Distilling Knowledge From a Deep Pose Regressor Network

Muhamad Risqi U. Saputra, Pedro P. B. de Gusmao, Yasin Almalioglu, Andrew Markham, Niki Trigoni
Department of Computer Science, University of Oxford
firstname.lastname@cs.ox.ac.uk

Abstract

This paper presents a novel method to distill knowledge from a deep pose regressor network for efficient Visual Odometry (VO). Standard distillation relies on "dark knowledge" for successful knowledge transfer. As this knowledge is not available in pose regression and the teacher prediction is not always accurate, we propose to emphasize the knowledge transfer only when we trust the teacher. We achieve this by using teacher loss as a confidence score which places variable relative importance on the teacher prediction. We inject this confidence score to the main training task via Attentive Imitation Loss (AIL) and when learning the intermediate representation of the teacher through Attentive Hint Training (AHT) approach. To the best of our knowledge, this is the first work which successfully distill the knowledge from a deep pose regression network. Our evaluation on the KITTI and Malaga dataset shows that we can keep the student prediction close to the teacher with up to 92.95% *parameter reduction and* $2.12\times$ *faster in computation time.*

1. Introduction

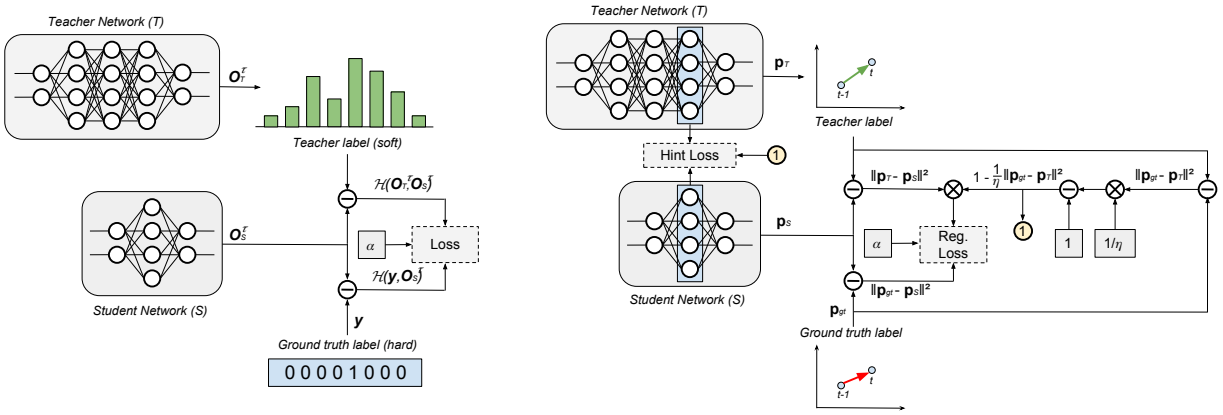
Deep Neural Networks (DNNs) have received increased attention in the last decade due to their success in image and natural language understanding. The availability of large datasets, increased computing power, and advancement of learning algorithms play a pivotal role in their glory. Despite their successes, DNN-based approaches typically require tens or hundreds of million weights. As a consequence, the huge computational and space requirement prevents DNN models from being widely implemented in resource-constrained environment (e.g. mobile phones, quadcopter, etc.). To compound the issue, these applications typically require near real-time inference.

Within the last few years, there have been tremendous efforts towards compressing DNNs. State-of-the-art approaches for network compression such as quantization [9, 6, 17], pruning [12, 10, 35], or low-rank decomposition [30, 2] can yield significant speed-ups but at the cost of accuracy. On the other hand, an approach called Knowledge Distillation (KD) proposed by Hinton et al. [16] offers to recover the accuracy drop by transferring the knowledge of a large teacher model to a small student model. Some recent works show that a small network trained by KD could match or even exceed the accuracy of a large network if it is trained with careful optimization [25].

Most works in network compression, including KD, focus on the problem of classification. KD works very well in classification since it has the advantage of "dark knowledge" which refers to the softened logits output of the teacher. This provides more information than mere one-hot encoding of the class label and contains hidden knowledge about the correlations of class labels [16]. By using the logits output for training, the student network can emulate the generalization capability of the teacher network. However, this advantage does not exist in **regression**. In the regression problem, a deep regression network predicts sequential, continuous, values which have the exact same characteristics as the ground truth, with the exception of being plagued with an unknown error distribution. Without access to any dark knowledge, it is unclear how KD could help in compressing a regression network. In recent surveys, it is even stated that the main drawback of KD is that it only works for classification problems [5].

KD methods for classification [16, 25, 20, 36, 32, 19, 23] rely solely on the teacher prediction without considering the error made w.r.t. ground truth. In regression however, the real-valued predictions are unbounded, and hence the teacher can give highly erroneous guidance to the student network. Previous work [4] alleviated this issue by using teacher loss as an upper bound. However, it was designed for standard bounding box regression which has different characteristic to pose regression as it belongs to $SE(3)$ (Lie Groups). Moreover, they directly transferred the knowledge from the teacher without filtering which one is good and which one is bad. To this end, our novel insight is to use the teacher loss as a confidence score to decide when we can trust the teacher. We demonstrate that this is key to successfully distilling deep pose regression networks.

