ADAPTING BILATERAL NETWORKS TO MONOCULAR DEPTH ESTIMATION FOR REAL-TIME INFERENCE

A Thesis

by

Sami Menteş

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
May 2022

ADAPTING BILATERAL NETWORKS TO MONOCULAR DEPTH ESTIMATION FOR REAL-TIME INFERENCE

Approved by:

Asst. Dr. M. Furkan Kıraç, Advisor
Department of Computer Science
Özyeğin University

Asst. Dr. Berk Gökberk
Department of Computer Engineering
Boğaziçi University

Asst. Dr. Reyhan Aydoğan
Department of Computer Science
Özyeğin University

Date Approved: 26 May 2022

To my family and the soul of my dear friend ...

ABSTRACT

Monocular Depth Estimation (MDE) is a fundamental computer vision application area for many industry-related advances. Due to its deployment needs, the inference time of the depth estimation algorithm also plays a crucial role among other accuracy metrics. With the recent advances in Convolutional Neural Networks (CNNs) on other time-constrained computer vision tasks, many efficient feature extractors have been studied and adopted from MDE models as the backbone. Although those feature extractors have shown significant improvement in throughput, the widely-used encoder-decoder architecture used by Real-time MDE models also relies on a decoder network for upsampling. Following a similar approach, stacking multi-channel convolutional layers on a decoder hinders the inference time. This study investigates the benefits of Bilateral Networks in Real-time MDE tasks. During our research, we first manipulate the structure of a recently introduced real-time segmentation model (STDC-Seg) for the MDE problem. Once we attain real-time inference speed, we tailor the backbone structure and attention modules of the model for the needs of MDE to improve prediction accuracy. Finally, we train the models on the well-known KITTI dataset and compare our results with the models of the KITTI Eigen Split MDE Benchmark along with the previous real-time models. Our experimental results show that our real-time method achieves on-par metric performance with state-of-the-art models that are not subject to any time-constraint.

ÖZETÇE

Monoküler Derinlik Tahmini (MDT), endüstri ile ilgili birçok gelişme için önemli bir bilgisayarlı görü uygulama alanıdır. Kullanım ihtiyaçları nedeniyle, derinlik tahmin algoritmasının çıkarım süresi de diğer doğruluk ölçütleri ile birlikte çok önemli bir rol oynar. Evrişimsel Sinir Ağlarının (ESA) diğer zaman kısıtlamalı bilgisayar görme görevlerinde kullanımındaki son gelişmelerle birlikte, birçok kodlayıcı incelendi ve MDT modellerinde omurga olarak kullanıldı. Bu kodlayıcılar verimde büyük gelişme göstermiş olsa da, gerçek zamanlı modeller tarafından kullanılan popüler kodlayıcı-kod çözücü mimarisi ayrıca çözünürlük yükseltebilmek için bir kod çözücü ağına dayanır. Benzer bir yaklaşımı izleyerek, çok kanallı evrişimli katmanları bir kod çözücü üzerinde istiflemek, hesaplama süresini önemli ölçüde yavaşlatır. Bu çalışmada, Gerçek Zamanlı MDT için İkili Ağların kullanımını araştırıyoruz. Bunu başarmak için, önce MDT problemi için yakın zamanda tanıtılan bir gerçek zamanlı segmentasyon modelini (STDC-Seg) manipüle ettik. Gerçek zamanlı çıkarım hızına ulaşıldığında, doğruluğu daha da geliştirmek için ağın omurga yapısını ve dikkat modüllerini MDT'nin ihtiyaçlarına göre uyarladık. Son olarak, modelleri KITTI veri seti üzerinde Eigen ayrımına bağlı kalarak eğittik ve sonuçlarımızı KITTI üzerinde çalışılan modellerle ve ayrıca gerçek zamanlı modellerle karşılaştırdık. Deneysel sonuçlarımız,metodumuzun gerçek zamanlı çıkarım elde ederken zaman kısıtlamasına tabi olmayan son teknoloji modellerle de yakın performansta sonuçlar elde ettiğini göstermektedir.

ACKNOWLEDGEMENTS

To begin with, I owe a debt of gratitude to my advisor Assist. Prof. Dr. Furkan Kıraç, who has guided me in the most prominent years of my life. Throughout his extraordinary courses of distilled knowledge, I had a chance to build a rare skill-set that has shifted my career to the next level. I am so glad to have a supervisor who not only has such a deep insight into his subject but also believes in collaborative success. During our research meetings and personal talks, he never hung back from sharing his immense knowledge and experience generously. Most importantly, his sincere and supportive personality has resulted in a culture that spreads out among the senior members of our lab. I have experienced how such an efficient pattern amplifies his help beyond what a single person can achieve. Besides profiting from his academic strengths, I also had a chance to examine his exceptional presentation skills and how he manages to scope subjects of vast complexity for adopting them in my future career.

I would like to thank the most senior member of our lab Barış Özcan for training me on academic writing and providing me with bright ideas for my research. Special thanks to Furkan Kınlı, for spending long hours day and night, patiently teaching me the intricate details of the crucial Deep Learning and Computer Vision frameworks, sharing his dexterous skills anytime he can, and walking aside me during my tough days. He is my big brother for life from now on.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Over the last decade, advances in robotics, drones, and augmented reality became prominent not only among industries but also influenced pop culture. Swarm of drones flying around in movies, big international companies searching for ways to integrate artificial intelligence into their solutions, and social media posts about futuristic augmented reality products that do not exist yet has thrilled our expectations. Although these advanced technologies are still facing quite complicated challenges, the advances in Computer Vision literature has matured the field to provide practical products in relevant industries. Only the mobile augmented reality market alone is estimated to have 16.84 million dollars of revenue by the end of 2023, more than quadruple of the year 2019 [15]. The numbers go monumental when the scope is enlarged to robotics and automation with a 27.78 billion dollars of market valuation for 2020 [16]. Similarly, the car industry moving towards fully and semi-autonomous systems is bringing a vast disruption.

A car moving autonomously in streets, an augmented reality app rendering a simulation into a real-life scene, or a drone navigating in a closed environment, desperately needs perception of the surrounding world to function properly. The living creatures with an eye have developed a robust compression of the world which generalizes quite nicely to various settings. As humans, for instance, we can easily understand a scene organization of a room or an open space under different lighting conditions and contexts. We can even perceive simulated scenes in computer games or cartoons that have ambiguous scales and styles. Considering the possible variations and the complexity of the world, formalizing hand-crafted features to solve a problem of perception is extremely difficult even in highly controlled environments.

How we interact with the three-dimensional world relies on perceiving relative distances of objects, namely the skill of depth estimation. Learnable methods in the field of computer vision, especially the movement that has been initiated by CNNs and deep learning, have shown to be a great technique for attacking this problem. The futile strategies aiming to structure depth estimation as a well-defined problem have been replaced with those methods, and the literature expanded rapidly upon improving existing techniques.

The deep learning solutions for the depth estimation problem have excelled in metric performances on certain datasets over time and have resulted in quite competitive benchmarks. However, inference time is also a crucial standard considering the deployment scenarios of depth estimation models. The accuracy-speed trade-off in the model composition is another challenge that attracted the attention of academia due to its industrial value. Most of the depth estimation models follow the well-studied U-Net-like encoder-decoder architecture which is designed solely for accuracy, discarding the speed concerns during inference. This thesis examines the potential advantages of using Bilateral Networks in the Monocular Depth Estimation problem to address this accuracy-speed trade-off.

Our proposed model, STDC-Depth is trained on 23k images of KITTI [17] and is tested concerning the famous Eigen Split [18]. Our experimental study shows that a bilateral approach adapted to the Monocular Depth Estimation problem achieves on-par metric performance with the standard state-of-the-art models while sustaining real-time throughput.

CHAPTER II

BACKGROUND INFORMATION

2.1 Introduction

In this section, we present the Depth Estimation Problem, reveal the branches of Depth Estimation along with their motivations, and examine Monocular Depth Estimation in particular. We then go over the differentiable learning methods, which are mainly used in our domain, and examine the mechanism of CNNs and why they are preferred for image understanding. Since most of the MDE literature is built on top of models using encoder-decoder architectures, we cover the most popular feature extractor architectures used for full-performance or real-time inference, along with upsampling strategies of previously proposed decoders. Next, we explore various loss functions and techniques tailored for the Monocular Depth Estimation task. Finally, we explain the quantitative metrics used to evaluate the performance of these methods.

2.2 Depth Estimation

Capturing the distance information in a scene is crucial for 3D scene understanding. Emerging technological advances in the industry such as Robotics, Augmented Reality, Scene/Object Reconstruction, and autonomous driving rely on algorithms that are capable of interpreting depth. Such a market demand has led depth estimation to become one of the most studied research fields in Computer Vision. Yet, estimating the depth accurately in an environment is an unsolved problem. Considering the complexity of the world we live in, even a highly restricted context has infinitely many scene variations.

2.3 Measurement Instruments

Due to the peculiar challenges of this task, depth estimation is not constrained by the usage of RGB cameras but has been studied with various sensory instruments possessing distinct strengths and drawbacks. LIDAR, an expensive active range sensor, is a popular choice in the autonomous driving industry owing to its metric precision in calculating laser reflections.

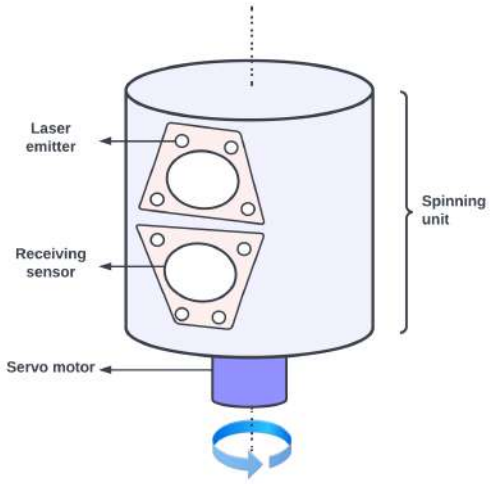

Figure 1: A basic demonstration of a typical LIDAR Sensor

As Figure 1 shows, the head unit spins with the help of a servo motor and shoots laser beams to the surroundings. Meanwhile, the receiver sensors collect the beams that are refracted from objects and calculate the refraction distance according to the relative motion of the head unit. At the end of the process, each refraction represents a point in the point cloud. With the rising interest in autonomous vehicles, LIDAR sensors have become popular car equipment for acquiring precise point clouds. However, mechanical limitations of this retrieval method result in a sparse representation of a scene, which may lack critical information for certain tasks. This limitation precipitated another research topic called depth completion, where the objective is to interpolate missing regions of an acquired depth map.

