Compact Depth-wise Separable Precise Network For Depth Completion

by

Tianqi Wang

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Predicting a dense depth map from LiDAR scans and synced RGB images with a small deep neural network is a challenging task. Most top-accuracy methods boost precision by having a very large number of parameters and as a result huge memory consumption. Whereas, depth completion tasks are commonly tackled in areas like autonomous driving such that edge devices, which are powered by embedded GPU, are the main platform. In this work, we present a methodology to produce an efficient depth completion model with high-performance fidelity inspired by PENet [1]. To create a compact model, we propose replacing convolutional encoder layers with depth-wise separable convolution and transpose convolutional decoder with up-sampling plus depth-wise separable convolution. Our technique of using random layer pruning as a stability test guilds the design of the architecture and avoids over-parameterization. We also provide a simple but robust knowledge distillation method to further turbocharge the network and make our model scalable to fulfill a better quality requirement. Our experiments have shown a decent improvement over the leading compact models while using significantly fewer parameters compared with other larger models.

Acknowledgements

I would like to express my gratitude to my supervisor Nilanjan Ray for his guidance and support throughout this study. I would like to especially thank Dr. Sara Elkerdawy for her support and advice in the model compression. Without their help, this thesis would not have been possible. I would also like to thank Mojow Autonomous Solutions INC. for the internship experience.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the last decade, computer vision via machine learning has been a rapidly progressed topic not only in academic research, but it is also widely employed in industry solutions.

A popular approach to improve the robustness of a visual cognition system includes multiple and multiple types of sensors, such as RGB cameras, stereo cameras, IR sensors, and LiDAR, depending on the requirement of the specific task. To integrate these sensors, a proper sensor fusion method is crucial. Deep learning and its variants can leverage the multiple hidden layers to automate the process of feature generation instead of manually handcrafting sophisticated algorithms for sensor fusion.

Regardless of the ease to deploy deep learning neural networks for solving single problems, generating a precise depth map, which is also a visual recognition task, has become a crucial step for various down-streamed visual recognition tasks, such as autonomous driving, 3D object detection, and Augmented reality.

These applications require an understanding of the spatial depth of surroundings to certain degrees. An effective method of acquiring precise dense depth maps is depth completion via RGB cameras and sparse LiDAR scans. Since such multi-sensor fusion can make use of the highly informative RGB imagery and direct discrete quantitative depth measured by the LiDAR sensor. Therefore, it has complementary advantages over implementation with sensors of a single type.

However, due to the sparsity of the LiDAR depth scans and their non-uniform distribution in a 2D viewpoint, blurriness and ambiguity around object edges are common issues. The sensors are also prone to noise and errors due to the environment. Thus, depth completion is a challenging task and has attracted high research interest in recent years. Many works have selected KITTI [2] as the benchmark dataset, in which 3D point cloud captured by Velodyne LiDAR HDL-64E is projected into an image plane that is synced with RGB cameras mounted on the same data collection platform.

After the success of convolutional neural networks in computer vision, deep convolutional neural networks have been a very popular choice for state of the art, such as the PE-Net [1]. Hourglass deep convolutional neural networks as the backbone for extracting multi-scale features and a refinement module to further process the coarse depth map have been a common approach. Whereas, it often consists of multiple layers and multiple branches with sophisticated connections. Such networks can be very heavy with more than a hundred Million parameters and a relatively slow inference time. Despite the prosperity of these methods, due to the nature of depth completion, in production, it is commonly deployed on embedded devices, where the hardware is constrained by both space and cost, furthermore, real-time inference is a fundamental requirement. Therefore, the demand for a compact and efficient while still precise depth completion implementation is needed to be addressed.

Inspired by PE-Net design, we propose a methodology of composing a Compact Depth-wise Separable Precise Network (CDSPN), which takes advantage of the highly efficient depth-wise separable technique to significantly reduce the parameters, selective fine-tuning via pruning to further compact the model without losing precision, black-boxed knowledge distillation for boosting the accuracy and model scaling that requires no knowledge of the teacher models.

Specifically, We have replaced convolutional layers including the encoder and decoder with highly efficient depth-wise separable convolution. During experiments,

we find the attention module, such as the squeeze-and-excitation in MBConv [3] has a limited utility. With wholly grouped convolution and point-wise convolution, the parameters needed and the inference time are reduced by a large margin. And up-sampling combined with convolutional operations replace the transpose convolution to remove some undesirable checkerboard artifacts.

After adopting the depth-wise separable convolution, we utilize random layer pruning as a stability test on our backbone model to selectively guide the fine-tuning of the model, furthermore preventing over-parameterization. As a result, after iterations of pruning, our redesigned backbone model can excel from a simple architecture with a high ratio of quality verse the number of parameters. Overall, our model trains on significantly fewer parameters than state of the art while outperforming existing models with Depth-wise separable convolution and other compact models at a nearly identical number of parameters, as well as fast enough to be ported on embedded devices.

To compensate for the precision loss from the architectural changes and further, endue the network with the ability of scaling, we provide a simple but effective knowledge distillation method that solely depends on the prediction of the teacher model which is treated as a black box. With distillation on the prediction and its gradient, our model is easy to scale for a higher accuracy requirement without complex tricks, fine-tuning of the network architecture or retraining of the teacher. It ensures that distillation is still possible when in real-world applications, only the API to the teacher is accessible.

1.1 Motivation

Although precision is utterly important in in-depth prediction, portability and speed are essential in real-world applications, where the performance is limited by the device hardware. Existing efforts focus on breaking the record in terms of accuracy, conversely, it has become larger and heavier to be deployed. Therefore, a bridge between

efficient implementation and existing high precision is seriously demanded, which has been the major motivation for this work.

Yet in nature, there is a trade-off between the performance and the resource utilized by the system, we aim to achieve as large an area as possible of this two axis. The goal being optimized is defined, such that at the same parameter size and inference time, our work presents an advantage in precision. When the precision is comparable with state of the art, our model leverages a smaller parameter footprint and a faster inference speed. The priority from high to low is parameter size, precision and inference speed.

Notably, the sparsity and nonuniform distribution of the LiDAR scans promote great difficulties at the edge of objects. While being further from the viewpoint, fewer signals are received by the LiDAR as well as a larger ratio of noise, making it even harder to predicting the depth. The challenge presents in the problem makes it an exciting research objective.

1.2 Contributions

The discussion mentioned above encourages us to propose a depth completion framework that focuses on the characteristics of light-weighted, efficient, and extendable. The main contributions of this work are as follows:

1. We propose a complete workflow of composing a compact model from the existing state of the art, creating flexible and efficient agents concerning the requirement of different scales for performing visual recognition tasks. Without laying tricks and sophisticated design, this methodology, at a large elasticity, can be applied to other and even future works of suitable.

2. We attract attention to depth-wise separable convolution as a promising approach to design compact models for model compression and constrained environment. In terms of depth completion tasks, it can be a helpful addition to

the literature as we observed attention might not be necessary for depth-wise separable convolution. By reducing the network fragmentation, the inference speed, which is one of the downsides of depth-wise separable convolutions, can be relieved.

3. Our models are capable of outperforming existing depth-wise separable implementation [4] in all metrics of accuracy, parameter count, and inference speed, and competitive root mean square error (RMSE) to state-of-the-art compact model [5] with [6] or without knowledge distillation at a similar number of parameters and inference speed. Our scaled model with the help of knowledge distillation can match state of the art results of the teacher PE-Net model by a saving of over 50 times parameters and faster inference speed.

1.3 Thesis Outline

In Chapter 2, we review existing works, their limitations, and other related efforts. In Chapter 3, we present a complete workflow to generate and fine-tune an efficient compact model from existing state of the art. In Chapter 4, we propose a simple knowledge distillation algorithm that treats the teacher as a black box and discuss how it can help to scale the model and boost its performance. Finally, in Chapter 5, we summarize and conclude this thesis and discuss possible future research directions.

Chapter 2

Background

2.1 General Introduction

Depth completion aims to up-sample sparse irregular depth (often from LiDAR scan or RGB-D cameras) to dense regular depth, which facilitates the downstream perception module. Commonly, not only sparse depth is employed, but additional sensor data as well, such as RGB images. Pure sparse depth suffers from the ambiguity around object edges, while further away from the viewpoint, the scarcer the measurement gets. In complementary, cameras can capture dense and high-resolution data. When multiple sensors work together, dense depth prediction can be more accurate. In recent years, deep learning based methods have shown overwhelming advantages in processing sensor fusion and depth enhancement.

In addition to depth completion, depth prediction is a similar task, in which only a single or multiple RGB cameras are exploited. There is also research on unsupervised depth estimation. But in this thesis, only techniques of depth completion are the emphasis. The research is drawn from computer vision and deep learning.

2.2 Related Works

2.2.1 Multi-scale Deep Convolution Neural Network

The depth completion task is a pixel-wise inference task similar to semantic segmentation. In these tasks, multi-scale networks are widely employed because these

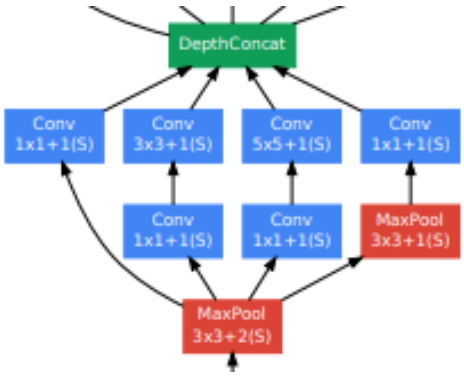

Figure 2.1: GoogLeNet [7] layers consist of convolutions with different kernel sizes.