We will demonstrate our work in distillation for cam-



(a) KD in Classification

(b) KD in Regression

Figure 1. Comparison between (a) standard Knowledge Distillation applied to classification problem and (b) our Knowledge Distillation approach applied to regression problem. Note that in regression, we are unable to use the dark knowledge provided by soft teacher labels.

era pose regression problem which is widely known as Visual Odometry (VO). In particular, we employ DNN-based VO methods [33, 34, 37, 38, 26, 1] which replaces the conventional VO pipeline based on multiple-view geometry [15, 27] to DNN-based pipeline which automatically learns useful features for estimating 6 Degree-of-Freedom (DoF) camera poses. To the best of our knowledge, this work is a first attempt to distill the knowledge from a deep pose regression network. Our key contributions are:

- We study different ways to blend the loss of the student both w.r.t. ground truth and w.r.t. teacher, and propose to use the teacher loss as a confidence score to attentively learn from examples that the teacher is good at predicting through Attentive Imitation Loss (AIL).
- We also propose Attentive Hint Training (AHT) as a novel way to learn the intermediate representation of the teacher, based on the teacher loss.
- We perform extensive experiment on KITTI and Malaga datasets which show that our proposed approach can reduce the number of student parameters by up to 92.95% ($2.12\times$ faster) whilst keeping the prediction accuracy very close to that of the teacher.

2. Related Work

Many compression methods have been recently developed. Besides KD, there are other approaches available such as quantization, pruning, and low-rank decomposition.

Network quantization reduces the number of bits required to compress the network. The quantization could be applied by using 16-bit or 8-bit representation as proposed by [31, 11]. As an extreme case, a 1-bit representation (or binary network) could also be used as seen in [6, 17]. By restricting the weights into two possible values (e.g. -1 or 1), binary networks can dramatically reduce both computation time and memory consumption at the cost of significant reduction in accuracy [5].

Pruning, as the name implies, removes redundant and non-informative weights. Weight pruning can be done by using magnitude-based method [14, 10] or dropout-based method [35, 22]. The pruning can be applied in the individual neuron connection [14, 13] or in convolutional filter itself [21]. This approach promises significant parameter reduction without greatly affecting accuracy, although it typically requires more training stages [13]. However, in practice, it requires additional implementation of sparse matrix multiplication which possibly needs more resource consumption and specialized hardware and software [21].

Low-rank decomposition reduces DNN complexity by exploiting the low-rank constraints on the network weights. Most approaches, such as [24, 18, 2], factorize a fully connected or convolutional layer through matrix/tensor decomposition techniques such as SVD or CP-decomposition. By using this factorization technique, the number of matrix multiplications becomes smaller than the original network. However, the obtained compression ratio is generally lower than the pruning-based approach and the low-rank constraint imposed on the network might impact the network performance if the rank is not selected with care.

3. Background: Distillation for Classification

Knowledege Distillation (KD) is an approach to transfer the knowledge of a large teacher network to a small student network. The main idea of KD is to allow the student to capture the finer structure learned by the teacher instead of learning solely from the true labels. Let T be the teacher network where $\mathbf{O}_T = \text{softmax}(\mathbf{a}_T)$ is the teacher output

probability and $\mathbf{a}_T$ is the teacher's logits (pre-softmax output). A student network S with $\mathbf{O}_S = \text{softmax}(\mathbf{a}_S)$ as the prediction and $\mathbf{a}_S$ as the logits is trained to mimic $\mathbf{O}_T$. Since $\mathbf{O}_T$ is usually very close to the one-hot representation of the class labels, a temperature $\tau > 1$ is used to soften the output probability distribution of T. The same temperature is used for training S such that $\mathbf{O}_T^\tau = \text{softmax}(\frac{\mathbf{a}_T}{\tau})$ and $\mathbf{O}_S^\tau = \text{softmax}(\frac{\mathbf{a}_S}{\tau})$, but $\tau = 1$ is then used for testing S. If $\mathcal{H}$ is the cross-entropy and $\mathbf{y}$ is the one-hot encoding of the true labels, then KD objective function is formed by minimizing both hard label ($\mathbf{y}$) error and soft label error (illustrated in Fig. 1 (a)) as follows

$$\mathcal{L}_{KD} = \alpha\mathcal{H}(\mathbf{y}, \mathbf{O}_S) + (1-\alpha)\mathcal{H}(\mathbf{O}_T, \mathbf{O}_S) \quad (1)$$

where α is a parameter to balance both cross-entropies.

The KD formulation with softened outputs ($\tau > 1$) in (1) gives more information for S to learn, as it provides information about the relative similarity of the incorrect predictions [16], [25]. For example, T may mistakenly predict an image of a car as a truck, but that mistake still has a much higher probability than mistaking it for a cat. These relative probabilities of incorrect prediction convey how T tends to generalize to new data [16]. However, this advantage does not exist in the regression problem. As seen in Fig. 1 (b), both teacher and ground truth label have the same characteristic. Intuitively, we would prefer to minimize S's prediction directly w.r.t. the ground truth labels since the teacher label is plagued with an unknown error distribution. However, our experiments show that training S only with the ground truth labels gives very poor results.

4. Blending Teacher, Student, and Imitation Loss

As it is unclear how we can take advantage of T's prediction in distilling regression networks, we study different ways to blend together the loss of S's prediction w.r.t. ground truth and w.r.t T's prediction. For simplicity, we refer to the error of S w.r.t ground truth as *student loss* and the loss of T w.r.t ground truth as *teacher loss*. We refer to *imitation loss* ($\mathcal{L}_{imit}$) for the errors of S w.r.t. T (since S tries to imitate T's prediction). The following outlines different formulations and rationales of blending teacher, student, and imitation loss.