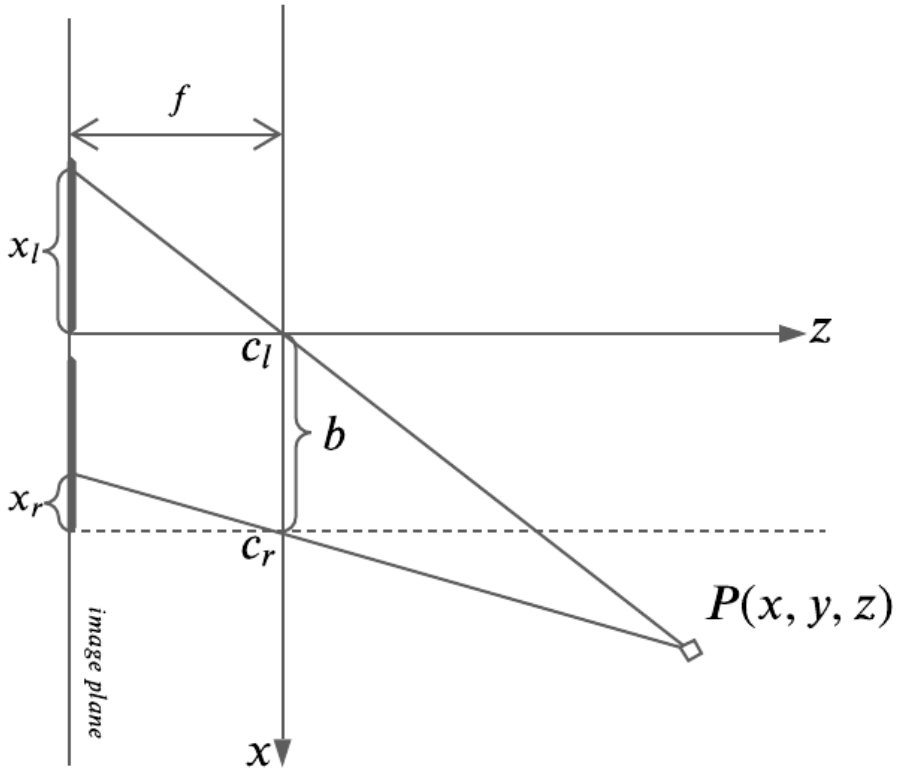

Figure 2: The representation of triangulation with a pinhole camera model

2.4 Binocular Depth Estimation

On the other hand, RGB cameras are tempting for depth retrieval due to their dense nature and notably lower price. Yet, as its name suggests, RGB cameras were built only to capture light for representing the visible spectrum as a combination of colors red, green, and blue. A well-defined approach for acquiring distance information with RGB cameras is to add an additional camera into a setup with a known distance and exploit geometrical properties. This method called triangulation utilizes the distance between stereo cameras, *i.e.*, baseline, for calculating each pixel's relative distance. Figure 2 demonstrates triangulation method for a single point for simplicity, where f denotes focal length, b denotes baseline, c_l and c_r denotes pinhole of left and pinhole of right cameras respectively. The y axis is perpendicular to the plane XZ.

Triangular similarities yield,

$$\frac{x_l}{f} = \frac{x}{z} \tag{1}$$

$$\frac{x_r}{f} = \frac{x-b}{z} \tag{2}$$

Combining those equations, we derive z as follows,

$$z = \frac{bf}{x_l - x_r} \tag{3}$$

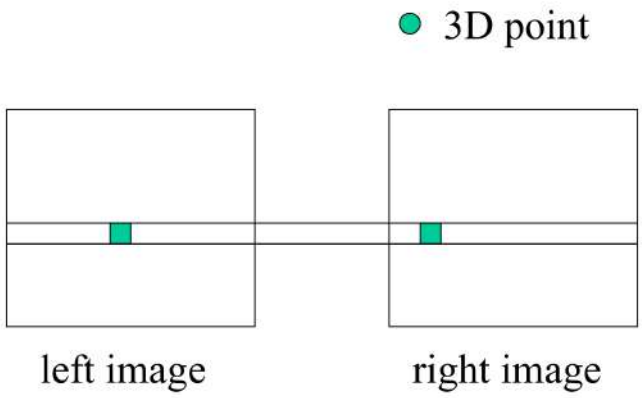

Figure 3: The representation of disparity map formation projected on left image

An image, however, is formed by multiple points which are discretely represented as pixels. In order to organize triangulation calculations for numerous pixels, disparity maps are used. A disparity map is a 2D matrix representation of the pixel-wise shift in horizontal axis among left and right images as shown in Figure 3. To construct the disparity map, an empty two-dimensional matrix with the size of the images is created. Later, assuming the left image is chosen as the reference, the disparity map is filled by subtracting the horizontal location of each pixel on the right image from the corresponding pixel's horizontal location on the left image. This subtraction operation that is referred to in the denominator of the Equation 3 is called disparity. When the baseline and focal distance information is unknown in a stereo setup, this map denotes the inverse depth, which is proportional to disparity by the unknown factor bf.

Figure 4: An example scenery having multiple grid like structures.

Nevertheless, for retrieving this pixel-wise horizontal shift, the corresponding pixels in the left and right images must be matched at first. This prerequisite is the major downside of binocular depth estimation and why stereo cameras are not the absolute solution. Although stereo matching is a well-studied method, it faces many obstacles due to the nature of scenes in real life. Particularly, reflections and patterns are a major issue even for advanced matching algorithms. Reflections, commonly occurring on the windshield of cars in driveway scenes, for instance, distorts the bijective nature of the matching problem. The same pixel block appearing also on a reflective surface is a subject of confusion, resulting in low-quality disparity maps. Another common issue is arising patterns in scenes. As the iron fence on the bridge in the example scene of Figure 4, any scene with a fence or a grid presents many key points hard to tell apart.

Therefore even strictly calibrated stereo camera setups such as ZED on advanced matching methods are not the perfect solution. Additionally, the pinhole camera model is not feasible in real life, modern cameras have lenses. This introduces additional degrees of freedom called lens radial distortion which must be adjusted additionally with respect to calibration information of the setup.

2.5 Motion Parallax

Another useful depth information is present in the temporal domain. In a video stream of RGB images where the camera location is not static, a phenomenon called Motion Parallax occurs. In a three-dimensional Euclidean Space, perspective projection into two dimensions results in far objects appearing smaller than closer objects. Likewise, the distance between two far objects also appears shorter, causing projections of the distant points to deviate less on the image plane during a camera motion. Tracking pixel deviations in a sequence of images, therefore, gives us an idea about the depth information of the scenery. On the other hand, such a method discards the relative motion of any dynamic object in a scene. For instance, an object moving with the same velocity as the global motion is deduced as infinitely far away. Considering the autonomous driving task, it is a quite luxurious assumption. Tracking pixels involved, this method also relies heavily on matching pixel-to-pixel correspondence.

2.6 Monocular Depth Estimation

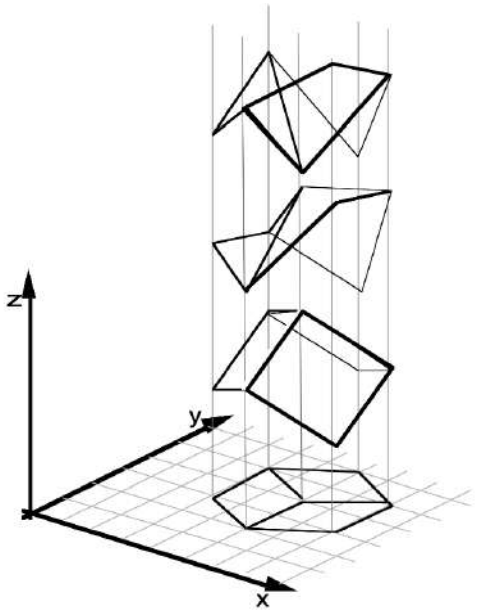

Figure 5: An illustration of infinitely many 3D objects having the exact same projection on XY plane, taken from [1]

We do not shoot lasers from our eyes, yet we are highly capable of orienting ourselves in this three-dimensional world even using a single eye. Likewise, when we look at a picture we can easily tell the relative depth relation of the scene composition. This is due to our neural network which has learned to interpret depth by exploiting pictorial cues. With the learnable methods in computer vision gaining traction, such a skill inspired research in Monocular Depth Estimation. Moreover, the simplicity and the relatively lower cost of using a single RGB camera encouraged advances in the field.

2.6.1 Converging lines

In a single image, an obvious depth cue is the behavior of parallel lines in a scene with respect to perspective. A line can be fitted into any continuum of a linear pattern in natural scenery, or more likely of a man-made structure. Those lines tend to converge in distance due to the properties of perspective projection, which creates a perception of depth.

2.6.2 Occlusion

A different monocular depth cue occurs because of the behavior of light on the macro scale. Most of the objects are opaque, meaning they do not let light pass through. This behavior presents as the phenomenon called occlusion in a captured image. Depending on the angle of the camera, some part of light refracted from an object in the scene will be trapped behind a closer object before reaching the camera lens, translating to occlusion on the formed image. Therefore, the occluding objects represent a closer distance.

