text{RMSE (mm)} & \text{MNS} \\
\hline
64 & 5 & 50.3 & 98.7 & 0.783 \\
32 & 5 & 12.6 & 105.0 & 0.775 \\
16 & 5 & 3.2 & 181.1 & 0.751 \\
64 & 4 & 12.6 & 219.2 & 0.779 \\
64 & 4 & 3.1 & 248.4 & 0.761 \\
32 & 4 & 3.1 & 274.8 & 0.767 \\
\hline
\end{array}
\]

sufficient to retrieve scale. Thus, we minimize

\[
\mathcal{L}(D) := \mathcal{L}_{\ell_1}(D) + \lambda_n \mathcal{L}_n(D)
\]

which balances between scale-dependent and scale-independent terms using \( \lambda_n \in [0, 1] \). This weight depends on the dataset and must be tuned empirically. To do so, we run preliminary tests of our approach (Fig. 2a) on SDS-ST1k under different\(^6\) \( \mathcal{L} := \{ \mathcal{L}_{\ell_2}, \mathcal{L}_{\ell_1}, \mathcal{L}_{\ell_1} + \lambda_n \mathcal{L}_n \} \) and \( \lambda_n := 10^{-3}, q = 0, \ldots, 4 \), in the same settings described in Sec. 6.1. Indeed, defining the conventional RMSE and Mean Normals Similarity

\[
\text{MNS} := \frac{1}{b \times h \times w} \sum_{j=1}^{J} \sum_{i=1}^{n_h \times n_v} \langle |(N_{GT,j})_i|, |(N_j)_i| \rangle
\]

as quality metrics we see that large values of \( \lambda_n \) degrade the RMSE, and \textit{vice versa} for MNS. We find the best tradeoff at \( \lambda_n^{opt} = 10^{-3} \). As confirmation of this, in Fig. 3b we report one sample error map \( \hat{D} - D_{GT} \) under different \( \mathcal{L} \). As we move from \( \mathcal{L}_{\ell_2} \) to \( \mathcal{L}_{\ell_1} \), the observed error map pattern artefact disappears. Remaining errors at depth map discontinuities are reduced by using the normals loss \( \mathcal{L}_n \).

4.5. Network Tuning

We investigated simple strategies to reduce the size of the initial model, such as tuning the \( UNet_{n_f, n_s} \) module by its number of feature maps \( n_f \) relative to the highest scale, and the number of scales \( n_s \). Our preliminary tests, in the same setting as Sec. 6.1 on SDS-ST1k, actually confirm (Tab. 1) that any pair smaller than \((n_f, n_s) = (64, 5)\) leads to a rapid degradation of RMSE and MNS, with the reduction of \( n_s \) (shallower network) being generally worse than that of \( n_f \). Thus, we choose to achieve memory size reduction by quantization of the full model with \((n_f, n_s) = (64, 5)\) which, as we will show, allows for a more compact model with limited impact on quality metrics.

5. Quantization-Aware Training

Let \( Q(x; \phi) \) be an arbitrary scalar quantizer with parameters \( \phi \), which maps \( x \in \mathbb{R} \) to discrete values \( \{q_1, ..., q_t\} \) represented by \( b \) bits, \( b : I \leq 2^b \). This quantizer may be applied to both weights and activations to reduce their memory size requirements. This can be done by either post-training quantization or by Quantization-Aware Training (QAT), which generally achieves better accuracy than the former. QAT consists in training the quantized version of a DNN (in this case, CNN) while applying \( Q(x; \phi) \) to its weights and/or activations. However, this is challenging as the gradient through the quantizer is zero almost everywhere; a solution to this is the Straight-Through Estimator (STE)\(^4\). A symmetric uniform quantizer \( Q_U(x; \phi) \) then maps \( x \in \mathbb{R} \) to uniformly quantized values by

\[
q = Q_U(x; \phi) := \text{sign}(x) \begin{cases} \frac{x}{q_{\text{max}}} + \frac{1}{2} & |x| \leq q_{\text{max}} \\ q_{\text{max}} & |x| > q_{\text{max}} \end{cases}
\]

where the parameter vector \( \phi := [d, q_{\text{max}}, b]^T \) with step size \( d \in \mathbb{R}^+ \), maximum value \( q_{\text{max}} \in \mathbb{R}^+ \), and number of bits or \textit{bit width} \( b \in \mathbb{N} : b \geq 2 \) of the quantized value \( q \).

5.1. Uniform Precision Models

We start by considering \textit{uniform precision} (UP) QAT, \textit{i.e.}, every weight (or activation) of every layer has the same predefined bit width \( b \) and QAT optimizes over the remaining parameters \( d, q_{\text{max}} \) of \( \phi \). We conducted extensive experiments to preserve depth completion accuracy while quantizing with (6) the weights and activations in our model. The most effective UP QAT training procedure was obtained by initializing from the pretrained \texttt{float32} model and using learnable \( q_{\text{max}} \) as in the Trained Quantization Threshold (TQT) approach\(^{23}\); to stabilize the QAT, we used cosine learning rate decay scheduling\(^{32}\), and we first trained with RMSprop until convergence (32 epochs in our case), followed by Adam\(^{26}\) for 20 more epochs.