Minimum of student and imitation. In the simplest formulation, we assume that T has good prediction accuracy in all conditions. In this case, as T's prediction will be very close to the ground truth, it does not really matter whether we minimize S w.r.t. ground truth or w.r.t. T. Then, we simply minimize whichever one is smaller between the student loss and imitation loss as follows

$$\mathcal{L}_{reg} = \frac{1}{n}\sum_{i=1}^{n} \min\left(\left\|\mathbf{p}_S - \mathbf{p}_{gt}\right\|^2, \left\|\mathbf{p}_S - \mathbf{p}_T\right\|^2\right) \quad (2)$$

where $\mathbf{p}_S$, $\mathbf{p}_T$, and $\mathbf{p}_{gt}$ are S's prediction, T's prediction, and ground truth labels respectively.

Imitation loss as an additional loss. Instead of seeking the minimum between the student and imitation loss, we can use the imitation loss as an additional loss term for the student loss. In this case, we regard the imitation loss as another way to regularize the network and prevent the network from overfitting [16]. Then, the objective function becomes

$$\mathcal{L}_{reg} = \frac{1}{n}\sum_{i=1}^{n} \alpha\left\|\mathbf{p}_S - \mathbf{p}_{gt}\right\|^2 + (1-\alpha)\left\|\mathbf{p}_S - \mathbf{p}_T\right\|^2 \quad (3)$$

where α is a scale factor used to balance the student and imitation loss. This formulation is similar to the original formulation of KD for classification as seen in (1) except the cross-entropy loss is replaced by regression loss.

Teacher loss as an upper bound. Equations (2) and (3) assume that T has very good generalization capability in most conditions. However in practice, T can give very erroneous guidance for S. There is a possibility that in adverse environments, T may predict camera poses that are contradictory to the ground truth pose. Hence, instead of directly minimizing S w.r.t. T, we can utilize T as an upper bound. This means that S's prediction should be as close as possible to the ground truth pose, but we do not add additional loss for S when its performance surpasses T [4]. In this formulation, (3) becomes the following equation

$$\mathcal{L}_{reg} = \frac{1}{n}\sum_{i=1}^{n} \alpha\left\|\mathbf{p}_S - \mathbf{p}_{gt}\right\|^2 + (1-\alpha)\mathcal{L}_{imit} \quad (4)$$

$$\mathcal{L}_{imit} = \begin{cases} \left\|\mathbf{p}_S - \mathbf{p}_T\right\|^2, & \text{if } \left\|\mathbf{p}_S - \mathbf{p}_{gt}\right\|^2 > \left\|\mathbf{p}_T - \mathbf{p}_{gt}\right\|^2 \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Probabilistic imitation loss (PIL). As stated before, T is not always accurate in practice. Since there is some degree of uncertainty in T's prediction, we can explicitly model this uncertainty with a parametric distribution. For example, we can model the imitation loss using Laplace's distribution

$$\mathbb{P}\left(\mathbf{p}_S|\mathbf{p}_T, \sigma\right) = \frac{1}{2\sigma}\exp\frac{-\left\|\mathbf{p}_S - \mathbf{p}_T\right\|}{\sigma} \quad (6)$$

where σ is an additional quantity that S should predict. In this case, the imitation loss is turned into minimizing the negative log likelihood of (6) as follows

$$-\log\mathbb{P}\left(\mathbf{p}_S|\mathbf{p}_T, \sigma\right) = \frac{\left\|\mathbf{p}_S - \mathbf{p}_T\right\|}{\sigma} + \log\sigma + \text{const.} \quad (7)$$

The final objective is retrieved by replacing $\mathcal{L}_{imit}$ in (4) with (7). We can view (7) as a way for S to learn suitable coefficient (via σ) to down-weight unreliable T's prediction. Besides Laplacian distribution, another parametric distribution like Gaussian can be used as well.

Attentive imitation loss (AIL). The main objective of modeling the uncertainty in the imitation loss is that we could then adaptively down-weight the imitation loss when a particular T's prediction is not reliable. However, modeling T's prediction with a parametric distribution may not accurately reflect the error distribution of T's prediction. Hence, instead of relying on S to learn a quantity σ to down-weight unreliable T's prediction, we can use the empirical error of T's prediction w.r.t. ground truth (which is the teacher loss) to do the job. Then, the objective function becomes

$$\mathcal{L}_{reg} = \frac{1}{n}\sum_{i=1}^{n} \alpha \left\|\mathbf{p}_S - \mathbf{p}_{gt}\right\|_i^2 + (1-\alpha)\Phi_i \left\|\mathbf{p}_S - \mathbf{p}_T\right\|_i^2 \tag{8}$$

$$\Phi_i = \left(1 - \frac{\left\|\mathbf{p}_T - \mathbf{p}_{gt}\right\|_i^2}{\eta}\right) \tag{9}$$

$$\eta = \max\left(e_T\right) - \min\left(e_T\right) \tag{10}$$

$$e_T = \{\left\|\mathbf{p}_T - \mathbf{p}_{gt}\right\|_j^2 : j = 1, ..., N\} \tag{11}$$

where Φ_i is the normalized teacher loss for each i sample, e_T is a set of teacher loss from entire training data, and η is a normalization parameter that we can retrieve from subtracting the maximum and the minimum of e_T. Note that $\|.\|_i$ and $\|.\|_j$ are not p-norm symbol. Instead we use i and j in $\|.\|_i$ and $\|.\|_j$ as index to differentiate which loss is computed from the batch samples ($i = 1, ..., n$) and which loss is calculated from entire training data ($j = 1, ..., N$).

Fig. 1 (b) shows how each component in (8)-(11) blends together. Note that we still keep α to govern the relationship between student and imitation loss. In this case, Φ_i's role is to put different relative importance, hence it is called *attentive*, for each component in the imitation loss as seen in weighted sum operation. Notice that (8) can be rearranged into $\mathcal{L}_{reg} = \frac{\alpha}{n}\sum_{i=1}^{n}\left\|\mathbf{p}_S - \mathbf{p}_{gt}\right\|_i^2 + \frac{1-\alpha}{n}\sum_{i=1}^{n}\Phi_i\left\|\mathbf{p}_S - \mathbf{p}_T\right\|_i^2$. As Φ_i is computed differently for each image sample and is intended to down-weight unreliable T's prediction, we could also say that by multiplying the imitation loss with Φ_i, we rely more on the example data which T is good at predicting in the process of knowledge transfer between T and S.