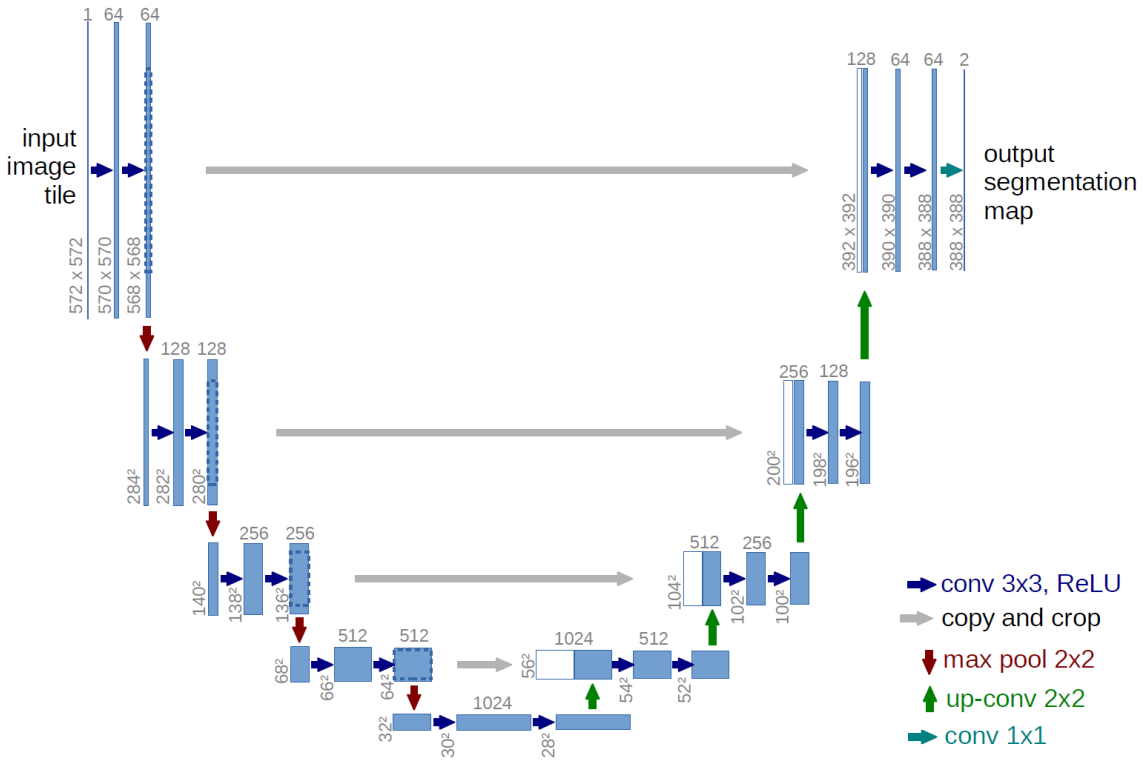

Figure 2.2: This figure shows UNet [7] architecture which is one of the most popular hourglass network designs.

architectures can extract and exploit features at different scales or spatial properties. Encoder-decoder architecture with skip connections, the Hourglass networks (shown in Fig.2.2), such as the U-Net [8] is one of the most popular backbones, which includes an encoder to extract multi-scale features sequentially, a transpose convolution decoder to reconstruct the prediction, and the residual connections aid the decoder

in collecting features.

Another approach to learning a coarse-to-fine representation is using convolution kernels of different sizes. An example layer in GoogLeNet [7] can be seen in Fig.2.1 By further composing an attention mechanism [9], the weights from different convolutions can be reweighed as the input to the next layer. However, the inference speed and training can be slow due to the fragmentation of convolutions.

2.2.2 Depth-wise Separable Convolution

Depth-wise separable convolution [10]was first proposed by Chollet from Google. It tries to represent a regular dense convolution by two separate convolutional operations. As shown in Fig.2.3, it includes one group convolution with a kernel on each channel of the input feature independently.

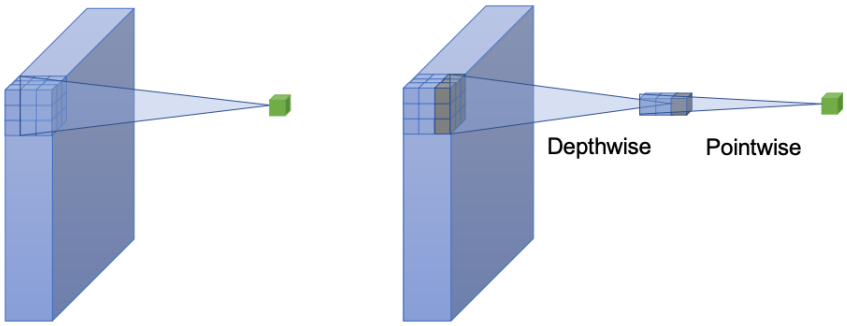

Figure 2.3: Standard convolution (left) and depth-wise separable convolution (right)

In this process, usually, the output channels are equal to the input channels while the spatial property of each channel may change. Then a point-wise (1x1 kernels) convolution follows, where the output feature is projected into a new channels space. Compared to dense convolution, depth-wise separable convolution has a significantly fewer total number of parameters. As a result, depth-wise separable convolution and its variants have been commonly utilized in many prior works [11][12][13][14], and depth completion [4] as well. However, some studies [15] also have shown that depth-wise separable convolution has a higher memory access consumption, leading to a

longer inference time than dense convolution at the same number of parameters. Our implementation balances accuracy and speed.

2.2.3 Multi-branch Depth Completion Networks

Many researches [16][1][17] in depth completion has adopted a multi-branch design for its network architecture, as it has a better capacity to exploit contextual features. PE-Net [1] is one of them that does not only have a simple architecture but also perform decently in the depth completion task. It employs a two-branch structure as shown in Fig.2.4, taking advantage of RGB data and spacial depth information respectively. The final depth prediction is calculated by the depth maps and their confidence values from each branch.

$$\hat{D}_f(u,v) = \frac{e^{C_{cd}(u,v)} \cdot \hat{D}_{cd}(u,v) + e^{C_{dd}(u,v)} \cdot \hat{D}_{dd}(u,v)}{e^{C_{cd}(u,v)} + e^{C_{dd}(u,v)}} \tag{2.1}$$

Where (u, v) denotes a pixel, C is the confidence map, and D is the depth prediction.

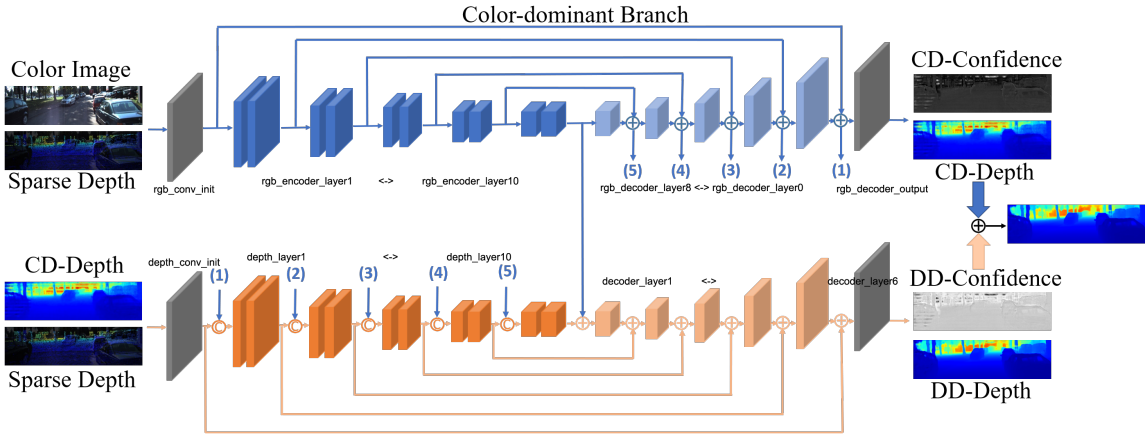

Figure 2.4: In the two-branch design of PE-Net, the first color-dominant branch takes in the color and sparse depth as inputs, and the second depth-dominant branch features the depth generated in the first branch and sparse depth as inputs. Furthermore, residual connections are placed between and within the two branches.

Geometric properties are fused into the convolutional layers to enhance the capability of the model. In each layer, three filters are concatenated which follow the formula below.

$$Z = D \tag{2.2}$$

$$X = \frac{(u - u_0)Z}{f_x} \tag{2.3}$$

$$Y = \frac{(v - v_0)Z}{f_y} \tag{2.4}$$

The u_0, v_0, f_x, and f_y are intrinsic parameters of the camera. The sparse depth map D is min-pooled to obtain Z at smaller scales, u and v are x and y coordinates respectively.

2.2.4 Convolutional Spatial Propagation Networks

The spatial propagation network (SPN) is first proposed by Liu et al. [18] to learn local affinities, a weighted graph, in which each pixel is a node, and an edge between each pair of pixels determines how close, or similar, two points are in latent space. However, it propagates in an inefficient column-wise and row-wise manner.

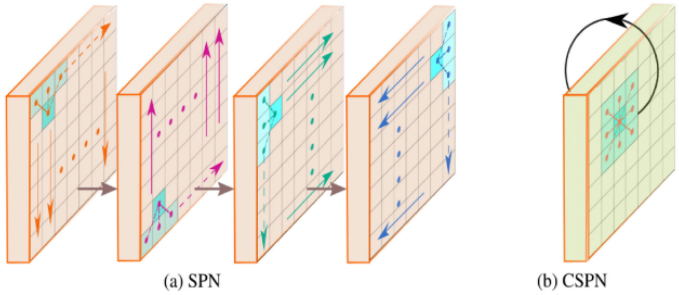

Figure 2.5: Comparison between the propagation process in (a) SPN, and (b) CPSN. SPN scans the whole image in four directions sequentially, while CSPN propagates a local area towards all directions simultaneously at each step in a recurrent manner.