2.6.3 Relative Size

The next monocular depth cue is a direct cause of perspective projection. Considering the pinhole camera model, the triangular similarities yield far away objects to appear smaller, whereas, the closer ones appear bigger on the image plane. Familiar structure's size relation, such as similar instances appearing in different

sizes, possesses depth information. For example, in a picture of an auditorium taken from a lecturer's perspective, students in the front seats appear significantly bigger than the students in the back even though they are both human and share similar sizes.

Note that extracting depth from a single image is an ill-posed problem. As Figure 5 shows, there are infinitely many three-dimensional objects that project the same shape on a two-dimensional plane.

2.7 Differentiable Learning

Explicitly programming computers to achieve vision-based tasks is not practical due to infinitely many scene variations occurring even in highly controlled environments. Perceiving a three-dimensional world is a complicated and complex skill that living creatures evolved over millions of years to handle light, scale, translation, and many more variations for extracting necessary information. Hand-crafted features and strong assumptions in the Machine Learning era for structuring visual perception as a defined problem fall short due to the high bias involved. Although these hand-crafted features are designed to provide a certain amount of robustness, they need to be tailored specifically for the task at hand, thus lacking generalizability.

End-to-end learning with a model capable of adapting its internal parameters for achieving a task, on the other hand, decreases bias significantly and resembles natural intelligence much more. This model, namely Neural Networks, is influenced by the interconnected neuron structure of the brain. Precisely, any function represented as a series of firing neuron activity is reduced into a computational graph.

Each neuron is represented by a function node, *i.e.*, perceptron, which has the following form;

$$p(x) = \sigma(Wx + b) \tag{4}$$

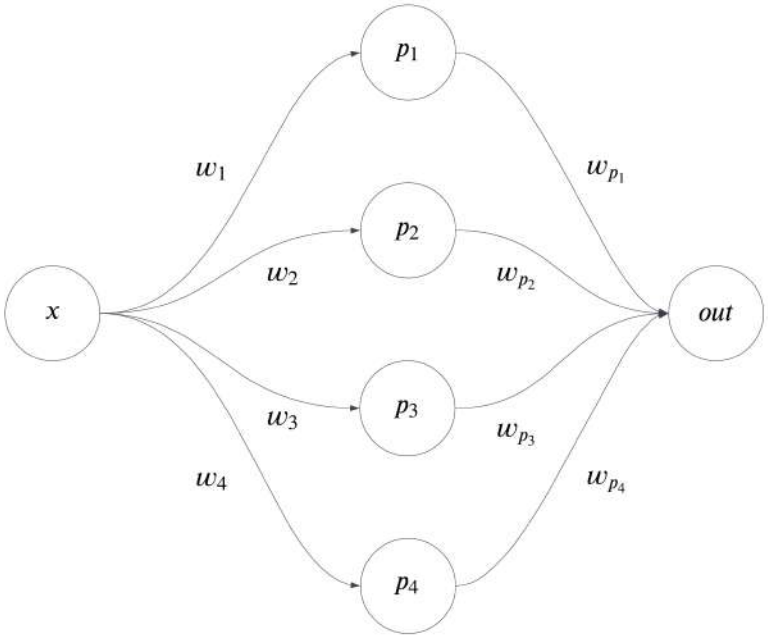

Figure 6: An example of a Neural Network with a single hidden layer of 4 perceptrons

where x is the input, W is the weight of the incoming inputs, σ is an activation function and b is the bias. During learning, *i.e.*, training, each perceptron of this computational graph updates relevant weights and biases appropriately in order to minimize a loss function.

Figure 6 shows a simple neural network architecture where each circle represents a perceptron and arrows show forward flow and their weights. For this example, the input layer is a single perceptron taking a value x. The group of the perceptrons aligned in the middle section of the figure is referred to as the *hidden layer* since it is an internal mechanism of a black-box model. Although the example NN figure has a single hidden layer, the number of hidden layers may vary regarding the need for variance. Each perceptron of the hidden layer in Figure 6 outputs as follows,

$$p_i(x) = \sigma_{sigmoid}(w_i x + b_i) \tag{5}$$

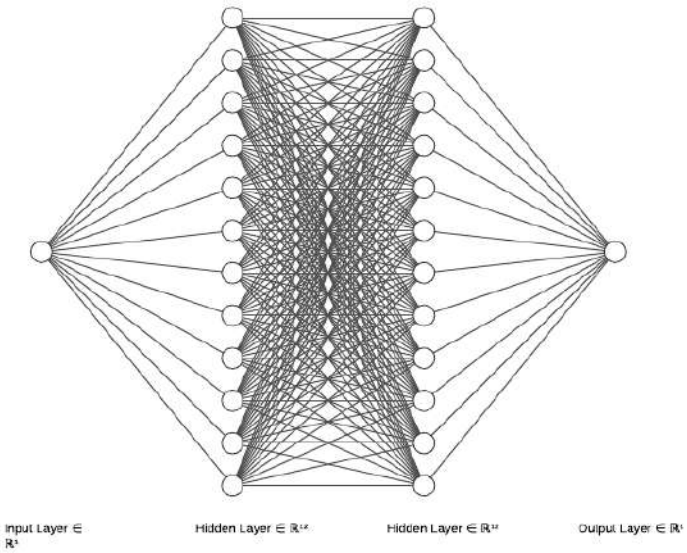

Figure 7: The representation of a Fully Connected Neural Network with 2 hidden layers and a single input output

Bias value b_i is specific to perceptron p_i. Connecting each perceptron to every other perceptron of the next layer, *i.e.*, fully-connected architecture, is a general design practice since the NN can set any connection weight w to zero during learning. Likewise, in our example, the output layer receives all four activations coming from the hidden layer and evaluates them as follows,

$$out = \sigma_{sigmoid} \sum_{i=1}^{4} (w_{p_i} p_i + b_{out}) \tag{6}$$

2.7.1 Activation Functions

Given the example in Figure 6, a NN is nothing but recursive perceptron functions. Non-linear activations are crucial for making this recursive setting capable of expressing a complex function. The number of interconnected perceptrons and their depth correlates with how complex the representing function can get, which is also referred to as the variance of the model. A trivial NN compromising two perceptrons shown in Figure 8 can be represented recursively as follows if it has no activation function, *i.e.*, with linear activation,

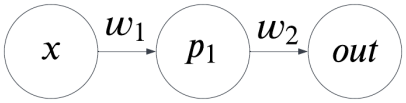

Figure 8: An illustration of two connected perceptrons

$$out = w_2(w_1 x + b_{p_1}) + b_{out} \tag{7}$$

by combining two equations,

$$out = w_2 p_1 + b_{out} \tag{8}$$

$$p_1 = w_1 x + b_{p_1} \tag{9}$$

At this point, although the example NN has two perceptrons, the resulting Equation 7 is nothing but a linear function. The overall expressivity of this NN can also be obtained by a single perceptron as the Equation 9 already represents a linear function. Introducing an intermediary non-linear function right in between the two perceptions, however, reinforces the practice of deeply stacking perceptrons since now the output function,

$$out = w_2 \sigma_{non-linear}(w_1 x + b_{p_1}) + b_{out} \tag{10}$$

has a broader variance.

One of the earliest activation functions is the sigmoid function [19]. Again inspired by the nature of neurons, sigmoid activation squeezes the input value into zero and one range. A zero output represents no activation of an organic neuron and one represents the maximum voltage capacity. Formally sigmoid activation is calculated as,

$$\sigma_{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{11}$$

which corresponds to Figure 9.

The most common activation function is Rectified Linear Unit or ReLU [20]

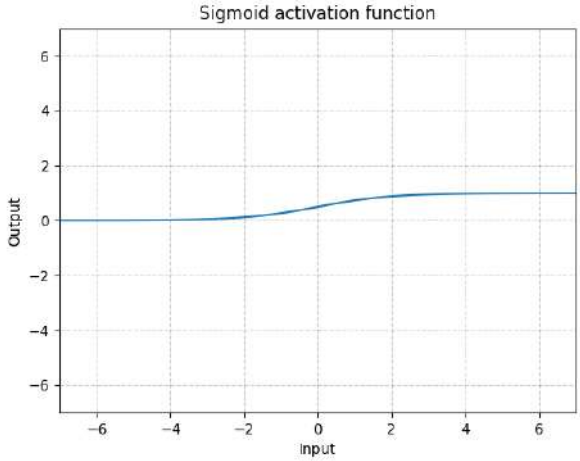

Figure 9: The sigmoid function, also referred as logistic curve

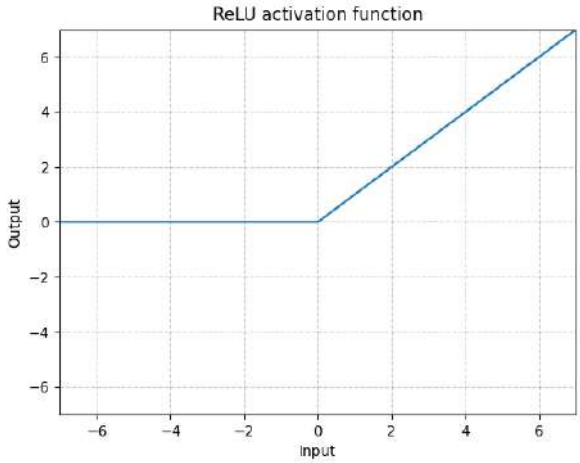

Figure 10: The Rectified Linear Unit function graph

in short. The ReLU activation function is,

$$ReLU(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 1 \end{cases} \tag{12}$$

which corresponds to Figure 10.

2.7.2 CNN

Convolutional Neural Network [21] is a design aiming to address the challenges of using fully-connected Neural Networks in tasks involving spatial information such as images. Although fully-connected NNs proved to be expressive, they are not practically desirable for vision-specific problems due to certain limitations. One of the prominent practical flaws is scalability. Providing each pixel as an input neuron to a Fully Connected Neural Network (FCN) not only restricts a fixed input resolution but also allocates a huge amount of redundant parameters considering the repetitive nature of the image domain. Given a high-resolution image of size 1920x1080, it takes more than 2 million nodes, thus parameters, just to receive an image from the input layer. This indicates even a following single hidden layer of the same size, full connectivity adds a square amount more to the learning process. A straightforward approach might be to limit the number of input nodes to a manageable amount and downsample any given image for feeding it to the network, but such a practice would cost the high-frequency information.