5.2. Mixed Precision Models

Recent results\(^{36, 46}\) show that, for a given memory size budget, better accuracy than UP QAT can be achieved using \textit{mixed precision} (MP) QAT, \textit{i.e.}, each layer uses its own bit width \textit{learned at training} to fit a target total network size. Following\(^{46}\) we parametrize \( Q_U(x; \phi) \) with their range and step size \( d \), from which the bit width \( b := [1 + \log_2(q_{\text{max}}/d + 1)] \) is then inferred. With this parametrization and STE\(^4\) gradient, we can use standard stochastic gradient descent methods to learn per-layer optimal bit widths jointly with network parameters. To achieve target network sizes we add to (4) \textit{size constraints} on the weights and activations as penalty terms\(^{5}\), minimizing

\[
\mathcal{L}_{MP} = \mathcal{L}(D) + \lambda_w \max(0, S_w)^2 + \lambda_A \max(0, S_A)^2
\]

\[
S_x = \left( \sum_{i=1}^{L} S_{x}^i \right) - S_{x}^0 ; S_{x}^i = N_x b_{x,i} S_{x} = b_x \sum_{i=1}^{L} N_x b_{x,i}
\]
Weights Size (MB)

<table>
<thead>
<tr>
<th>Precision</th>
<th>W4</th>
<th>W3</th>
<th>W2</th>
<th>W1</th>
<th>W0</th>
<th>BN</th>
</tr>
</thead>
<tbody>
<tr>
<td>float32</td>
<td>198.5</td>
<td>98.7</td>
<td>0.783</td>
<td>20</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>Uniform (W4^UP)</td>
<td>25.1</td>
<td>100.3</td>
<td>0.762</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mixed (Wx)</td>
<td>13.9</td>
<td>94.5</td>
<td>0.783</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 5.3. Weights Quantization

We initially compare weights-only UP QAT and MP QAT against the pretrained float32 model, with the same dataset and setting as Sec. 4.5. In all our models the last layer is at float32 precision as it directly produces the regression output in Fig. 2a. We refer to UP models W4^UP and MP models Wx at (fixed or average) bit width b. We report the most compact UP model W4^UP (b = 4) providing best quality metrics, and compare it against the most compact MP model Wx, where x denotes target size Sx, := 14 MB, i.e., bW := 2.35 bit when setting the size constraint.

The results in Fig. 4a allow us to establish that MP is capable of achieving superior performances than UP for the weights-only case, our final results (Sec. 6.3) will consider MP QAT with weights and activations quantization at 4 to 8 bits; we therefore denote them by Wx,b where b', b'' ≥ 2 are average bit widths that set the respective size constraints.

### 6. Results

We present our results using the same quality metrics of most prior works [8, 35]: RMSE, Mean Absolute Error (MAE), Mean Absolute Relative Error (MRE), and δi, i = 1, 2, 3; we also add the MNS metric (5) to study normals similarity. Where reported, the median GPU time per prediction (tGPU) is measured on an NVidia GTX 1080 Ti. The reported results are given on full-frame test sets from Sec. 3.

### 6.1. Training Setup

We train our depth completion model for 40 epochs until convergence using RMSprop as the optimizer with learning rate ρ := 10^{-4} and batch size of 8 (SDS-ST1k) or 4 (NYU-Depth v2). Given its very large resolution, all trainings of our model on SDS-ST1k leverage patches of 160 × 160, but we switch to full 640 × 480 inference when computing quality metrics since our networks are fully convolutional.

We also retrain other competing methods for comparison, by following the supervised training procedures described by the respective authors as reproducible from their codes (S2D, CSPN and NLSPN) or descriptions (D3), while providing as input our sparse ToF datasets. We also tested the authors’ CSPN and NLSPN models pretrained on NYU-Depth v2 with randomly sampled patterns, but the resulting performances were worse than our retraining.

In addition to the float32 model, we consider our quantized MP models Wx,b, Wx,b', Wx,b'' (Sec. 5.4) and Wx, (Sec. 5.3)^a; the subscripts denote the respective average bit widths. For MP QAT, we use the NNabla [19] implementation of [46]. In this case we obtained the best results by training our MP models starting from the pretrained float32 model, with the Adam optimizer for 60 epochs until convergence, starting from learning rate ρ := 10^{-4} with cosine learning rate decay scheduling [32].

### 6.2. Depth Completion Models

We first discuss some quantitative results of our method (in float32 precision) against others in Tab. 2. We achieve very good results on both evaluated datasets despite their different sparsity level K: our method ranks second-best on and activations. Indeed, activations can dominate memory requirements at inference (especially for encoder-decoder networks such as ours) and may additionally cost many read/write accesses of buffer memory, with large impact on energy consumption. As we observed that MP is capable of achieving superior performances than UP for the weights-only case, our final results (Sec. 6.3) will consider MP QAT with weights and activations quantization at 4 to 8 bits; we therefore denote them by Wx,b where b', b'' ≥ 2 are average bit widths that set the respective size constraints.