Cheng et al. [19] thereby propose a convolutional spatial propagation network (CSPN) for efficiency and meanwhile apply it to refine depth completion results. Later, Cheng et al. [20] further improve the effectiveness and efficiency by learning adaptive convolutional kernel sizes and the number of iterations for the propagation, thus the context and computational resource needed at each pixel could be dynamically assigned upon requests.

In PE-Net, the involvement of the dilated accelerated convolutional spatial propagation networks (DA-CSPN++) acts as a refinement module to perform operations in

2.5, further recovering structural details. The coarse depth map is defined as D^0 and for pixel i, at each iteration t, it aggregates information propagated from neighbor pixels $N(i)$. A_{ji} denotes the affinity between pixel i and pixel j.

$$D_i^{t+1} = A_{ii} D_i^0 + \sum_{j \in N(i)} A_{ji} D_j^t \tag{2.5}$$

Two modifications to make it more effective and efficient. First, the convolution is dilated [21] to enlarge the propagation neighborhoods, aggregating a larger receptive field without losing coverage.

Then, by using one-hot convolutional kernels, it makes propagation from each neighbor truly parallel, which greatly accelerates the propagation procedure compared to the original pixel-wise spatial propagation. In our work, we choose to follow the design of DA-CSPN++ and keep this refine module untouched.

2.2.5 Model Pruning

Pruning is one model compression technique that allows the fine-tuned model to become smaller and more efficient. Model pruning involves two distinct paths, weight pruning, and layer pruning.

Weight pruning [22] aims to increase model sparsity by eliminating the weights with low magnitude, such that both the original and pruned model has the same architecture, but the pruned model becomes sparser where the weights with the low magnitude are set to zeros.

Weight pruning can be applied during training, such that at every t step, a binary mask of setting all the selected weights to zeros, then continues with the training. With an identical memory footprint, large, but pruned models (large-sparse) can achieve better accuracy than their smaller, but dense (small-dense) counterparts.

The second way is filter pruning [23] and layer pruning. It directly eliminates the filters and layers in the model instead of the weights. Compared with weight pruning, which sometimes can be tricky to benefit from memory saving and speed improvement

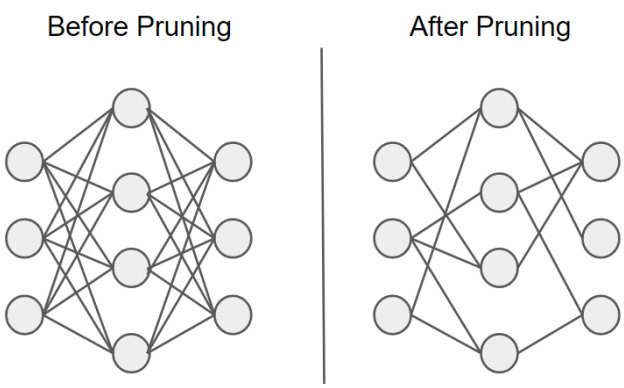

Figure 2.6: Illustration of weight pruning, the connection with the low magnitude is removed, resulting in a sparser network.

when deploying onto edge devices due to irregular network connections, filter pruning and layer pruning can be much easier to apply. Furthermore, because the entire filter or layer is removed, the resulting model can be easily exported for production. In each step t, one of the layers or filters that minimize the increase in error is pruned. Or instead of calculating the error of each layer being removed, Li et al. [24] state that removing the filter with the 'smallest' absolute kernel weights, defined by 2.6,

$$s_j = \sum_{l=1}^{n_i} \sum |K_l| \tag{2.6}$$

where n_i is the input channel number, K_l is the kernel of the filter l, i and j are the input and output filters respectively,

is better than 'random' or 'largest' when the pruned parameter counts are the same. It allows filter pruning without ground truth or when error calculation is expensive or not possible.

2.2.6 Knowledge Distillation

Knowledge distillation was first proposed [25] to compress the model without significant loss in performance. It is the process of transferring knowledge from a larger and more accurate teacher network to a smaller student network. So that the student model can have a better convergence with no overhead during inference. By learning

the patterns of the teacher network, the student can avoid some learning pitfalls to have better performance than training with only the ground truth. It has been a popular approach to improving the performance of compact models without adding additional weights and costs.

Most common distillation methods can be classified into two categories to help the student model mimic the teacher.

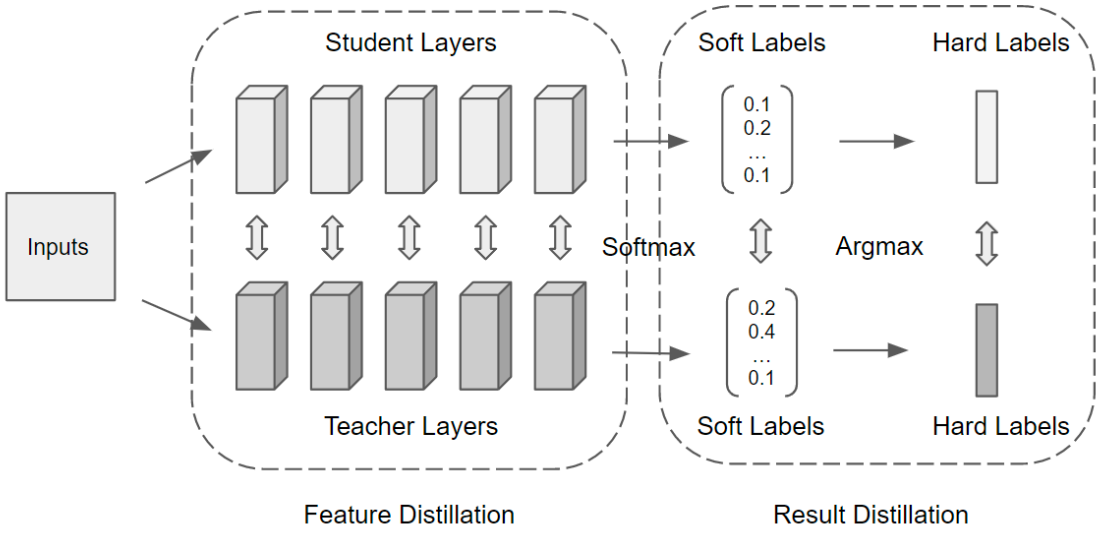

Figure 2.7: Different stages to apply distillation in a classification neural network.

Naturally, the first direction is to train the student to have similar outputs as the teacher. It can further be categorized into hard labels and soft labels [26]. Hard labels simply compare the predictions, whereas soft labels compare the output distribution or relations. In classification problems, the soft labels are usually generated from the softmax function. Using soft targets can provide more information, such as the probability, thus having better generalization and less variance in gradients between training examples.

The second direction is about feature similarity between the teacher and the student. The goal is to train the student model to learn the same feature activation as the teacher model. It requires the teacher and the student to have a similar layer

structure so that the pair-wise comparison can be formed.

Regarding the depth completion task, Wang et al. [6] propose a self-paced knowledge distillation method. They propose learning with the easy-to-hard curriculum by gradually introducing hard pixels to distill the depth and structure (gradient) knowledge from the teacher to the student. Not only the final teacher model is required, but in order to mine the hard pixels, snapshots of the teacher model in intermediate steps are also taken as well. The hardness R is calculated by 2.7, where the intermediate and final teachers predicted dense depth maps are D_{ti} and D_t respectively.

$$R = \frac{|D_{ti} - D_t|}{D_t + \epsilon} \tag{2.7}$$

When compared to hyper-parameter γ, a pixel-wise mask is generated to leave out the hard pixels in distillation loss calculation.

And Liu et al. [27] exhibit a knowledge distillation method using multiple teachers so that the student can avoid learning the error modes of a particular teacher. The author state that different error modes can be observed in different teachers at different depth ranges. Hence, the distillation loss is constructed by adaptive selecting predictions from the teacher that best minimize the error. For regions where all of the teachers yield high errors, a default loss is chosen instead of the distillation loss.

2.3 Gap in Research

As discussed in motivation, prior efforts are concentrated on exploiting the properties of the contextual information from the input data or network architecture to squeeze more precision. However, efficient models are more practically demanded in the field. In this direction, existing works are heavily on variants of knowledge distillation. These distillation methods either require modification on the models to form feature connections or training one or more teacher models together with the student model. While in the real-world application, these implementations can be sophisticated, and sometimes the teacher model is accessible only through API. There is a gap in how

we can transform the highly accurate state of the art into efficient models that can be deployed in the real world. Knowledge distillation can greatly boost the performance of the model while we believe a simple distillation that only requires the teacher as a black box can be helpful.

Chapter 3

Workflow To Create CDSPN

In this chapter, we first discuss the architecture overview, then introduce the process of creating the compact model, and finally the related experiments.

3.1 Overview

Our model employs a two-branch network developed from PE-Net, which was refined by a stability test via random layer pruning. The stability test was deployed to improve the cost-efficiency of trainable parameters of our model in addition to controlling the total parameter count. We utilize lightweight depth-wise separable convolution for replacing regular convolutions in our proposed model. Unlike many current works, no attention module, e.g. squeeze and excitation attention, was added to have a better inference speed. In the decoder, instead of transpose convolutions [28], we choose bilinear upsampling to avoid checkerboard artifacts. Additionally, the geometric encoding in PE-Net is excluded for simplicity.

3.1.1 Model Architecture

Same as PE-net, our model is also constructed with the two-branch hourglass backbone and the DA-CSPN++ refinement module.