5. Learning Intermediate Representations

Blending teacher, student, and imitation loss have set the objective function for the main KD task. Another important aspect in KD's transfer process is Hint Training (HT). HT is the process of training the intermediate representation of S such that it could mimic the latent representation of T. It was designed as an extension of the original KD [16] and formulated by [25] to transfer the knowledge of T to S with deeper but thinner architecture. Even if it is devised to help training S with deeper layers than T, we would argue that it is also an important step for training a regressor network with shallow architecture. HT could act as another way to regularize S such that it could better mimic the generalization capability of T [25].

In Hint Training, a *hint* is defined as a layer in T that is used to guide a *guided* layer in S. Let $\mathbf{W}_{guided}$ and $\mathbf{W}_{hint}$ be the parameters of S and T up to their guided and hint layers respectively. With the standard HT formulation, we can train S up to the guided layer by minimizing $\mathcal{L}_{hint} = \frac{1}{n}\sum_{i=1}^{n}\left\|\Psi_T(\mathbf{I};\mathbf{W}_{hint}) - \Psi_S(\mathbf{I};\mathbf{W}_{guided})\right\|^2$, where Ψ_T and Ψ_S are T's and S's deep neural functions up to their respective hint or guided layers. The drawback with this formulation is that it does not take into account the fact that T is not a perfect function estimator and can give incorrect guidance to S. While in Section 4 we describe how to tackle this issue through down-weighting unreliable T prediction by multiplying it with normalized teacher loss, we argue that this step is also required for HT. Then, we propose a modification of HT termed *Attentive Hint Training* (AHT) as follows:

$$\mathcal{L}_{hint} = \frac{1}{n}\sum_{i=1}^{n}\Phi_i\left\|\Psi_T(\mathbf{I};\mathbf{W}_{hint}) - \Psi_S(\mathbf{I};\mathbf{W}_{guided})\right\|_i^2 \tag{12}$$

where Φ_i is the normalized teacher loss as seen in (9). While (8) and (12) can be trained jointly, we found out that training separately yields superior performance especially in absolute pose error. Then, the knowledge transfer between T and S becomes 2 stages optimization procedures. The 1st stage trains S up to the guided layer with (12) as the objective. The 2nd stage trains the remaining layer of S (from guided until the last layer) with (8) as the objective.

6. Implementation Details

6.1. Camera Pose Regression with DNNs

As we demonstrate our distillation approach for Visual Odometry (VO) problem, we will briefly review VO approaches. Conventional VO estimates the camera poses by finding feature correspondences between multiple images and applying multiple-view geometry techniques [15, 27]. On the other hand, DNNs learn the camera ego-motion directly from raw image sequences by training the network in an end-to-end manner. Let $\mathbf{I}_{t-1,t} \in \mathbb{R}^{2\times(w\times h\times c)}$ be two concatenated images at times $t-1$ and t, where w, h, and c are the image width, height, and channels respectively. DNNs essentially learn a mapping function to regress the 6-DoF camera poses $\{(\mathbb{R}^{2\times(w\times h\times c)})_{1:N}\} \rightarrow \{(\mathbb{R}^6)_{1:N}\}$, where N are the total number of image pairs. In the supervised case, learning 6-DoF camera poses can be achieved by minimizing the discrepancy between the predicted poses $\mathbf{p}_{pr} \in \mathbb{R}^6$ and the ground truth poses $\mathbf{p}_{gt} \in \mathbb{R}^6$ as fol-

lows $\mathcal{L}_{reg} = \frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{p}_{pr} - \mathbf{p}_{gt}\right\|^2$, given n sample images. However, since translation and rotation have different constraints, we usually decompose $\mathcal{L}_{reg}$ into

$$\mathcal{L}_{reg} = \frac{1}{n}\sum_{i=1}^{n}\beta\left\|\mathbf{t}_{pr} - \mathbf{t}_{gt}\right\|^2 + (1-\beta)\left\|\mathbf{r}_{pr} - \mathbf{r}_{gt}\right\|^2 \tag{13}$$

where $\mathbf{t} \in \mathbb{R}^3$ and $\mathbf{r} \in \mathbb{R}^3$ are the translation and rotation components in x, y, and z axes. $\beta \in \mathbb{R}$ is used to balance $\mathbf{t}$ and $\mathbf{r}$. Here the rotation part is represented as an Euler angle. Another representation such as quaternion or rotation matrix can be used as well [33].

To apply our distillation approach to DNN-based VO, we decompose each translation and rotation component in (13) to (8). We also decompose AHT formulation in (12) into translation and rotation representation, and apply Φ_i differently for each representation. As the teacher loss distribution is also different for translation and rotation (as seen in Fig. 2), η in (9) is computed differently for each of them.

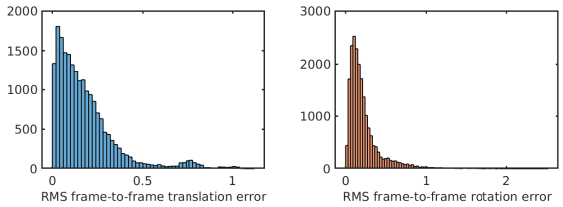


Figure 2. Empirical error distribution of the teacher network for translation and rotation on KITTI dataset Seq 00-08.