---

^aIn Wx activations are left float32.

---

^7This tuning can arbitrarily refer to an average bit width or to a total size (in which case the average as defined in (8) can be fractional).
Tables 2 and 3. Depth completion models. The median $t_{\text{GPU}}$ is given as $t_{\text{total}}(t_{\text{initialization}})$. In ours and D$^3$, initialization is computed on CPU.

NYU-Depth v2 with a RMSE of 63.7 mm and best on SDS-ST1k with a RMSE of 98.7 mm. MAE and MNS confirm the RMSE ranking. We also achieve very competitive $t_{\text{GPU}}$ by virtue of the simple initialization obtained via NNI and simple encoder-decoder architecture of Fig. 2a.

NLSPN provides the most competitive performances: it is quality-wise the best approach on NYU-Depth v2 (but the RMSE gap w.r.t. ours is very limited, only 2.3 mm) and second-best on SDS-ST1k, as confirmed by all quality metrics as well as visual results (Sec. 6.4). However, its $t_{\text{GPU}}$ is relatively high even if the model is substantially compact. This is due to spatial propagation, which requires several iterations (18 as recommended in [37]) that linearly increase $t_{\text{GPU}}$ even at constant memory footprint. $t_{\text{GPU}}$ also notably increases significantly between $304 \times 224$ and $640 \times 480$ resolutions. Moreover, the initialization cost (i.e., the input encoder-decoder module of NLSPN) is substantially higher than that of our method, which runs on CPU.

As for the other methods, it is possible to see that CSPN has slightly but consistently inferior performances than NLSPN and ours on both datasets. We also observed that S2D, even if very fast, yields sub-optimal quality metrics on NYU-Depth v2, and poor quality metrics on SDS-ST1k given its larger resolution at lower $K$. This confirms the findings in [34, Sec. 6.4] that the RMSE of S2D deteriorates for lower sparsity level $K$ and that structured subsampling patterns yield worse performances than random ones. Finally, we observed that D$^3$, even if starting from the same initialization as ours, yields worse RMSE and MAE.

6.3. Quantized Network Models

The QAT results for our MP models are given in Tab. 3. Among those models, we highlight our best weights-only MP QAT result for $W_*$; the weights size constraint of 14 MB is met by weights memory size of 13.9 MB after training. This yields 14-fold memory size reduction w.r.t. the float32 model at 198.5 MB. There, we observe limited and graceful degradation of the quality metrics\(^8\) (e.g., the RMSE loss is only 1.2 mm on NYU-Depth v2) when reduc-

\(^8\)We do not report $t_{\text{GPU}}$ as comparable with that of the float32 model and not indicative of computation time on dedicated hardware, since fixed-precision operations are here emulated by float32 operations as in [36,46].
Figure 5. Qualitative results. We report, for three arbitrary frames in the test sets of NYU-Depth v2 (top rows) and SDS-ST1k (bottom rows), the predicted depth maps $\hat{D}$ and error maps $\hat{D} - D_{GT}$ (range: $[-500, 500] mm$). Figure best viewed in color.

6.4. Qualitative Analysis

We now highlight the visual differences observed between different depth completion approaches. We consider the best competitor from Sec. 4.2, NLSPN, against our method in its float32 and MP $W_4A_8$ flavors. We report three samples from each dataset with both depth maps $\hat{D}$ and error maps $\hat{D} - D_{GT}$ in the same range to allow for a visual comparison. On NYU-Depth v2 NLSPN yields sharp edges with minimal amount of “mixed depth” pixels. However, the float32 and $W_4A_8$ models are as sharp, e.g., on the wall discontinuities (middle row). The error patterns seen in all approaches on far planar regions (top row) are probably due to the acquisition of $D_{GT}$ with a Kinect sensor, whose accuracy decreases with the distance [53].

As for the higher-resolution SDS-ST1k dataset, we see that while depth map quality of our models is comparable to that of NLSPN, on several scenes the latter yields larger errors on planar surfaces (e.g., the ground plane) than ours as shown in the corresponding error maps (top, middle row); our model also recovers more accurately some complex object details such as the cupboard (middle row), trash bin and shelves (bottom row).

7. Conclusion

We described a depth completion model based on a compact CNN with mixed-precision quantization. Our model performs within a few percent units of the state-of-the-art in terms of quality metrics on standard datasets such as NYU-Depth v2, but achieves faster GPU time at competitively small memory footprint and is suitable for real-time applications. A dedicated hardware implementation of our QNN will take full advantage of mixed-precision by fixed-point operations. We will investigate knowledge distillation with MP QAT to compress the network, as well as domain adaptation techniques [1, 31] to improve depth map quality on real RGB-ToF datasets.
References