Another crucial issue of using vanilla neural networks for the image domain is how images represent information. The generic fully connected neural network architectures assume that each input node encodes unique and atomic information, thus each node has its role to process specific information and convey it to the next nodes. In an image, however, an ambiguous group of pixels forms meaning as a composition. For instance, as can be seen in Figure 11, an image shifted by a single pixel to any direction means the same thing for human visual perception to the point that it is unnoticeable.

For an FCN, the two images are entirely different since input nodes that are ignorant of the other's presence, are now taking the next pixel which is likely not to represent the same information. Figure 12 illustrates how image translation affects FCN perception, where each pixel is represented by an image patch for the sake of intuition. An FCN explicitly configured for taking a pixel-wise flattened image and output a positive activation would fail miserably to replicate the same output

Figure 11: An image shifted one pixel right

after a small translation. In figure 12 for instance, the fifth perceptron expecting the face patch would receive an orange background patch which has nothing to do with a human body. This inconvenience is a result of image data not being pure information. An image is a representation, a processed result obtained by framing and projecting a 3D world on a limited 2D plane. In the image processing field, this representation issue is handled by interpreting the image as a signal and applying convolution operations.

Formally convolution operation is represented as;

$$(f * g)(t) = \int f(\tau)g(t-\tau)d\tau \tag{13}$$

where the function f is the subject signal to be transformed or analyzed and function g is the operator for the desired transformation. Integrating the product of the two functions with respect to τ, slides the reversed function g over f continuously. This practice allows convolution operation to be translation and scale invariant as a practical solution to the above-mentioned problems.

Since images are digitized, the function f corresponds to a discrete matrix representation of the image and the function g corresponds to a square filter matrix, *i.e.*, kernel. The continuous flow of the convolution integral becomes a kernel

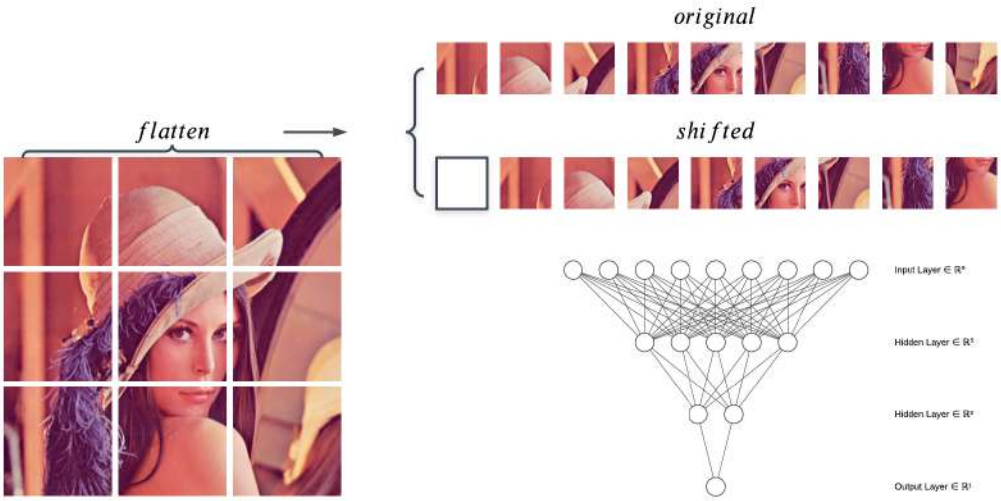

Figure 12: The image translation problem from a Fully Connected Network input perspective, input pixels are represented by image patches

translation, *i.e.*, stride, in the image domain.

Processing images with convolutions is yet performed by predefined hand-crafted kernels customized for specific use. Figure 13 shows some examples and their purpose respectively.

LeNet [21], is the first work to adapt the differentiable learning framework into convolution operations for image understanding. The model replaces the fully connected layers with stacks of multiple convolution kernels with learnable parameters. As in the case with neural networks, an activation function is applied at the end of each layer for achieving non-linearity. In this design, stacked convolution layers are extracting distinct features at each layer. In order to capture different levels of scale, the feature map sizes must be reduced towards deeper layers. Pooling operation, shown in Figure 14, down-samples a given feature map by a chosen strategy. Max-pooling strategy, for instance, selects the brightest pixel in the receptive kernel as a representation, whereas, the average pooling layer averages pixel values.

Downsampling by pooling allows combining features from multiple frequencies. The convolution operations capture high frequencies in the initial layers such as lines, textures, and detail patterns. For instance, initial convolution kernels might

Figure 13: Some examples of custom kernels and their purposes from top to bottom; mean kernel for blurring, Sobel kernels for vertical and horizontal edge detection

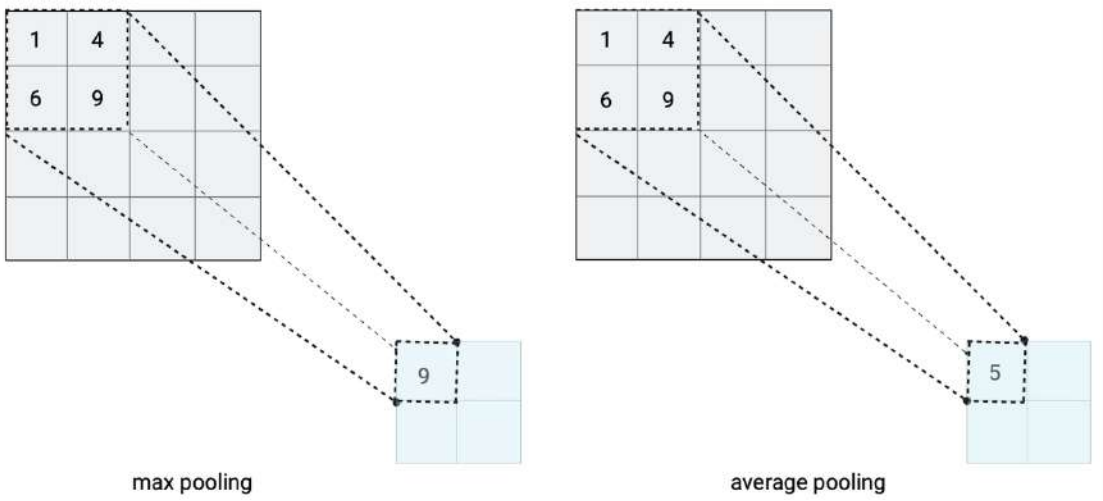

Figure 14: The pooling operation with a kernel of 2×2, max pooling shown on left and average pooling on right

adapt their weights to form Sobel filters [22] for capturing vertical and horizontal lines. Deep CNN architectures aim binding those high-frequency information going through deeper layers. Consecutive convolutions and pooling layers leverage larger receptive fields during this process. Pooled features coming from subsequent layer unites via following convolutions and results in lower frequency information. This

corresponds to capturing more global features, such as people, cars, or the sky in the background.

2.7.3 Normalization

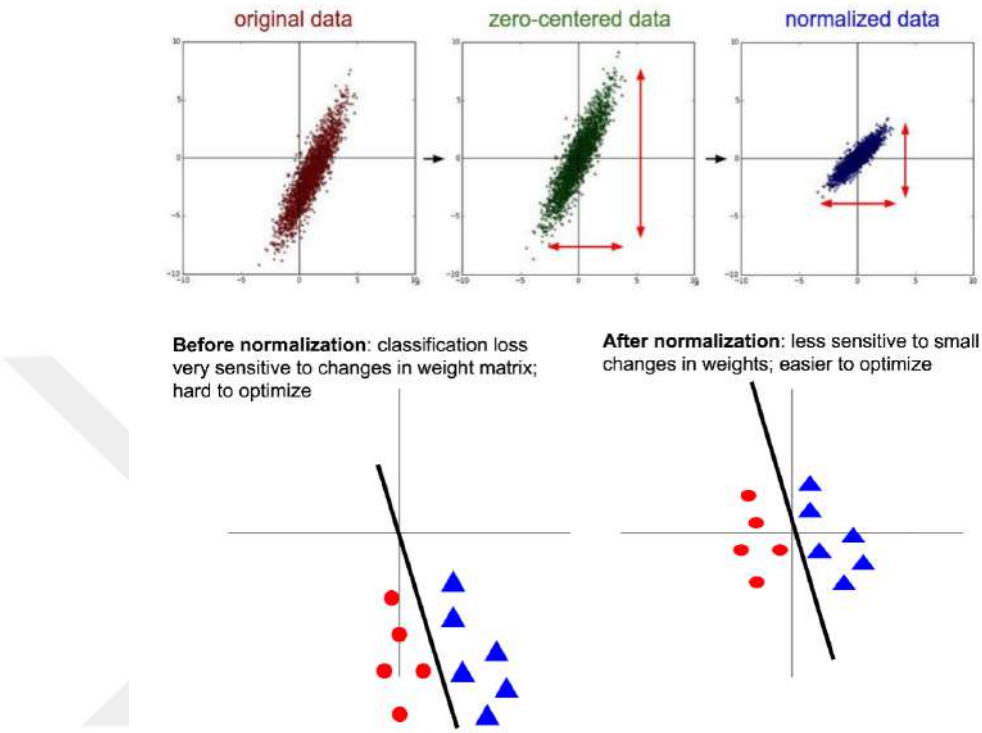

Figure 15: The data normalization process visualized on top, how it benefits demonstrated on bottom from Stanford Lectures [2]

Normalization of the data in learnable models is a crucial pre-processing step for making the model training easy and stable. The units of the different features of an arbitrary dataset are likely to be on different scales and fall far from the center point 0. Even a simple dummy case of a linear regression algorithm would largely benefit from normalizing the dataset. A problem of estimating the number of medals of an Olympic boxer given the number of training years and weight in pounds is immensely sensitive to model parameters, therefore, learning relies on minuscule parameter adjustments. While the training years of the dataset are likely to range on the scale of a tenth, the weight in terms of pounds will be in the hundreds. This difference in scale squeezes the distribution of the dataset in an axis as Figure 15 shows.