6.2. Network Architecture

We employ ESP-VO [34] for the teacher network T in which the architecture is depicted in Fig. 3 (left). It consists of two main parts, namely the feature extractor network and a pose regressor network. The feature extractor is composed from a series of Convolutional Neural Networks (CNNs) to extract salient features for VO estimation. Since VO estimates the camera pose between consecutive frames, optical-flow like feature extractor network (FlowNet [7]) is used to initialize the CNNs. The pose regressor consists of Long-Short Term Memory (LSTM) Recurrent Neural Networks (RNNs) and Fully Connected (FC) Layers to regress 6-DoF camera poses. The LSTM is utilized to learn long-term motion dependencies among image frames [33].

Fig. 3 (right) depicts S with 92.95% distillation rate (d_{rate}). The main building blocks of S are essentially the same as T except we remove a number of layers from T to construct a smaller network. To specify the structure of S, in general, we can remove the layers from T which contribute the most to the number of weights, but S should still consist of a feature extractor (CNN) and a regressor (LSTM/FC). In the feature extractor, the largest number of weights usually corresponds to the few last layers of CNNs, while in the regressor part it corresponds to the LSTM layers. Thus, for $d_{rate} = 92.95\%$, we remove the last five layers of the CNN and the two RNN-LSTM layers. However, we still initialize the CNN with the FlowNet's weight as in ESP-VO. To compensate for the loss of removing the CNN and LSTM layers, we add 1 FC layer in the regressor part for $d_{rate} < 75\%$ and 2 FC layers for $d_{rate} > 75\%$. This corresponds to fewer than 1% additional parameters.

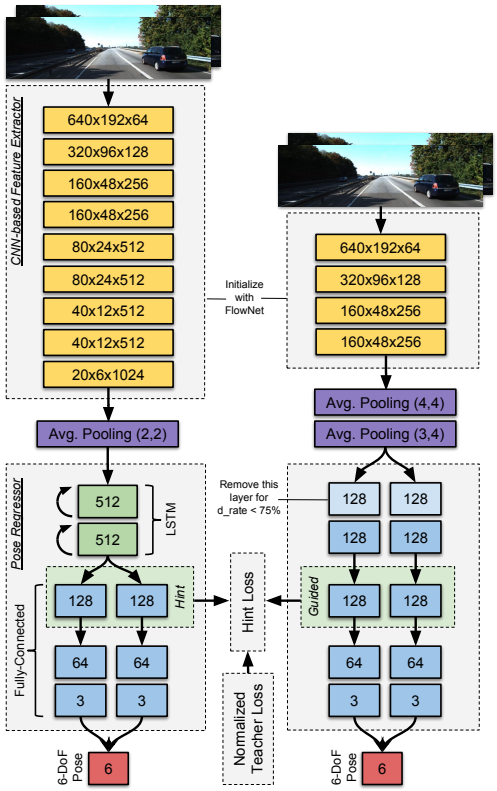


Figure 3. Details of network architecture for teacher (left) and student network with 92.95% distillation rate (right).

6.3. Training Details

As stated in Section 5, we employ two stages of optimization. The first stage is training the intermediate representation of S through AHT. As seen in Fig. 3, we select the 1st FC layer of T as a hint and the 3rd FC layer of S (or the 2nd FC layer for $d_{rate} < 75\%$) as the guided layer. We used the FC layer of T as a hint not only to provide easier guidance for training S, since both FC layers in T and S have the same dimensions, but also to transfer the ability of T to learn the long-term motion dynamics of camera poses as the FC layer of T is positioned after the RNN-LSTM layers. In the second stage, we freeze weights of S trained from the first stage and train the remaining layers of S using (8) as the objective.

7. Experimental Results

7.1. Experiment Environments

We implemented T and S in Keras. We employed NVIDIA TITAN V GPU for training and NVIDIA Jetson TX2 for testing. The training for each stage goes up to 30 epochs. For both training stages, we utilize Adam Optimizer with $1e-4$ learning rate. We also applied Dropout [28] with 0.25 dropout rate for regularizing the network. For the data, we used KITTI [8] and Malaga odometry dataset [3]. We utilized KITTI Seq 00-08 for training and Seq 09-10 for testing. Before training, we reduced the KITTI image dimension to 192×640. We only use Malaga dataset for testing the model that has been trained on KITTI. For this purpose, we cropped the Malaga images to the KITTI image size. Since there is no ground truth in Malaga dataset, we perform qualitative evaluation against GPS data.

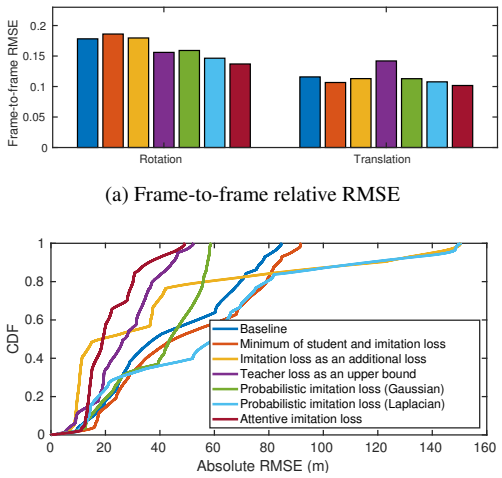
7.2. Metrics

In this work, we want to measure the trade-off between accuracy and parameter reduction. In VO, accuracy can be measured by several metrics. We use Root Mean Square (RMS) Relative Pose Error (RPE) for translation (**t**) and rotation (**r**) and RMS Absolute Trajectory Error (ATE) as they has been widely used in many VO or SLAM benchmarks [29]. For parameter reduction, we measure the percentage ($\%$) of S's parameters w.r.t. T's parameters. We also measure the associated computation time (ms) and the model size (MB) for each reduction rate.