In the backbone, there are residual connections inside the layer, between the encoder and decoder in each branch and between the encoder of the color-dominant

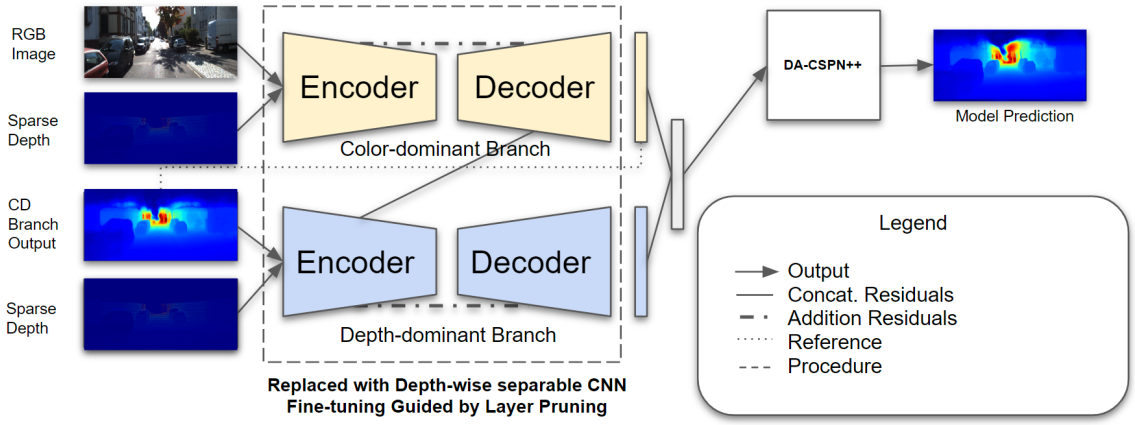

Figure 3.1: Overview of the proposed model. Our network consists of two hourglass branches denoted as the color-dominant branch and the depth-dominant branch, in which there are residual connections. A refinement module DA-CSPN++ [1] further processes the depth predicted by the backbone.

branch and the decoder of the depth-dominant branch. Because of the layer pruning, the residuals between layers are moved or excluded when needed. The entire model is built upon the same layer design for both the encoder and decoder, which is shown in Fig.3.2. A global scaling factor m allows custom scaling with ease, which is discussed in the next chapter.

Each branch in the backbone can output a dense depth map and a confidence map. The intermediate dense depth map is calculated with the two depth maps and confidence maps from each map, which is shown in 3.1

$$\hat{D}_f(u,v) = \frac{e^{C_{cd}(u,v)} \cdot \hat{D}_{cd}(u,v) + e^{C_{dd}(u,v)} \cdot \hat{D}_{dd}(u,v)}{e^{C_{cd}(u,v)} + e^{C_{dd}(u,v)}} \tag{3.1}$$

Where (u, v) denotes a pixel, C is the confidence map, and D is the depth prediction.

When we train the model, the backbone is first trained for about 15 epochs then its weights are frozen, and then train the refinement module for another 3-5 epochs. Finally, the entire model is trained to convergence.

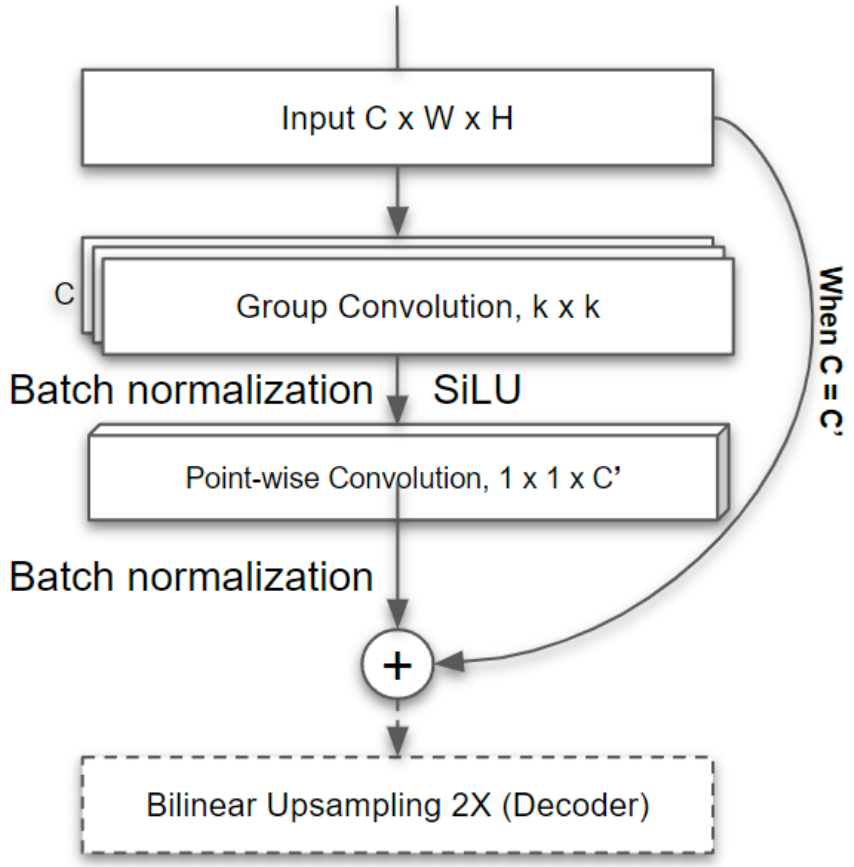

Figure 3.2: Layer composition for the encoder and decoder. Bilinear upsampling only exists in the decoder part of the model. Scaling is treated as a multiplier to the channel number C and C'.

3.2 Methodology

3.2.1 Base Model Selection

One of our aims is to create a top-down pipeline to convert the existing works, which is high precision and has a large number of parameters, to accurate but efficient compact solutions. We first explore the leaderboard and find a few candidates. Among them, we have chosen the PE-Net because it has two very deep branches with a relatively simple layer design. It is a good fit for deploying our model compression techniques. Our work is around two main objectives, a small count of parameters and the accuracy should be comparable with other state of the art which are designed to be compact from scratch.

3.2.2 Efficient Layer Design

From chapter 2 we have discussed that depth-wise separable convolution has drastically fewer learnable parameters than a regular convolution with the same input size and output size. Inevitably, depth-wise separable convolution will sacrifice some accuracy, but under the same number of parameters, it still has better cost-efficiency.

The base model makes use of both convolution and transpose convolution, we have replaced the convolution in the encoder with depth-wise separable convolution, then we choose to use the same depth-wise separable convolution and bilinear upsampling cooperatively, because it can not only remove the undesired checkerboard pattern, which is due to the uneven overlap when the kernel size is not divisible by the stride, but also has no trainable parameters. It makes our basic block unified, see Fig.3.2, in the architecture. And then we can continue decorating the layers globally.

3.2.3 Random Layer Pruning Stability Test

Convolution with large input and output channel sizes creates a huge parameter footprint even though we can use depth-wise separable convolution. Our proposed random layer pruning stability test is used to fine-tune the network. The stability test is done by randomly pruning filters from the target layer. One layer at a time, after the target layer is pruned, we reevaluate the new model. With a proportion of layer filters disabled, the model usually suffers a drop in accuracy. If the layer is important, that it has a large influence on prediction, we can see a large drop in accuracy. Otherwise, a very small difference from the full model. Sometimes, pruning can even increase the accuracy, which means the model is over-parameterized. To take advantage of this observation, we can purge the over-parameterized layers, reduce the number of filters in insignificant layers and carefully increase the number of filters in important layers.

Detailed implementation can be viewed in Alg.1, where ϵ is chosen respectively to the base accuracy. An example is shown in Tab.3.1.

More importantly, with careful pruning [29], the production model can be even smaller, despite a little reduction in accuracy.

Algorithm 1 The workflow of compression via pruning

Require: model N
  $Acc \leftarrow test(N)$
  while $countParams(N) \gtrapprox target$ **do**
    Train the model N
    for all layer $M \in N$ **do**
      $N^{'} \leftarrow N$
      $M^{'} \leftarrow M$
      Randomly prune $\gamma\%$ filters in $M^{'}$
      $Acc_{new} \leftarrow test(N^{'})$
      if $Acc_{new} + \epsilon_1 > Acc$ **then**
        M will be pruned
      else if $Acc_{new} + \epsilon_2 > Acc$ **then**
        M will have less number of filters
      else if $Acc_{new} + \epsilon_3 < Acc$ **then**
        M will have more number of filters
      end if
    end for
    Apply changes to N
    Maintain residual connections in N
  end while

After iterations of fine-tuning cycles, the resulting model, shown in Tab.3.2, is no longer symmetrical in the encoder and decoder, because we find the decoder needs fewer layers and filters than the encoder.

Color Dominant Branch				
Layer	Ker., Str.	Channel I/O	Out Shape	Input
rgb_init	5, 1	4/32	H xW	rgb, d
rgb_e1	3, 2	32/32	1/2 H x 1/2 W	rgb_init
rgb_e2	3, 1			rgb_e1
rgb_e2_2	3, 1			rgb_e2
rgb_e3	3, 2	32/64	1/4 H x 1/4 W	rgb_e2_2
rgb_e4	3, 1	64/64		rgb_e3

rgb_e4_2	3, 1			rgb_e4
rgb_e5	3, 2	64/128	1/8 H x 1/8 W	rgb_e4_2
rgb_e6	3, 1	128/128		rgb_e5
rgb_e7	3, 2	128/256	1/16 H x 1/16 W	rgb_e6
rgb_e8	3, 1	256/256		rgb_e7
rgb_d1	3, 1	256/128	1/8 H x 1/8 W	rgb_e8
rgb_d2	3, 1	128/64	1/4 H x 1/4 W	rgb_d1 + rgb_e6
rgb_d3	3, 1	64/32	1/2 H x 1/2 W	rgb_d2 + rgb_e4_2
rgb_d4	3, 1	32/32	H x W	rgb_d3 + rgb_e2_2
rgb_d5	3, 1	32/2		rgb_d4 + rgb_init
Depth Dominant Branch				
Layer	Ker., Str.	Channel I/O	Out Shape	Input
depth_init	5, 1	2/32	H x W	d, rgb_d5[0]
depth_e1	3, 2	32/32	1/2 H x 1/2 W	depth_init
depth_e2	3, 1			depth_e1
depth_e2_2	3, 1			depth_e2
depth_e3	3, 2	64/64	1/4 H x 1/4 W	rgb_d3 + rgb_e2_2, depth_e2_2
depth_e4	3, 1			depth_e3
depth_e5	3, 2	128/128	1/8 H x 1/8 W	rgb_d2 + rgb_e4_2, depth_e4
depth_e6	3, 1	128/128	1/8 H x 1/8 W	depth_e5
depth_d1	3, 1	256/64	1/4 H x 1/4 W	rgb_d1 + rgb_e6, depth_e6
depth_d2	3, 1	64/32	1/2 H x 1/2 W	depth_e4 + depth_d1
depth_d3	3, 1	32/32	H x W	depth_e2_2 + depth_d2
depth_d4	3, 1	32/2		depth_d3

Table 3.2: The layer components of the proposed model. K. is the kernel size and Str. is the stride of the group convolution. H and W are the height and width of the input features, Ch.I/O are the channel numbers of the input and output features in the layer. Input consists of the source of the input features, where a comma means concatenation and a plus means addition.