As a model can benefit from normalization as a pre-processing step, the middle layers of a deep network are no different. Additionally, those layers face a secondary challenge. We have already covered that the parameters of a model are updated constantly during training. From a middle layer perspective, that corresponds to data distribution coming from a previous layer changing constantly. This phenomenon called covariance shift cripples learning in the middle layers.

Batch-normalisation [23] addresses this problem by normalizing inter-layer data flow at each step. A forward pass involving a batch of the data is processed at every layer of the network with the following equation;

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta \tag{14}$$

where x is the incoming batch of features, y is normalized features, γ, and β are learnable parameters for normalizing data over batch dimension. Generally, γ and β are initialized as a vector of ones and vector of zeros respectively. Since the input is not in the batch form during inference, *i.e.*, $batchsize = 1$, those parameters are replaced with respect to running averages calculated during training time.

2.7.4 AutoEncoders

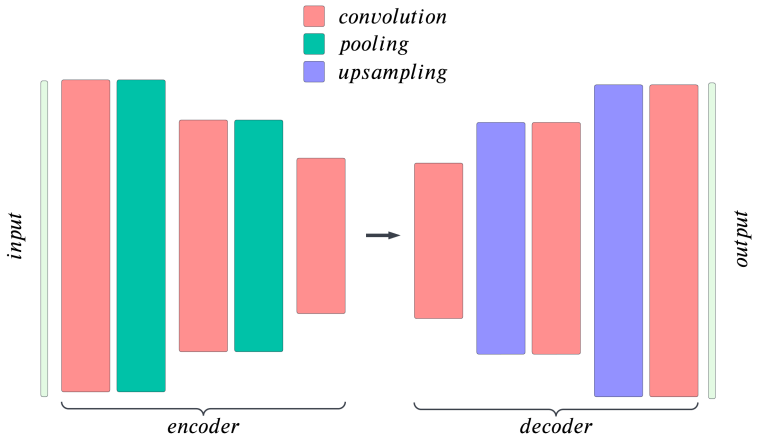

Figure 16: The overview of a simple autoencoder

Downsampling is crucial for CNNs to output feature maps that convey information from various levels and scales. Processing those feature maps on final

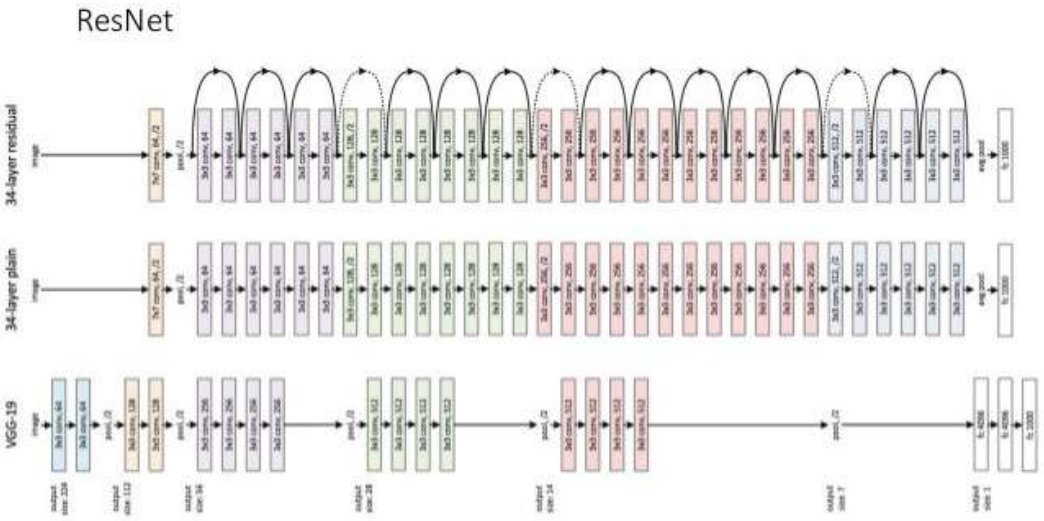

Figure 17: ResNet [3], depth equivalent of ResNet and AlexNet from top to bottom

layers, for instance, flattening and feeding it to an FCN as in the case of [21], is a practice to extract meaning from images for an appropriate aim. [21] use its final layers to predict the class of a given handwritten digit by applying a softmax activation [24] to the output layer where each perceptron activation represents the likelihood of a class. That is called one-hot vector encoding and is used extensively for classification tasks. On the other hand, many computer vision tasks; Segmentation, depth prediction, and image-to-image translation to name a few, require a pixel-level output, which corresponds to an image of the same size as the input. Formally monocular depth prediction task enters the pixel-wise regression category, meaning the output is a depth map having the dimensions of the input image. A straightforward approach might be to design the output layer of the FCN network as the desired image format by encoding each pixel with a perceptron. Such a design, as using perceptrons in the input layer, is not only unscalable concerning the number of parameters, but also non-generalizable since the input image size might vary in a broader use case.

Autoencoders are Fully Convolutional Neural Networks, meaning it has no other layers but only convolutional ones designed specifically for pixel-level guidance cases. This structure allows the output feature map to vary depending on

the input image size and protects the aspect ratio. As shown in Figure 16, an autoencoder is formed by two main partitions, the encoder, and the decoder.

The encoder part follows the general classification CNNs without the fully-connected output layers. This section of the network aims to retract the high-level information present in the image and encode this information concisely into feature maps. Those feature maps are also referred to as the latent vector when it is flattened into a single dimension. A typical encoder structure has multiple convolutional layers followed by pooling layers for decreasing the spatial complexity of the input while maintaining the necessary information. In this sense, an encoder network is a data compression structure. In various computer vision tasks, this section is generally adopted directly from well-known classification networks, therefore also called the backbone. One of the practices is to keep encoder weights if it already has trained on a large dataset since the convolutional layers are expected to learn to extract the most representative features for the classification task, therefore can convey enough information for any other task. This approach is called Transfer Learning.

The selection of backbone depends on performance factors such as size in terms of parameters, inference speed and accuracy. ResNet-50, ResNet-101, ResNext-101 [4], DenseNet-161 [25], MobileNet-V3 [26] are the most popular choices among depth estimation tasks.

ResNet was a game-changer in the image classification domain due to its conspicuous depth advantage. Although it seems like just a deeper version of its closest competitor VGG-16 [27], repeating a few stages over in depth was not the solution to their success. Their study has shown increasing solely the depth even cripples the classification performance due to a phenomenon called vanishing gradients. Since differentiable learning methods calculate weight changes with respect to its derivative against the resulting loss value, the gradient values become smaller towards the initial layers. In a sufficiently deep architecture, this behavior results in gradients becoming insignificant no matter the high loss value, thus

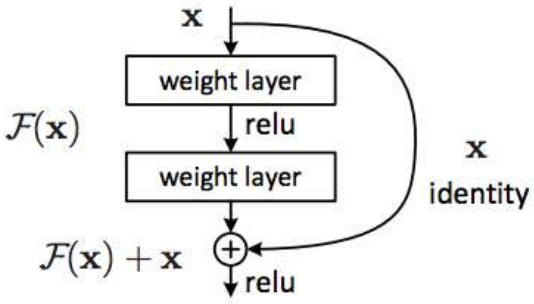

Figure 18: The skip connection in a residual block, taken from [3]

weights of initial layers do not update. Assuming a random initialization, initial layers are quite likely to not perform in favor of the task. That is why not being able to learn appropriate weights in early stages damages overall performance in a deep architecture.

As can be seen in Figure 17 on top, ResNet has high-way arrows connecting the start of each block to its end. Those arrows denote residual connections that let a CNN encoder get significantly deep without degrading performance. Figure 18 shows how residual connections work in detail. It allows any block, referred to as res-block, to learn to represent identity operation which makes the network possible to get deeper without losing updates on initial layers. This practice allowed ResNet architecture to become drastically deeper and consequently more accurate on the image classification task. The study also showed how making the network aggressively deeper after a certain point decreases the performance due to overfitting.

ResNext is an improvement over ResNet as its name suggests. The main contribution of ResNext is going wider rather than deeper for performance improvement. Making the network wider was already put into use before in AlexNet, however, such a choice was made for being able to split a network into two GPUs as a trick to overcome the memory constraints of those days. Whereas ResNext converts each ResNet block into its wider equal in terms of the number of floating-point operations (FLOPs) with the sole purpose of increasing classification accuracy. FLOPs and multiply-accumulate operations (MACs) are popular performance metrics for

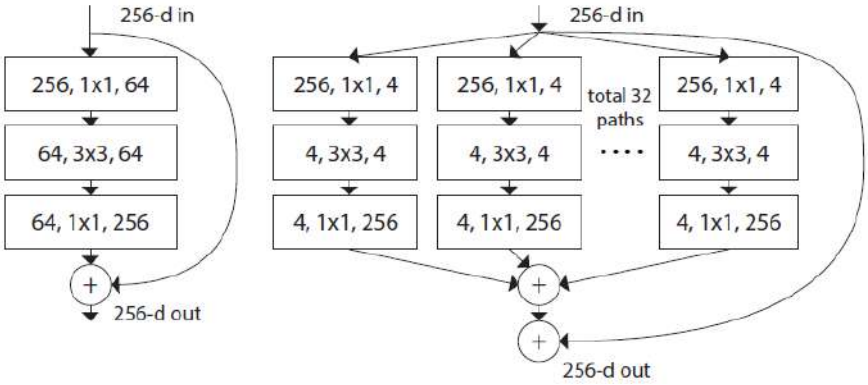

Figure 19: The difference of ResNext block and ResNet Block with the quivalent number FLOPs. ResNet block on the left, ResNext Block on the right. Taken from [4]

denoting the computational complexity of deep learning design choices. Although those metrics are inversely-correlated with inference speed, they are not the absolute marker since the inference speed also massively relies on the parallelizability of those operations.