7.3. Sensitivity Analysis

The Impact of Different Methods for Blending Teacher, Student, and Imitation Loss. In this experiment, we want to understand the impact of different approaches to blending teacher, student, and imitation loss as described in Section 4. We used S with $d_{rate} = 72.78\%$ constructed from removing the last 3 CNNs and replacing 2 LSTMs with 1 FC layer. In order to get a fair comparison without having bias from the AHT process, we trained S with standard HT approach in the first stage. Then, we trained the remaining layer(s) with all formulations described in Section 4 in the second stage. For additional comparison, we add a baseline approach, in which we only minimize the student loss.

Fig. 4 (a) and (b) depicts the RPE and the CDF of ATE of different methods in blending the losses. It can be seen that AIL has the best accuracy in both RPE and ATE. This indicates that distilling knowledge from T to S only when we trust T does not reduce the quality of knowledge transfer, but instead improve the generalization capability of S. Two approaches (minimum of student and imitation; imitation loss as additional loss) that rely on the assumption that T's prediction is always accurate have inferior performance even if compared to the baseline. PIL, either using Laplacian or Gaussian, yields good accuracy in RPE, but lacks robustness since they have larger overall drift (as seen in Fig. 4 (b)). This is probably due to the failure of the parametric distribution function to model the teacher error distribution accurately. The upper bound objective has good balance between RPE and ATE but the performance is inferior to AIL.



(a) Frame-to-frame relative RMSE

(b) CDF of Absolute Trajectory Errors

Figure 4. The impact of different ways in blending teacher, student, and imitation loss to (a) RPE and (b) ATE. Same legend is used for both graphs.

Table 1. The impact of using Attentive Imitation Loss (AIL) and Attentive Hint Training (AHT) algorithm

	Architecture CNN-FC[a]	HT	Final Obj.	Rec. Error[b]	ATE (m)
1	6 - 2 (72.88%)	-	student	0.6242	80.842
2	6 - 2 (72.88%)	HT	student	0.0252	65.251
3	6 - 2 (72.88%)	HT	AIL	0.0252	36.459
4	6 - 2 (72.88%)	AHT	student	**0.0166**	52.320
5	6 - 2 (72.88%)	AHT	AIL	**0.0166**	**32.259**
6	5 - 3 (79.69%)	-	student	0.1341	68.350
7	5 - 3 (79.69%)	HT	student	0.0177	53.661
8	5 - 3 (79.69%)	HT	AIL	0.0177	29.751
9	5 - 3 (79.69%)	AHT	student	**0.0168**	37.645
10	5 - 3 (79.69%)	AHT	AIL	**0.0168**	**25.857**

[a] Total FC layers until intermediate layer used for HT and AHT. The number in the bracket indicates d_{rate}.

[b] Reconstruction error of S's output intermediate representation w.r.t. T's output intermediate representation.

The Impact of Attentive Hint Training. As we want to inspect the effect of the proposed AHT approach, we trained the model with 3 different procedures: without HT, with HT, and with AHT. We also alternate between using the student loss and AIL to see the effect of applying attentive transfer mechanism in both intermediate (as AHT) and final layer (as AIL), or only in one of them. We used the same model architecture as the previous ablation to conduct this experiment. We compare RMS Reconstruction Error

of S's output latent representation w.r.t. T's representation and ATE w.r.t. ground truth.

Table 1 lists the results of this study which clearly shows that as soon as HT is applied, S's reconstruction error w.r.t. T reduces dramatically (see row [1, 2] or [6, 7]). This shows that without having guidance in the intermediate layer, it is very difficult for S to imitate the generalization capability of T. AHT then reduces further the reconstruction error of HT by giving different relative importance to T's representation and placing more emphasis on representations that produce accurate T predictions. Fig. 5 visualizes the output latent representation for different training procedures. It can be seen that AHT's output representation is very close to T. Slight differences with T's representation are due to different relative importance placed on T's predictions. However, even if AHT does not try to blindly imitate T's representation, the average reconstruction error is still lower than the HT approach which attempts to perfectly imitate T's representation (see Table 1 row [2, 4] or [7, 9]).

The last column of Table 1 shows ATE for different combinations of applying attentive knowledge transfer. As it can be seen in row [2, 4] (or [7, 9]) that applying attentive loss in the intermediate layer (AHT) can significantly reduce the ATE of S. However, the reduction rate is not as large as when applying it in the final task (AIL) (see Table 1 row [2, 3] or [7, 8]) as it can reduce the ATE up to $1.8\times$ smaller. This is sensible because the accuracy of a DNN model depends on the output from the final task. A better guidance in the final layer (main task) can yield stronger performance than a better guidance in the intermediate layer. Finally, applying attentive loss in both intermediate (AHT) and final layers (AIL) consistently gives the best result for 6 and 5 CNNs architecture (see Table 1 row 5 and 10).

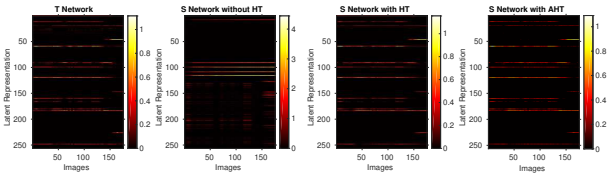


Figure 5. The difference of latent feature representation between T and S, trained without HT, with HT, and with AHT.