3.2.4 Apply to Other Architectures

Our approach to create a compact network is model-agnostic. Although, we have only demonstrated on PE-Net due to time consumption in training, in this section, we will briefly introduce how to apply the same techniques on another network architecture. SemAttNet [16] is one of the top accuracy models on the leaderboard Its backbone consists of three branches, which is one more branch than the PE-Net. Likewise, with the SAMMAFB block unchanged, the same depth-wise separable convolution unit can replace the Resblocks and deconvolutions in these branches. The number of channels and the filters counts can be guided by the same pruning test that has been shown on PE-Net. By this approach, a compact network from SemAttNet are available for deploying on embedded device.

3.3 Experiments

In this section, we conduct experiments and ablation studies on our model.

3.3.1 Dataset

Our methods are evaluated on the KITTI depth completion dataset [2], a popular benchmark dataset for outdoor depth completion tasks. It provides RGB images and aligned sparse depth maps. The ground truths are semi-dense depth maps that are generated by 11 laser scans to increase the density of the generated depth maps. A consistency check can remove noise and outliers, which makes use of stereo camera pairs to compare the scanned depth to results from a semi-global matching stereo

Layers	RMSE [mm]	Random Pruning	= 75% filters
Baseline	886		
rgb_init	9843	depth_init	4905
rgb_e1	15706	depth_e1	6685
rgb_e2	3913	depth_e2	3693
rgb_e3	5598	depth_e3	1966
rgb_e4	1125	depth_e4	1132
rgb_e5	1236	depth_e5	1046
rgb_e6	948	depth_e6	1031
rgb_e7	1076	depth_e7	926
rgb_e8	999	depth_e8	907
rgb_e9	938	depth_e9	899
rgb_e10	919	depth_e10	889
rgb_d1	915	depth_d1	885
rgb_d2	952	depth_d2	890
rgb_d3	1040	depth_d3	1351
rgb_d4	1977	depth_d4	921
rgb_d5	5418	depth_d5	4055

Table 3.1: An example of the pruning test on each layer, 75% of the filters are pruned separately and then evaluated for accuracy. In this example, because there are too many filters near the bottleneck layers e10 and d1, even though most of the filters are pruned, the accuracy is very close to the baseline of which the parameter number is several times more.

reconstruction approach (SGM) [30]. Therefore, the ground truths are reliable to be used for comparing the results.

The dataset contains 85895 samples for training, 1000 samples as the selected validation, and 1000 test samples without ground truth for benchmarks.

The data collection platform is a Volkswagen Passat B6 with the following sensors:

1. 1 inertial navigation system (GPS/IMU): OXTS RT 3003

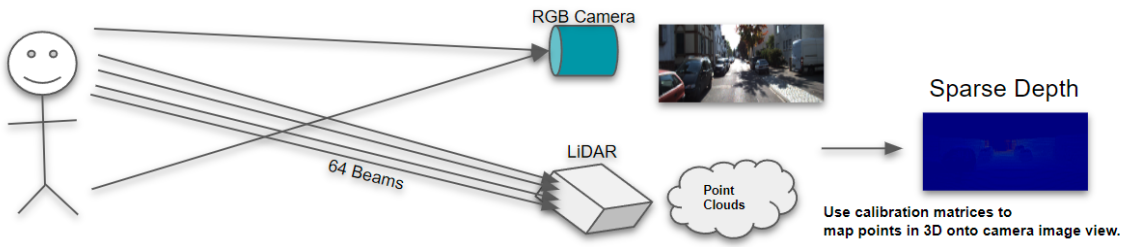

Figure 3.3: How the sensors capture data, from rays to image. The camera and the LiDAR are synchronized, such that the camera is triggered when the rotating LiDAR is facing forward.

2. 1 laserscanner: Velodyne HDL-64E

3. 2 grayscale cameras, 1.4 megapixels: Point Grey Flea 2 (FL2-14S3M-C)

4. 2 color cameras, 1.4 megapixels: Point Grey Flea 2 (FL2-14S3C-C)

5. 4 varifocal lenses, 4-8 mm: Edmund Optics NT59-917

The LiDAR scans at 10 frames per second, 64 beams, 0.09 degree angular resolution, 20 mm distance accuracy, 120 m maximum range with field of view of 360◦ horizontal and 26.8◦ vertical. The camera images are cropped to a size of 1382 x 512 pixels, capturing synchronously at 10 frames per second, when the LiDAR is facing forward. The configuration is shown in Fig.3.4.

As a common practice in KITTI benchmarks, we select four metrics for evaluation: the root mean square error (RMSE, [mm]), mean square error (MAE, [mm]), inverse depth iMAE [1/km] and iRMSE [1/km]. Whereas, we report our results on the test dataset with no ground truth open to us, most experiments such as ablation studies are carried out on the selected validation dataset. The reason is that we have limited submission for benchmarks, as requested by the dataset provider, such studies should be on the validation dataset.

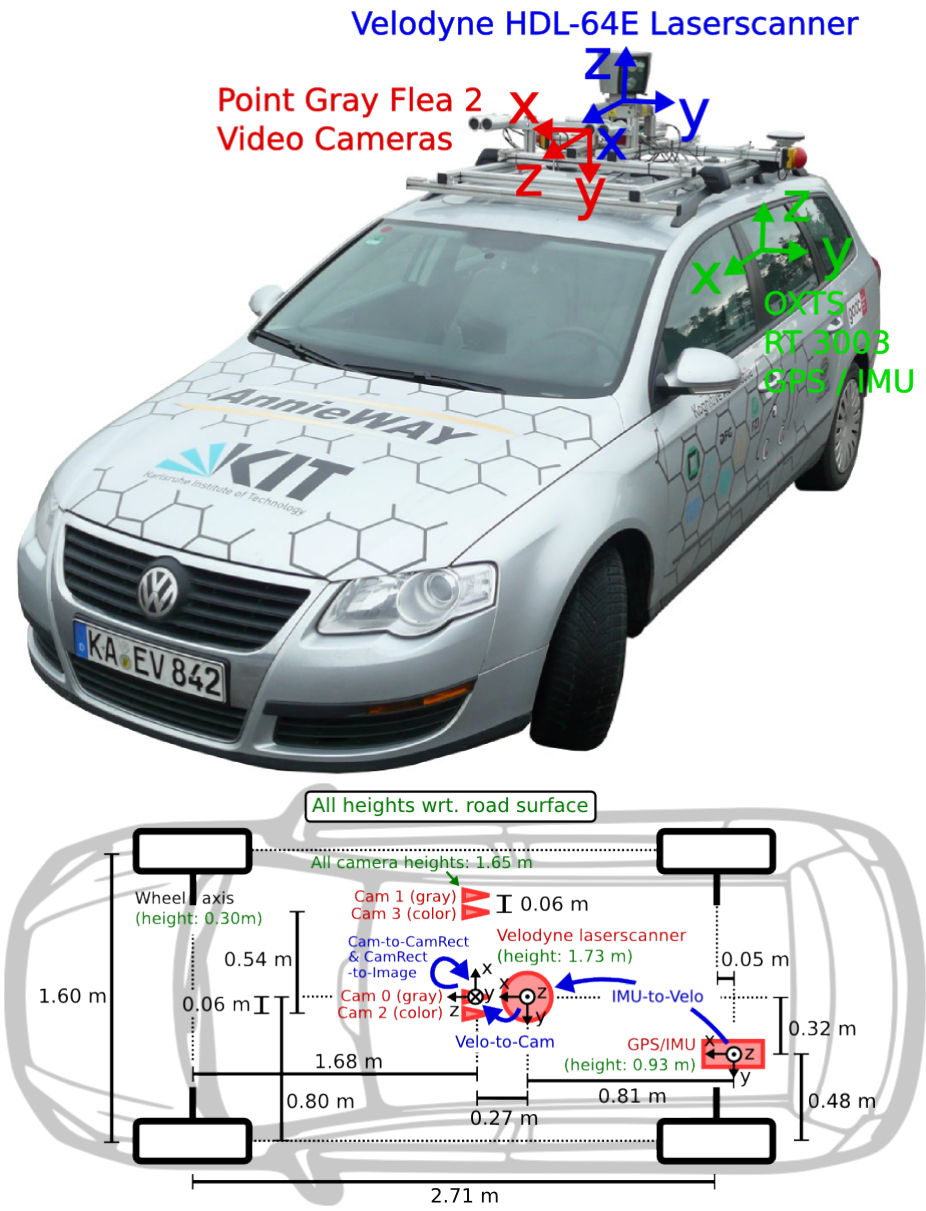

Figure 3.4: KITTI recording platform setup.

3.3.2 Experiment Setups

Most of our training hyper-parameters are consistent with PE-Net, as tweaking these hyper-parameters is not the main focus of this work. We also use the same multi-staged training strategy by first training the backbone model, then it is frozen while the DA-CSPN++ module is trained, and finally training the entire network.