The formula above shows how to calculate FLOPs for a single convolution,

$$FLOPs = 2 \times C_{input} \times k^2 \times C_{output} \times H \times W \tag{15}$$

where C_{input}, k, C_{out}, H, W, denotes number of input channels, kernel size, number of output channels, height and of the output feature map respectively.

Similarly the formula for calculating MACs for a single convolution,

$$MACs = C_{input} \times k^2 \times C_{output} \times H \times W \tag{16}$$

Figure 19 shows how they redesigned ResNet blocks into a multi path setting. They use a 1×1 kernel convolution for decreasing the depth channel from 256 to 4. Although this is considered a quite aggressive channel bottleneck for a regular CNN architecture, it makes up for the representation space by reproducing the operation 32 times in parallel. Later, all 32 feature map groups having only 4 channels go through a regular 3×3 kernel. Now filter size for feature map groups has dropped significantly in comparison to ResNet, from 64 times 3×3 to 4 times 3×3. A smaller filter size allows the network to go wider, and more paths, while

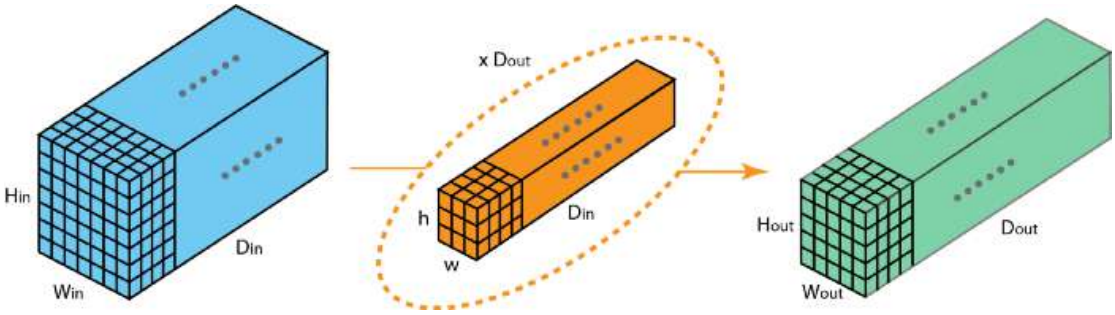

Figure 20: A regular convolution with multiple input channels, taken from [5]

preserving FLOPs. Finally, all feature map groups are projected to 256 channels via 1×1 kernels and summed yielding the same channel size.

The study further shows that going wider by projecting feature maps into multiple shallow feature maps for further processing corresponds to using group convolutions. Figure 20 shows a regular convolution operation and what a kernel corresponds to when the input feature map has multiple channels. A single step of a convolution operation calculates a single pixel on the output features. Figure 21 visualizes the group convolution. Instead of using filters of the same depth as incoming feature map channels, group convolutions groups channels with a smaller number. Later, each of those channel groups goes through convolutions independently and fuses at the end with a channel-wise concatenation. Intuitively, the feature maps are sliced through the channel dimension and then the slices are treated as independent feature maps.

MobileNet [28], a very popular backbone choice for real-time tasks, takes group convolutions a step further and treats each channel independently. They call it depth-wise convolution and it is a special case of group convolution where the number of groups is equal to the number of channels. The approach remarkably speeds up the convolution operation by decreasing the number of operations. In contrast, treating each channel independently removes inter-channel information transition. For recovering that, every depth-wise convolution is followed by a point-wise convolution. Figure 22 illustrates the procedure, where point-wise convolution basically is a 1×1 kernel convolution.

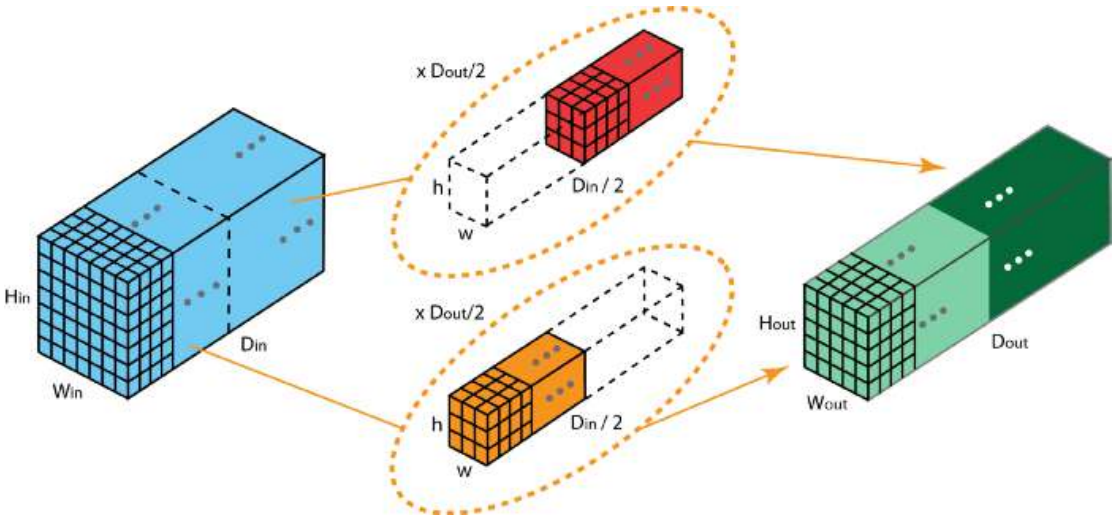

Figure 21: A regular group convolution of group size two with multiple input channels, taken from [5]

MobileNet V2 [7] and V3 [26] introduced further convolutional techniques to improving on depth-wise convolutions. The model changes the classical approach of increasing channel size over deep layers. As Figure 23 shows, each block first increases the channel size by 1×1 kernel convolutions. Following that, the channel augmented feature maps go through depth-wise and point-wise convolution respectively. The final point-wise convolution reduces the channel size to its initial condition and the skip connection binds the layer to the beginning of the block with the addition operation.

This structure is the opposite of the residual block strategy which follows wide-narrow-wide feature maps. Making middle feature maps wider allows harvesting depth-wise convolutions intensely for inference speed. Another technique they used is called squeeze-and-excitation optimization. During depth-wise convolution operation, they average pool every channel to represent a single superpixel. This superpixel vector then goes through a sigmoid activation to range from 0 to 1. Finally, this vector is applied by channel-wise multiplication to incoming feature maps. Intuitively, this allows the network block to assign a significance score to each channel, acting as an attention mechanism.

DenseNet [25], aims to build on skip connections by densely connecting each convolution operations in a block as Figure 24 shows. Dense connections are

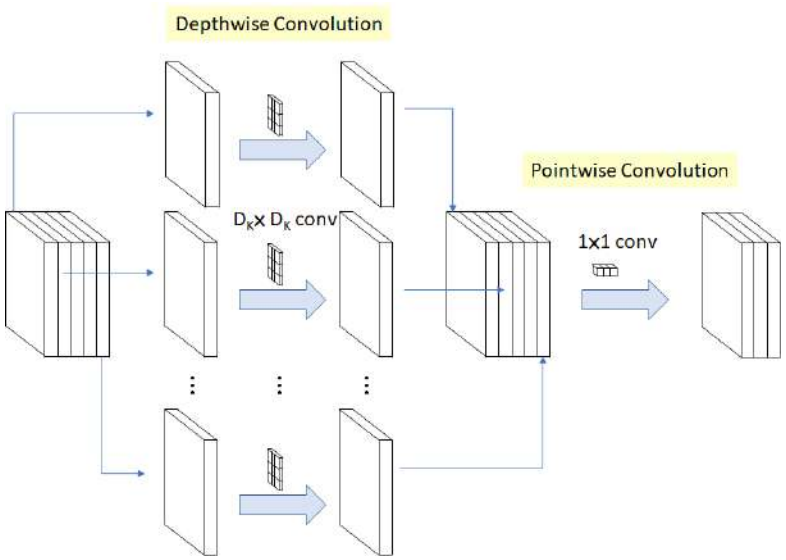

Figure 22: The depth-wise convolution and the point-wise convolution in MobileNet demonstrated, taken from [6]

made via channel-wise concatenation. Using multi-layer connections makes existing channels to forward propagate and results containing richer features throughout the network. This results in slimmer convolution layers, therefore an improvement in computational efficiency. In order to keep the layers slim after multiple concatenations, they employ 1×1 kernel convolution to reduce channel size and then use 3×3 kernel convolution on top of it to further reduce it to initial channel size.

Another advantage of dense connections appears during learning. Likewise the ResNet skip connections, dense connections improve gradient flow. Even better than ResNet, which can receive direct gradients from the following block, each DenseNet convolution layer receives gradients from every upcoming block. This helps the error signal coming from the end of the network to reach every layer without losing any power which creates a strong "implicit supervision" as the paper calls it.