7.4. Trade-off between Accuracy, Model Size, and Execution Time

In this experiment, we want to understand the trade-off between the model size, execution time, and accuracy for different d_{rate}. Fig. 6 shows that our proposed distillation approach can keep S very close to T up to $d_{rate} = 92.95\%$. It can even achieve better performance than the teacher for $d_{rate} = 65.77\%$ and 79.69% as T might be over-parameterized (see also the output trajectory in Fig. 7). For $d_{rate} > 92.95\%$, the performance starts to degrade more rapidly as it becomes too difficult to transfer the knowledge from T to S without other constraints. It can also be seen that if S is trained directly to fit the ground truth with hard loss (supervised student), it shows very poor performance.

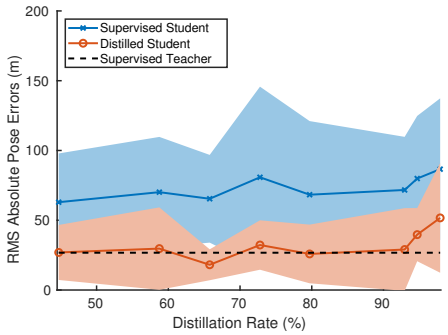


Figure 6. RMS absolute pose errors between Supervised and Distilled Student for different d_{rate}.

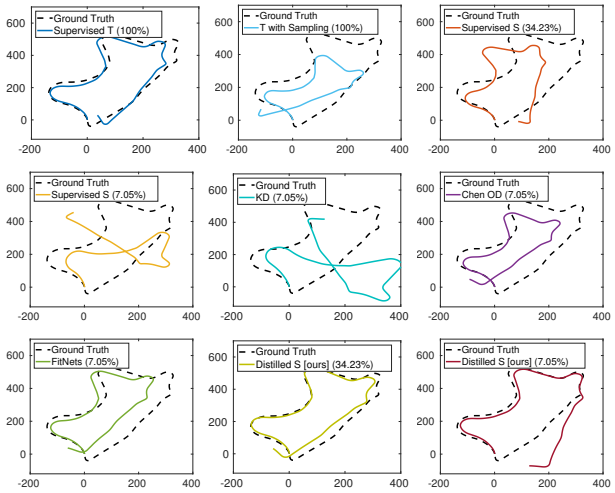


Figure 7. Trajectory prediction from T and S trained with various distillation approaches in KITTI Seq 09. The number in the bracket indicates the percentage of S parameters w.r.t. T.

Table 2. Trade-off between the number of parameters, model size, computation time, and accuracy (ATE)

Network (Weights %)	Parameters (millions)	Size (MB)	Ex. Time (ms)	ATE (m)
T (100%)	33.64	286.9	87	26.74
S (55.28%)	18.59	74.6	82	26.92
S (41.25%)	13.88	55.7	71	29.69
S (34.23%)	11.52	46.3	62	18.09
S (27.22%)	9.16	36.8	58	32.26
S (20.30%)	6.83	27.5	47	25.86
S (7.05%)	2.37	7.3	41	29.03

Table 2 shows the comparison between T and S in terms of number of parameters, model size, and computation time. As we can see, with $d_{rate} = 92.95\%$ we can reduce the model size from 286.9MB to 7.3MB (2.5%). Removing 2 LSTMs, which are responsible for 44.72% of T's parame-

Table 3. Comparison with other distillation approaches

Method	RMS RPE (**t**)	RMS RPE (**r**)	RMS ATE
Supervised T	0.1197	0.2377	26.7386
Supervised S	0.1367	0.1627	71.7517
KD [16]	0.1875	0.1439	165.2182
Chen's OD [4]	0.1197	0.1416	46.2320
FitNets [25]	0.1450	0.1409	31.9624
Ours	**0.1053**	**0.1406**	**29.0294**

ters can already reduce T's model size to 78MB (27%) but it has less impact in the computation time as the LSTM has been implemented efficiently for NVIDIA cuDNN. With $d_{rate} = 92.95\%$, we reduce the computation time from 87ms to 41ms (2.12×), effectively doubling the frame rate. This has significant practical implication. If we re-train T given subsampled images such that the frame rate is similar to S with $d_{rate} = 92.95\%$, T's prediction accuracy will degrade 160% (see the trajectory in Fig. 7). This is probably due to the difficulty of estimating accurate optical flow representation in large stereo baseline. Meanwhile with the same computation budget, the distilled S yields stronger performance than the subsampled T with only 8.63% accuracy drop w.r.t supervised T as seen in Fig. 8.

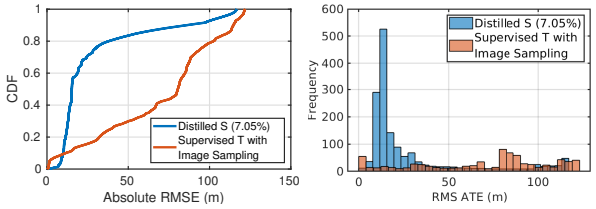


Figure 8. Distribution and histogram of ATE between distilled S and supervised T with image sampling.

7.5. Comparison with Related Works

As there is no specific KD for pose regression, to compare our proposed KD with other related works, we used some well known KD approaches for classification and object detection. However, we modify the objective function to fit our regression problem. We used 3 baselines for this experiment: KD [16], FitNets [25], and Chen's Object Detection (OD) model [4]. For KD [16], we trained with standard training procedure (without HT or AHT) and replaced the objective function with (3). For FitNets [25], we used HT approach for the 1st stage of training and utilize (3) as the objective in the 2nd stage. For Chen's OD [4], we also used standard HT for the 1st stage and employ (4) as the objective in the 2nd stage. For all models, we used $d_{rate} = 92.95\%$ to train S as an extreme example (see supplementary material for experiment with $d_{rate} = 65.77\%$).