3.3.3 Results

In this section, our model series are tested on the KITTI dataset among other state of the art. Our models have been one of the best compact models in this range, especially in KITTI's primary metric 'RMSE'. The test results are available at Tab.3.3.

Datasets	Test						
Metrics	RMSE	MAE	iRMSE	iMAE	#Params	Runtime1	Runtime2
E-Net [1]	741.30	216.26	2.14	0.95	131.8M	0.02	0.010
PE-Net [1]	730.08	210.55	2.17	0.94	131.9M	0.03	0.018
DeepLiDAR [31]	758.38	226.50	1.15	2.56	144.0M	0.07	—
Sparse-to-Dense [32]	814.73	249.95	2.80	1.21	26.1M	0.08	—
DepthNet [4]	991.88	261.67	2.99	1.09	1.0M	0.09	—
NConv-CNN [33]	829.98	233.26	2.60	1.03	0.4M	0.02	—
MSGCHN32 [5]	783.49	226.91	2.35	1.01	0.4M	0.01	0.013
Ours x32	779.36	227.71	2.63	1.07	0.4M	—	0.014

Table 3.3: Comparison with state of the art on KITTI benchmark. RMSE and MAE are measured in mm. IRMSE and iMAE are measured in 1/km. Runtime is measured in seconds. Runtime1 is published on the benchmark site. Runtime2 is evaluated on the same hardware. Our base models can achieve competitive results compared with the state of the art. It can also outperform the current best compact model around the same number of parameters in terms of 'RMSE'.

3.3.4 Ablation Studies

Use of Attention Module

Attention is commonly used in Inverted Residual Blocks (MBConv [34]) and many depth completion networks [35][36] to improve performance. The attention module is employed in the dynamic weight adjustment process based on the features of the input image.

Initially, we experiment with the standard MBConv [3] blocks, but then the squeeze and excitation attention has been proven not beneficial, because of no improvement

in precision with 40% more parameters. Hence, our base block only includes depth-wise convolution and point-wise projection to compensate for the latency and further decrease the number of parameters.

In the experiments, to analyze the effect of the attention module, we test on classic squeeze and excitation [37] and coordinate attention [38]. 'SE' refers to squeeze and excitation and 'CA' refers to coordinate attention. Squeeze and excitation in the inverted residual block dynamically allocate weights to channels of the feature. It transforms the feature tensor to a single 2D feature vector via 2D global pooling and then performs point-wise convolutions followed by a sigmoid function, and finally multiplies the original feature tensor. Coordinate attention further embeds positional information by aggregating features along the two spatial directions X and Y.

Our studies in Tab.3.4 show although attention may improve the MAE, it is significantly slower because of too many small operators causing a reduction of parallelism [15] and requires more parameters in general.

Datasets	selected	validation		
Metrics	RMSE	MAE	#Params	Runtime
base	882	315	2.54M	0.0114
base+SE	915	305	3.49M	0.0167
Base+CA	900	267	2.74M	0.0259

Table 3.4: Comparison of an early backbone model with and without attention on KITTI selected validation dataset. RMSE and MAE are measured in mm. Runtime is measured in seconds. 'SE' refers to squeeze and excitation and 'CA' refers to coordinate attention.

Chapter 4

Knowledge Distillation and Scaling

In this chapter, we present techniques to make our model extendable and flexible.

4.1 Model Scaling

Model scaling aims to scale up a base network model to endow it with more representational power at the cost of greater computational complexity. Model scaling approaches typically focus on maximizing accuracy versus resource demands. Compact models are often sensitive to scales, for example, MobileNets [39] provides two hyperparameters, width multiplier α, and resolution multiplier ρ. Both multipliers control the size of the feature map, the width multiplier thins a network uniformly at each layer, and the resolution multiplier ensures the input image and the internal representation of every layer is subsequently reduced by the same multiplier.

Similar to the width multiplier, in our model, a global scaling factor is provided, which simply acts as a multiplier for the number of filters in each layer. However, unlike MobileNets compressing the model with the hyperparameters, our scaling factor is mainly for expanding the model. Because our network has been fine-tuned to the desired size during the pruning test, and when requiring better precision, the global scaling factor helps produce a larger model. As shown in Fig.4.1, the parameter count is increasing in a quadratic manner.

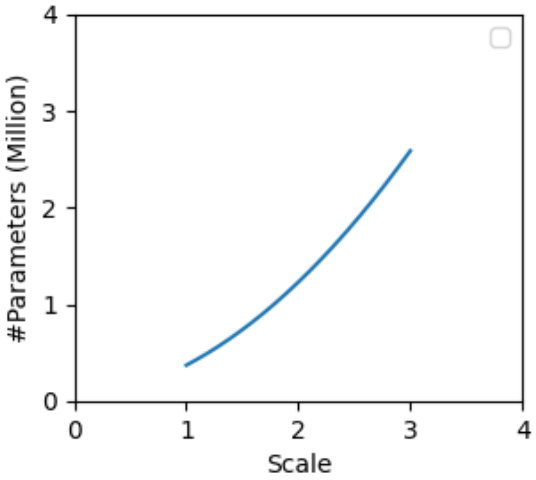

Figure 4.1: Model parameter counts verse the scaling factor.

4.2 Black-box Knowledge Distillation

One major challenge of training a lite or sub-optimal model is being stuck at some local maximum. To tackle the problem without introducing more parameters and overhead in production, we distill knowledge from a pretrained E-Net and PE-Net. Unlike others [6][40], only the prediction part of the models is involved in the distillation as part of the loss function. In other words, our knowledge distillation trains the student to have similar outputs as the teacher. Since it does not compare the intermediate features, it is friendly to deploy when the teacher model is a black box or only accessible via API. During experiments, we find two loss terms are helpful for reducing the error of the student model.

These distillation losses are formulated as:

$$L_{dd} = MSE_{(D_t, D_s)}. \tag{4.1}$$

$$L_{gd} = MSE_{(g(D_t), g(D_s)}. \tag{4.2}$$

MSE refers to Mean squared error, D_t is the predicted depth map generated by the teacher model, D_s is the predicted depth map generated by the student model. $g(x)$ calculates the gradient of x.

Together our loss function is:

$$L_D = \lambda_1 L_{dd} + \lambda_2 L_{gd}. \tag{4.3}$$

$$L_{GT} = MSE_{(m(D_g),m(D_s))} \tag{4.4}$$

$$L = (1 - \lambda_1 - \lambda_2) L_{GT} + L_D. \tag{4.5}$$

D_g is the groundtruth depth map, $m(x)$ is binary mask where $D_g^{i,j}$ is not 0. λ_1, λ_2 are hyperparameters, where $\lambda_1 = 0.1$, $\lambda_2 = 0.1$

Knowledge distillation can be added in each training stage or as the fourth stage of training after the base model converges. No conclusive gap can be observed in the final performance when both models converge. However, training with knowledge distillation can be slower and require a larger memory, unless caching the predictions from the teacher model. Thus, it is recommended to apply knowledge distillation as a booster. In summary, after adding the distillation, our entire workflow is represented in Fig.4.2.

Studies [41] also show that a stronger teacher model tends to have a better distillation effect. Our approach ensures one can easily find a stronger model and achieve even higher performance without retraining or modifying the teacher model. This simple approach dramatically improves the performance of the student model.

In our experiment, we also find that a larger or less fine-tuned model benefits more from distillation. Although a scaled model has better precision when compared with the base model, the cost-efficiency is actually lower being a sub-optimal model. Our hypothesis is smaller models have too few parameters for representation and well-tuned models have already learned more optimal weights. With a combination of the simple channel multiplier and knowledge distillation, the resulting scaled models can perform well at different parameter sizes.

4.3 Experiments

4.3.1 Knowledge Distillation Loss

Our knowledge distillation consists of two loss terms. To validate the effect of each loss, we conduct ablation studies on the cropped validation dataset, the results are in Tab.4.1 and Fig.4.3. Instead of training the backbone and the entire network separately, the training is done in one stage to better reflect the effect of distillation. Depth distillation only distills from the teacher's depth prediction, while gradient distillation takes the gradient of the teacher's depth prediction. When both distillations are employed, each of them weighs half of the total distillation loss.

Datasets	selected	validation
Metrics	RMSE	MAE
base (no distillation)	829	231
Depth Distillation	813	241
Gradient Distillation	805	239
Both	806	243

Table 4.1: Comparison with different distillation methods. RMSE and MAE are measured in mm.

From the experiment, as both of the distillation losses help reduce the primary metric 'RMSE', the structure gradient loss is superior to direct distillation with the teacher's depth prediction. However, there is a trend of overestimating the depth, predicting the depth further than the ground-truth, when using only the gradient loss, in real work applications like autonomous driving, it is often not desired. Although involving depth distillation will transfer some of the behaviors from the teacher, such as checkerboard patterns when using PE-Net as the teacher, a mixture of the depth distillation loss helps fix the above problem while still having better accuracy. Therefore, we choose to use the combination of both losses in our later experiments.

For all of the distillation methods, an increase of MAE can be seen, this is possibly caused by the shift of the local minimal where the model converges when involving the distillation losses in addition to the base L2 loss.

4.3.2 Knowledge Distillation in Scaling

Our distillation method is a helpful piece when scaling the model, we compare different scaled models with or without distillation to different scaled PE-Net. In this test, the scaled PE-Net applies the same global channel scaling, such that each layer has the input and output channels scaled by a certain amount. The experiment is conducted on the cropped validation dataset, and results are in Tab.4.2 and Fig.4.4. Our distillation method greatly improves the performance of larger models, hence we can achieve a similar precision in 'RMSE' with 50 times fewer parameters and faster speed, allowing developers to scale a compact light model flexibly without fine-tuning.