At this point, each CNN classification network, backbones in our case, has improved accuracy, speed, and size by proposing new architectural designs and advanced techniques. Different than those studies, Efficient Net examines the effects of CNN parameter tuning over classification accuracy. First, the study

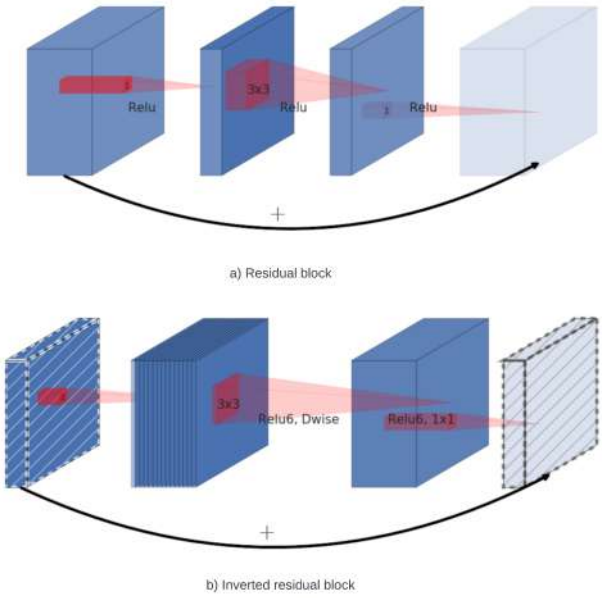

Figure 23: The comparison of standard residual block on top and the inverted residual block in MobileNet bottom, taken from [7]

shows going deeper, wider, and increasing input resolution in a CNN architecture all alone improves overall classification performance as shown in Figure 25. The main contribution of the paper is to provide a guide on how to efficiently scale those three hyper parameters under a size constraint. Instead of arbitrarily choosing the number of layers, channels, and input size, they propose the following compound scaling method to balance a CNN;

$$\begin{aligned} depth : d &= \alpha^{\phi} \\ width : w &= \beta^{\phi} \\ resolution : r &= \gamma^{\phi} \\ s.t. \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\ \alpha \geq 1, \beta \geq 1, \gamma &\geq 1 \end{aligned} \tag{17}$$

where ϕ represents the resources at hand and α, β, θ are all constants to be selected according to the proposed ratio. The study redefines CNN parameter decisions as an optimization problem so that the above-mentioned constants can be found with a grid search on a small setting. Later, they scale this setting by increasing ϕ

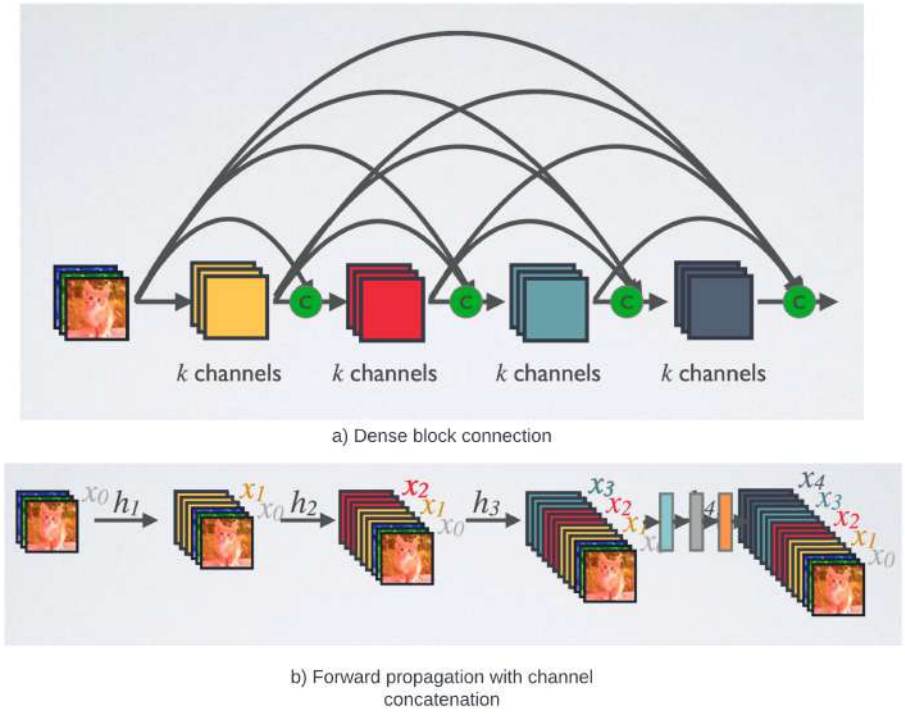

Figure 24: The representation of a dense block and the forward propagation of the network visualized, taken from [8]

coefficient. Simply, their compound scaling method shows a decision to go deeper in a CNN network results in linear growth in complexity, while increasing the channel size or feature resolutions results in quadratic growth.

EfficientNet [9] allocates resources with respect to the compound scaling method and structures existing layer strategies from previous works. In detail, it uses inverted residual blocks from MobileNet and leverages squeeze-and-excitation optimization. The table 1 shows the baseline setting EfficientNet-B0 according to $\phi = 1$, therefore the ratio yields $\alpha = 11.2$, $\beta = 1.1$, $\gamma = 1.15$. By increasing ϕ value they scale this architecture up to EfficientNet-B7.

For tasks like Monocular Depth Estimation, those classification networks are adopted as backbones or encoders in auto-encoder terminology. The features extracted with selected encoder architecture guide the decoder section to form desired output in image format. This output might be a depth map of the input size for the Depth Estimation task, an RGB image for a denoising task, or a bigger RGB image for the super-resolution task.

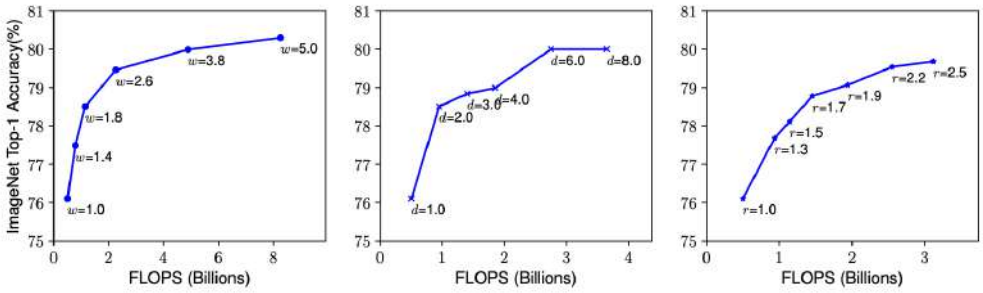

Figure 25: The effects of depth, width and input resolution on accuracy, taken form [9]

Table 1: EfficientNet-B0 architecture.

Stage	Operator	Resolution	Channels	Repetition
1	Conv3x3	224 × 224	32	1
2	MBConv1, k3x3	112 × 112	16	1
3	MBConv6, k3x3	112 × 112	24	2
4	MBConv6, k5x5	56 × 56	40	2
5	MBConv6, k3x3	28 × 28	80	3
6	MBConv6, k5x5	14 × 14	112	3
7	MBConv6, k5x5	14 × 14	192	4
8	MBConv6, k3x3	7 × 7	320	1
9	Conv1x1-Pooling-FC	7 × 7	1280	1

Decoders use upsampling strategies to form image format outputs out of feature maps. Figure 26 shows two non-learnable upsampling methods. The Nearest Neighbours approach simply repeats a pixel value on the target feature maps while preserving the aspect ratio. Similarly, bilinear upsampling interpolates a pixel value on the target feature map into the neighboring pixel. Those methods are also used for upsampling feature maps and combined with convolution operations become learnable methods. Assuming the encoder compressed the necessary information into those feature maps, upsampling stages of the decoder are designed to extract this information qualitatively.

Another learnable solution for upsampling is to use transpose convolutions [29]. Figure 27 shows a 2×2 feature map upscaled to 3×3 with a transpose convolution. The operation is also referred to as Up-convolution. Stride must be selected carefully in this method since certain areas on the output map are exposed to this operation multiple times. As the stride of the example figure is

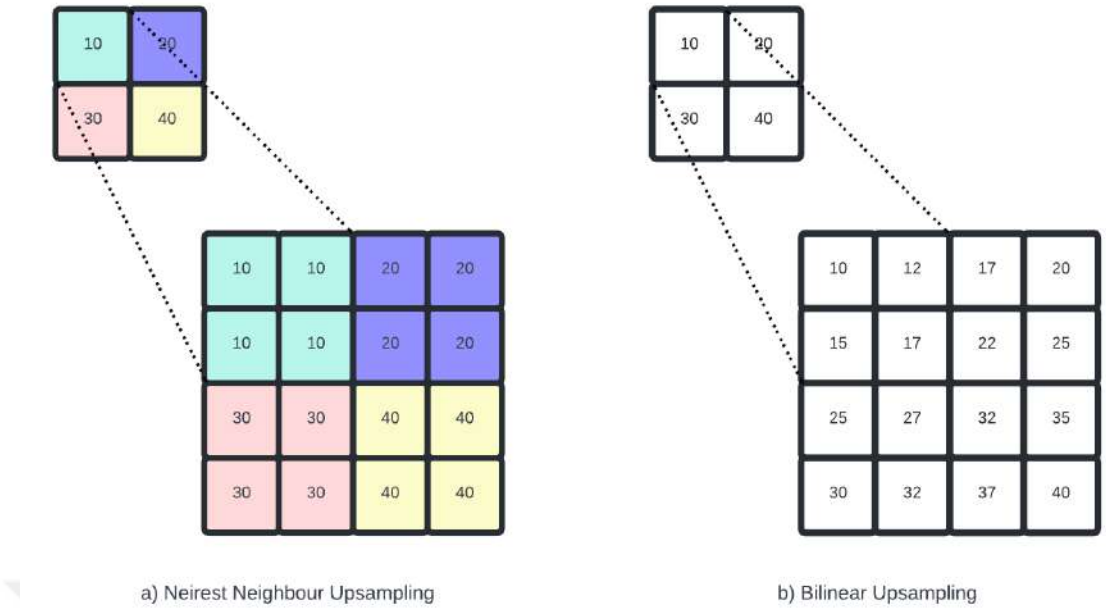

Figure 26: Upsampling a 2×2 matrix to double its size, a-) shows nearest neighbour upsampling and b-) shows bilinear upsampling

1 unit, a cross-like structure appears on the output map. The second row and column of the output map are exposed to the operation two times, meanwhile, the middle pixel is exposed four times. This decreases the visual plausibleness of the image due to checkerboard artifacts.

U-Net [10] influenced the auto-encoder structuring widely. It is designed for biomedical cellular segmentation tasks and its structure is widely adopted in many image-to-image translation tasks. The main contribution of this architecture relies upon its usage of skip connections. U-Net has a symmetrical structure when its encoder is compared to its decoder. This allows the network to connect encoder and decoder layers of the same resolution by depth-wise concatenation. The long-range skip connections help the network to restore image structure from the compression stage and enable the decoder to manipulate upsampled feature maps easily during decoding for the aimed purpose. Each decoder layer uses transposed convolutions for up-scaling which is followed by consecutive standard convolutions. Figure 28 demonstrates overall structure of the U-Net architecture.