Table 3 shows the result of this experiment. It can be seen that our proposed approach have better accuracy for

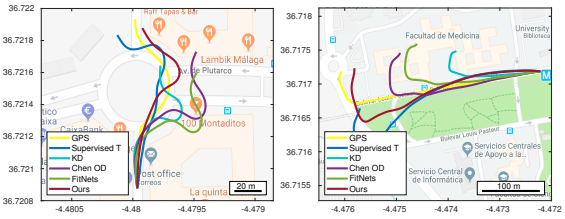


Figure 9. Qualitative evaluation in Malaga dataset Seq 04 and Seq 09. All model are only trained on KITTI Seq 00-08.

both RPE and ATE. Even if most of the competing approaches have better RPE than the supervised T, it has huge bias in the relative pose prediction such that the integration of these relative poses yields very large ATE. T tackle this bias to some extent by using LSTM layers which are supposed to learn the long-term motion dynamic of the camera poses [33]. Since S removes the LSTM layers, most approaches fail to recover this knowledge from T but our proposed approach is able to reduce ATE by focusing to learn the good predictions from T. Fig. 7 shows the comparison of the output trajectory between the competing approaches on KITTI dataset. It can be seen that our distillation output trajectory is closer to T than the competing approaches.

Fig. 9 depicts the qualitative evaluation in Malaga dataset. Note that we have not trained on this dataset, demonstrating generalization capacity. We can see that our proposed approach yields a closer trajectory to GPS, even when it is compared to T. This signifies that our distillation approach yields good generalization ability even when it is tested on a different dataset. This result also shows that T may overfit the training data as it has many redundant parameters. However, this redundancy seems necessary for the initial stage of training as a DNN requires large degree-of-freedom to find better weights and connections [16]. Meanwhile, directly training S without any supervision from T seems to be very difficult. Our results show that we can have better generalization when distilling large degree-of-freedom T to small degree-of-freedom S if we transfer the knowledge from T to S only when we trust T.

8. Conclusion

We have presented an approach to distill the knowledge from a deep pose regressor network to a much smaller network with a small loss of accuracy. We have shown that the teacher loss can be used as an effective attentive mechanism to transfer the knowledge between teacher and student. For future work, we will investigate whether another compression technique can be combined with our distillation approach to further reduce the computation time.

Acknowledgement. This research is funded by the US National Institute of Standards and Technology (NIST) Grant No. 70NANB17H185. M. R. U. Saputra was supported by Indonesia Endowment Fund for Education (LPDP).

References

[1] Yasin Almalioglu, Muhamad Risqi U. Saputra, Pedro P. B. de Gusmao, Andrew Markham, and Niki Trigoni. Ganvo: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 2

[2] Sourav Bhattacharya and Nicholas D. Lane. Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, pages 176–189, 2016. 1, 2

[3] José-Luis Blanco-Claraco, Francisco-Ángel Moreno-Dueñas, and Javier González-Jiménez. The málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario. *The International Journal of Robotics Research (IJRR)*, 33(2):207–214, 2014. 6

[4] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 742–751, 2017. 1, 3, 8

[5] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018. 1, 2

[6] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9, 2015. 1, 2

[7] Alexey Dosovitskiy, Philipp Fischery, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *IEEE International Conference on Computer Vision (ICCV)*, volume 11-18-Dece, pages 2758–2766, 2016. 5

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 6

[9] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing Deep Convolutional Networks using Vector Quantization. In *arXiv:1412.6115*, 2015. 1

[10] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic Network Surgery for Efficient DNNs. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 1, 2

[11] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning (ICML)*, pages 1737–1746, 2015. 2

[12] Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations (ICLR)*, 2016. 1

[13] Song Han, Huizi Mao, Enhao Gong, Shijian Tang, and J. Dally. DSD : Dense-Sparse-Dense Training for Deep Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[14] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both Weights and Connections for Efficient Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9, 2015. 2

[15] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. 2, 4

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop (2015)*, pages 1–9, 2015. 1, 3, 4, 8

[17] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 1, 2

[18] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014. 2

[19] Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. Self-supervised knowledge distillation using singular value decomposition. In *European Conference on Computer Vision (ECCV)*, pages 339–354. Springer, 2018. 1

[20] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations (ICLR)*, 2016. 1

[21] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5058–5066, 2017. 2

[22] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational Dropout Sparsifies Deep Neural Networks. In *International Conference on Machine Learning (ICML)*, 2017. 2

[23] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[24] Roberto Rigamonti, Amos Sironi, Vincent Lepetit, and Pascal Fua. Learning separable filters. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2754–2761, 2013. 2

[25] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for Thin Deep Nets. In *International Conference on Learning Representations (ICLR)*, 2015. 1, 3, 4, 8

[26] Muhamad Risqi U. Saputra, Pedro P. B. de Gusmao, Sen Wang, Andrew Markham, and Niki Trigoni. Learning monocular visual odometry through geometry-aware curriculum learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 2

[27] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual SLAM and Structure from Motion in Dynamic Environments : A Survey. *ACM Computing Surveys*, 51(2), 2018. 2, 4

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 6

[29] Jurgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. IEEE, 2012. 6

[30] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, and Weinan E. Convolutional neural networks with low-rank regularization. In *International Conference on Learning Representations (ICLR)*, 2016. 1

[31] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, volume 1, page 4. Citeseer, 2011. 2

[32] Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2769–2775, 2018. 1

[33] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 2, 5, 8

[34] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research (IJRR)*, pages 1–30, 2018. 2, 5

[35] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher. DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework. In *15th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, number 17, 2017. 1, 2

[36] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4133–4141, 2017. 1

[37] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 340–349, 2018. 2

[38] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018. 2