Datasets	selected	validation		
Metrics	RMSE	MAE	#Params	Runtime
PE-Net	757	209	131.9M	0.018
PE-Net (1/2 channels)	780	214	33.1M	0.018
PE-Net (1/4 channels)	807	223	8.4M	0.020
Ours x32	808	228	0.4M	0.014
Ours x32 (distillation)	806	243	0.4M	0.014
Ours x64	788	221	1.2M	0.014
Ours x64 (distillation)	770	226	1.2M	0.014
Ours x96	777	214	2.6M	0.014
Ours x96 (distillation)	760	223	2.6M	0.014

Table 4.2: Comparison with different scales of models on the KITTI selected validation dataset. RMSE and MAE are measured in mm. Runtime is measured in seconds. Our larger-scaled models are initially worse than PE-Net with larger scales. But after knowledge distillation, the improvement is higher than PE-Net with larger scales.

However, the distillation is not as effective on the fine-tuned compact model. Our hypothesis is when the model is already very compact, converging very close to the global minimal at its parameter count, then adding a teacher does not help guild the network to better convergence.

4.3.3 Summary

In addition to local comparison carried out on the selected validation dataset, we have submitted our series of models to KITTI for test set benchmarking. When we compare our results with the state of the art in Tab.4.3, our base model can outperform other compact models in the same range of parameters, with or without distillation. Even compared with the much larger models, ours are still competitive in accuracy and speed.

Datasets	Selected Validation				Test						
Metrics	RMSE	MAE	iRMSE	iMAE	RMSE	MAE	iRMSE	iMAE	#Params	Runtime1	Runtime2
E-Net [1]	772.77	215.46	2.19	0.94	741.30	216.26	2.14	0.95	131.8M	0.02	0.010
PE-Net [1]	757.20	209.00	2.22	0.92	730.08	210.55	2.17	0.94	131.9M	0.03	0.018
DeepLiDAR [31]	—	—	—	—	758.38	226.50	1.15	2.56	144.0M	0.07	—
Sparse-to-Dense [32]	—	—	—	—	814.73	249.95	2.80	1.21	26.1M	0.08	—
DepthNet [4]	—	—	—	—	991.88	261.67	2.99	1.09	1.0M	0.09	—
NConv-CNN [33]	—	—	—	—	829.98	233.26	2.60	1.03	0.4M	0.02	—
MSGCHN32 [5]	817.08	224.83	2.48	0.99	783.49	226.91	2.35	1.01	0.4M	0.01	0.013
MSGCHN64 [5]	—	—	—	—	762.19	220.41	2.30	0.98	1.2M	0.01	—
SKPD_MSGCHN64 [6]	—	—	—	—	750.90	212.66	2.16	0.92	1.2M	0.04	—
Ours x32	807.71	227.55	2.57	1.05	779.36	227.71	2.63	1.07	0.4M	—	0.014
Ours x64	785.88	217.76	2.38	0.98	756.13	218.95	2.31	0.99	1.2M	—	0.014
Ours x64 (with distillation)	769.80	225.58	2.57	1.09	746.37	226.24	2.53	1.10	1.2M	—	0.014

Table 4.3: Comparison with state of The art on KITTI benchmark. RMSE and MAE are measured in mm. IRMSE and iMAE are measured in 1/km. Runtime is measured in seconds. Runtime1 is published on the benchmark site. Runtime2 is evaluated on the same hardware.

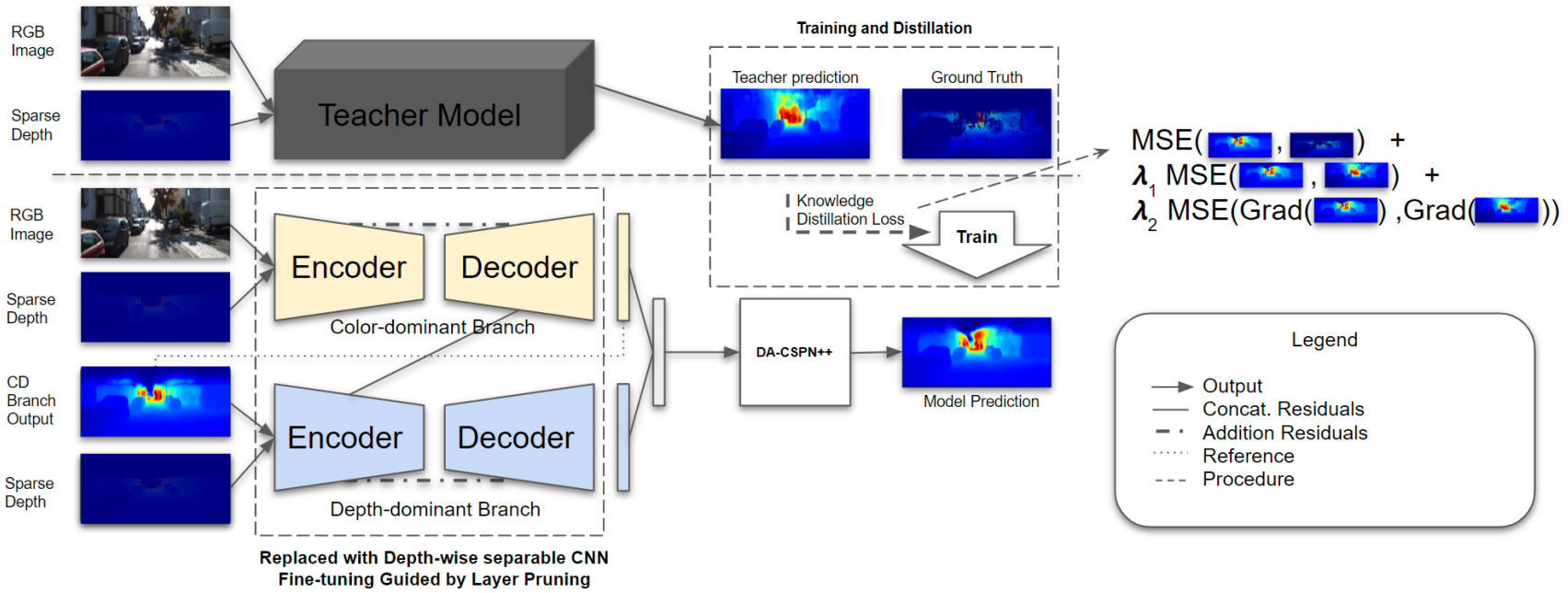

Figure 4.2: Overview of the proposed methodology with knowledge distillation. Offline knowledge distillation can be applied when the teacher model is treated as a black box.

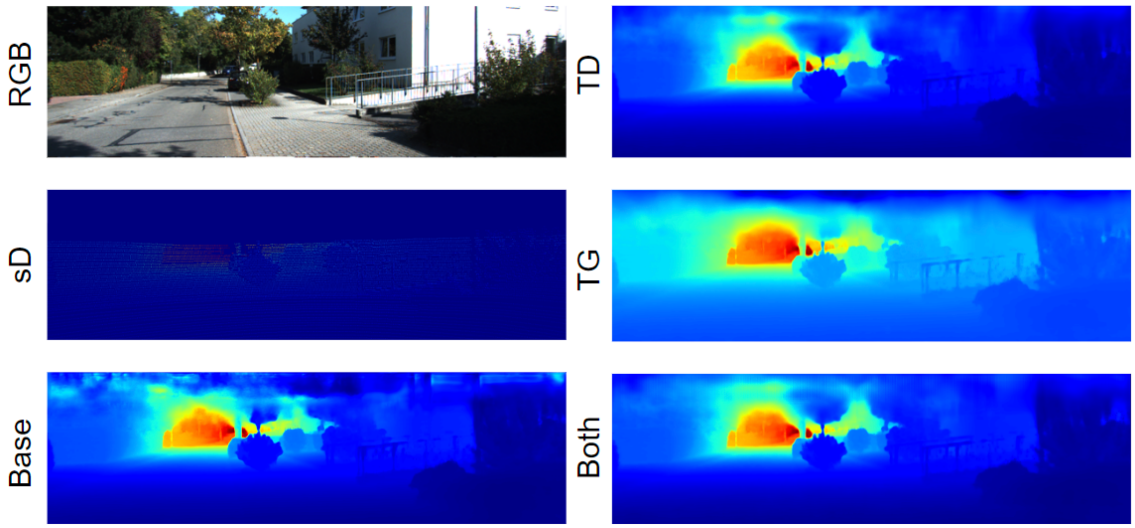

Figure 4.3: Comparison with different distillation methods. The inputs to the model are the RGB image and sparse depth data. The base is the prediction when trained without any distillation loss. TD is the prediction when trained with teacher depth distillation loss, TG is the prediction when trained with teacher gradient distillation loss, and Both utilizes both distillation loss in training.

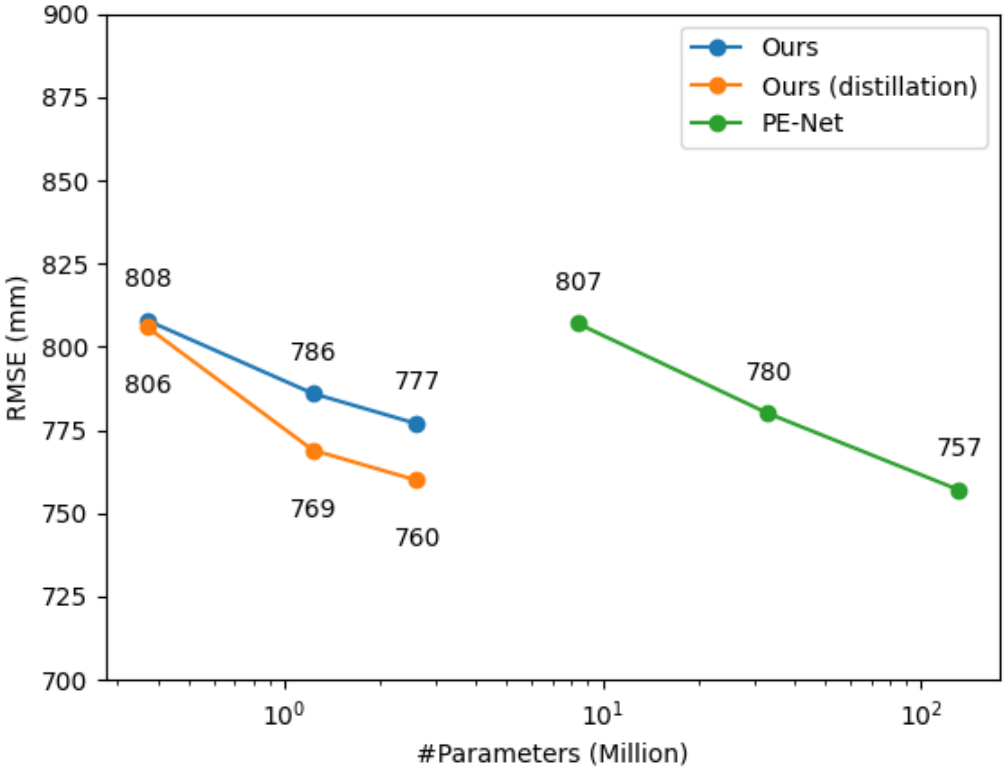

Figure 4.4: Comparison with different scales of models on KITTI Selected Validation Dataset verse parameter counts in logarithmic scale. As a common practice, no samples in the validation set are used in training the following models.

Chapter 5

Conclusions, Recommendations and Future Work

5.1 Conclusions

In this thesis, we have proposed a methodology for generating an efficient compact depth completion network from existing larger precision models. This top-down approach can take advantage of the existing efforts which represent the leading network architecture of high accuracy and make them suitable to be deployed in real-time applications.

After selecting the PE-Net as the base model, the proposed workflow starts with converting the normal convolution layer with efficient depth-wise separable convolutions, reducing the parameter footprint by a remarkable amount. Since inference speed is crucial, unlike many popular layer designs, no attention module is added in our unified depth-wise separable convolution blocks, given that the precision loss is not noticeable.

Further network compression exploits selective pruning via a random layer pruning test, in which every layer is pruned to get how much it affects the overall prediction. Utilizing this stability report, the over-parameterized layers are purged or shrunk, giving us the ability to fine-tune the network structure. We also provide a global scaling factor to customize the scale of the entire network by controlling the multiplier to the number of filters in each layer.

To compensate for precision loss from the compression, black-boxed knowledge distillation can be applied to further boost the performance of the compact model at different scales. The knowledge distillation method only requires the prediction results from the teacher model, and the distillation loss is formulated by the mean squared error between the teacher's and student model's predictions and their gradient.

As benchmarked in the KITTI dataset, not only the series of the model are competitive compared to other compact implementations as well as some larger ones, but also it can be a valuable addition to the literature. Because this work is not limited to a single model or problem, as it can be generalized to other model compression topics as well.

5.2 Future Work

In our current implementation of the decoder layers, bilinear upsampling replaces the transpose convolution operation and the checkerboard artifacts generated along it. However, this checkerboard artifact may return in the form of knowledge distillation. The distillation loss also provokes the 'MAE' when reducing the 'RMSE' as a limitation. Depending on the requirements, further work can explore different knowledge distillation methods that best fit.

Although we have tested with two attentions and proved to exclude the attention, a proper attention module is still considered helpful for better precision and generalizability. Further experiments may experience a better block design that balances both accuracy and speed.

The efficient network is targeting real-world applications, these applications are commonly deployed in embedded environments, such as the edge device. One technique that can dramatically speed up the inference and save resources is quantization. Quantization functions to execute some operations at lower memory precision, i.e., from fp64 to fp32 and int8. Some hardware can benefit from the reduced precision, accelerating the inference at a small cost of accuracy. We have not included the down-

stream process to quantize the model, but such research can surely provide insights into the area.

Bibliography

[1] M. Hu, S. Wang, B. Li, S. Ning, L. Fan, and X. Gong, "Towards precise and efficient image guided depth completion," 2021.

[2] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017.

[3] M. Tan and Q. V. Le, *Efficientnet: Rethinking model scaling for convolutional neural networks*, 2019. arXiv: 1905.11946 [cs.LG].

[4] L. Bai, Y. Zhao, M. Elhousni, and X. Huang, "Depthnet: Real-time lidar point cloud depth completion for autonomous vehicles," *IEEE Access*, vol. 8, pp. 227 825–227 833, 2020. DOI: 10.1109/ACCESS.2020.3045681.

[5] A. Li, Z. Yuan, Y. Ling, W. Chi, C. Zhang, *et al.*, "A multi-scale guided cascade hourglass network for depth completion," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 32–40.

[6] S. Wang, M. Hu, B. Li, and X. Gong, "Self-paced knowledge distillation for real-time image guided depth completion," *IEEE Signal Processing Letters*, vol. 29, pp. 867–871, 2022. DOI: 10.1109/LSP.2022.3154243.

[7] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[9] M.-H. Guo, C.-Z. Lu, Q. Hou, Z. Liu, M.-M. Cheng, and S.-M. Hu, "Segnext: Rethinking convolutional attention design for semantic segmentation," *arXiv preprint arXiv:2209.08575*, 2022.

[10] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[11] G. Yunhui, Y. Li, L. Wang, and T. Rosing, "Depthwise convolution is all you need for learning multiple visual domains," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8368–8375, Jul. 2019. DOI: 10.1609/aaai.v33i01.33018368.

[12] L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise separable convolutions for neural machine translation," *arXiv preprint arXiv:1706.03059*, 2017.

[13] L. Bai, Y. Zhao, and X. Huang, "A cnn accelerator on fpga using depthwise separable convolution," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 10, pp. 1415–1419, 2018.

[14] K. Kamal, Z. Yin, M. Wu, and Z. Wu, "Depthwise separable convolution architectures for plant disease classification," *Computers and Electronics in Agriculture*, vol. 165, p. 104 948, 2019.

[15] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," *Lecture Notes in Computer Science*, 122–138, 2018, ISSN: 1611-3349. DOI: 10.1007/978-3-030-01264-9_8. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-01264-9_8.

[16] D. Nazir, M. Liwicki, D. Stricker, and M. Z. Afzal, *Semattnet: Towards attention-based semantic aware guided depth completion*, 2022. arXiv: 2204.13635 [cs.CV].

[17] J. Liu and C. Jung, "Nnnet: New normal guided depth completion from sparse lidar data and single color image," *IEEE Access*, vol. 10, pp. 114 252–114 261, 2022. DOI: 10.1109/ACCESS.2022.3215546.

[18] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz, "Learning affinity via spatial propagation networks," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[19] X. Cheng, P. Wang, and R. Yang, "Learning depth with convolutional spatial propagation network," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2361–2379, 2019.

[20] X. Cheng, P. Wang, C. Guan, and R. Yang, "Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 10615–10622, 2020, ISSN: 2159-5399. DOI: 10.1609/aaai.v34i07.6635. [Online]. Available: http://dx.doi.org/10.1609/AAAI.V34I07.6635.

[21] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions. arxiv 2015," *arXiv preprint arXiv:1511.07122*, vol. 615, 2016.

[22] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[23] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1389–1397.

[24] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[25] C. Buciluundefined, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06, Philadelphia, PA, USA: Association for Computing Machinery, 2006, 535–541, ISBN: 1595933395. DOI: 10.1145/1150402.1150464. [Online]. Available: https://doi.org/10.1145/1150402.1150464.

[26] A. Galstyan and P. R. Cohen, "Empirical comparison of "hard" and "soft" label propagation for relational classification," in *International Conference on Inductive Logic Programming*, Springer, 2007, pp. 98–111.

[27] T. Y. Liu, P. Agrawal, A. Chen, B.-W. Hong, and A. Wong, *Monitored distillation for positive congruent depth completion*, 2022. arXiv: 2203.16034 [cs.CV].

[28] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[29] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, *Pruning convolutional neural networks for resource efficient inference*, 2016. arXiv: 1611.06440 [cs.LG].

[30] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008. DOI: 10.1109/TPAMI.2007.1166.

[31] J. Qiu *et al.*, "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image," Jun. 2019, pp. 3308–3317. DOI: 10.1109/CVPR.2019.00343.

[32] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera," 2019.

[33] A. Eldesokey, M. Felsberg, and F. S. Khan, "Confidence propagation through cnns for guided sparse depth regression," *arXiv preprint arXiv:1811.01791*, 2018.

[34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[35] X. Liang and C. Jung, "Agnet: Attention guided sparse depth completion using convolutional neural networks," *IEEE Access*, vol. 10, pp. 10 514–10 522, 2022. DOI: 10.1109/ACCESS.2022.3144596.

[36] D. Nazir, A. Pagani, M. Liwicki, D. Stricker, and M. Z. Afzal, "Semattnet: Towards attention-based semantic aware guided depth completion," *IEEE Access*, pp. 1–1, 2022. DOI: 10.1109/ACCESS.2022.3214316.

[37] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. DOI: 10.1109/cvpr.2018.00745. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00745.

[38] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. DOI: 10.1109/cvpr46437.2021.01350. [Online]. Available: http://dx.doi.org/10.1109/CVPR46437.2021.01350.

[39] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[40] S. Hwang, J. Lee, W. J. Kim, S. Woo, K. Lee, and S. Lee, "Lidar depth completion using color-embedded information via knowledge distillation," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2021. DOI: 10.1109/TITS.2021.3129901.

[41] Z. Shen and M. Savvides, *Meal v2: Boosting vanilla resnet-50 to 80imagenet without tricks*, 2020. arXiv: 2009.08453 [cs.CV].