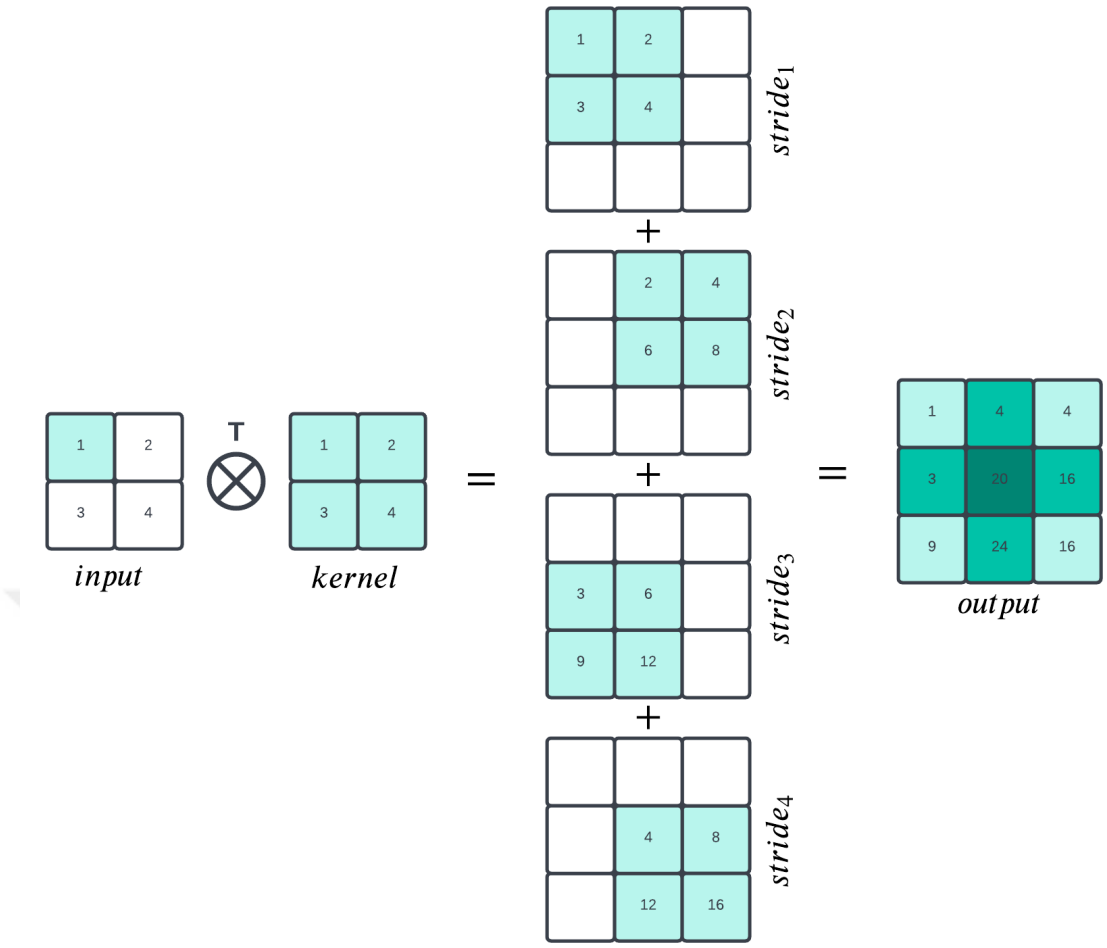

Figure 27: Upsampling a 2×2 matrix to 3×3 matrix via transpose convolution

2.7.5 Loss and Metrics

Models update their internal parameters with respect to a loss function during learning. The training aims to find parameters that minimize chosen loss function. The main interest of this thesis, namely monocular depth estimation, is defined as a regression problem. Given an RGB image, the model tries to estimate depth values for each pixel. The most straightforward losses are $\mathcal{L}_1$ and $\mathcal{L}_2$ in the regression domain. These two distance metrics are used for penalizing how much the model predictions deviate from the ground truth data on the pixel level.

$\mathcal{L}_1$ loss, namely Mean Absolute Error(MAE), averages the absolute differences of data points. Formally MAE is calculated as follows,

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - y_i^*| \tag{18}$$

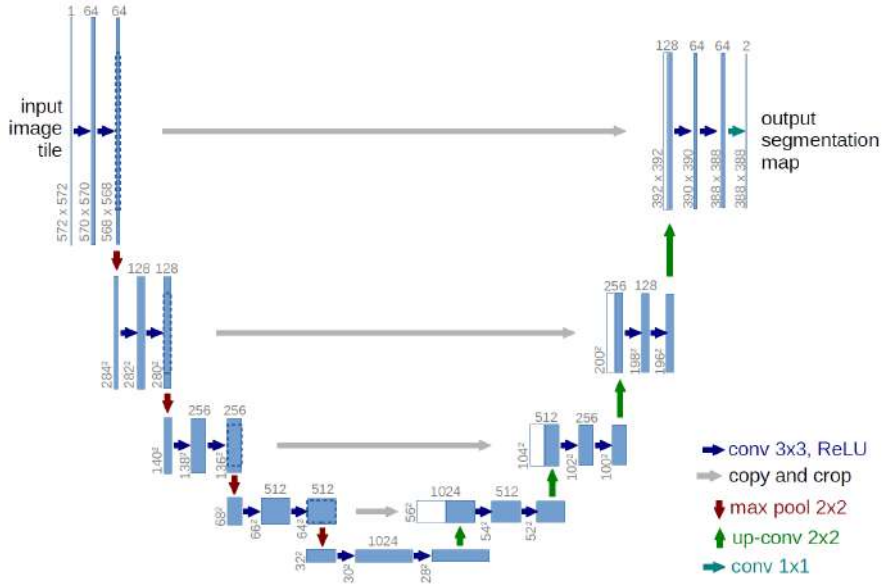

Figure 28: The U-Net architecture, taken from [10]

where N is the number of data points, y_i is the ground truth value and $\hat{y}_i$ is the model prediction. Using the absolute value operator for the differences covers negative differences that might arise during evaluation. Therefore, the cases where prediction is greater than the ground-truth value do not distort the average value. However absolute value operation is not differentiable, thus might complicate the differentiable learning setup.

$\mathcal{L}_2$ loss, namely Mean Squared Error(MSE), uses another method for properly taking care of negative differences. Similar to MAE, it averages deviations but squares the differences instead. The formula of MSE is,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^*)^2 \tag{19}$$

where N is the number of data points, y_i is the ground truth value and $\hat{y}_i$ is the model prediction similarly. Using squares of the differences for averaging does not only covers for negative differences but also makes it differentiable. Due to the nature of squaring operation, an additional behavior appears with MSE. Higher differences results in quadratically higher errors. As a consequence, $\mathcal{L}_2$ loss weighs bigger differences much more on the resulting average, penalizing big errors harshly. Any outlier on the dataset might distort the metric. Conversely,

any error in the decimal scale will be neglected.

Root Mean Square Error (RMSE), takes the root of MSE differences to overcome the above-mentioned robustness challenge. The formula for calculating RMSE is,

$$RMSE = \sqrt{\frac{1}{|N|}\sum_{i}^{N}(y_i - y_i^*)^2} \tag{20}$$

Adding the root operation also recovers original unit on resulting error. RMSE is also used widely as a metric to evaluate model performance.

Monocular Depth Estimation is a pixel-wise regression task. However, using RMSE is not a great loss for this task as the work of [18] has shown. Scale ambiguity may distort linear metrics such as RMSE due to harsh penalties in larger scenes. Likewise, the error of a very distant point prediction can shadow overall performance in a single instance. For instance, a misprediction of a sky pixel in an outdoor scene or a window pixel in an indoor scene will be so large that accurate predictions won't get enough credits. In order to prevent such cases, the log operator is applied both to prediction and the ground truth yielding the equation,

$$RMSE(log) = \sqrt{\frac{1}{|T|}\sum_{y\in T}\|\log y_i - \log y_i^*\|^2} \tag{21}$$

The Berhu loss [30] is a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$ norm where one or the other is applied depending on a threshold c as follows,

$$\mathcal{B}(x) = \begin{cases} |x| & |x| \leq c, \\ \frac{x^2+c^2}{2c} & |x| > c. \end{cases} \tag{22}$$

$\mathcal{L}_2$ loss penalizes the pixel-wise deviation of the output from the ground truth. The global scale differences occurring among scenes may result in inconsistent supervision. For the depth estimation task, this implies $\mathcal{L}_2$ loss penalizing large-scaled scenes more than the small-scaled scenes irrespective of the network performance. Motivated by that, [18] introduced an additional regularization term for prioritizing the relative depth relations.

Given that y_i and y_i^* are the outputs of network and ground-truth respectively, the scale-invariant loss is calculated as follows,

$$D(y, y^*) = \frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}\sum_{i,j} d_i d_j \tag{23}$$

where $d_i = \log y_i - \log y_i^*$ and i, j denotes the pixel pairs in the output. The first term is well known $\mathcal{L}_2$ loss. The latter is the regularization term which favors an output pixel being in the same direction as other output pixels. This term credits the prediction if the predicted pixel is consistent with other pixels by decreasing the loss accordingly. Consequently, the loss aims to scale the global depth for penalizing the depth differences fairly, thus named scale-invariant error.

Finally, the threshold error is defined as,

$$\%\ of\ y_i\ s.t.\ max(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}) = \delta < threshold \tag{24}$$

, where threshold values are set to 1.25, 1.25^2 and 1.25^3. Those δ metrics signifies three different levels of descending sensitivity on predicting accuracy. Since the metric unit denotes the percent of the accurate pixels, higher values denotes better performance of prediction. Precisely, this metric shows the proportional resemblance of the ground truth depth maps with the predictions.

CHAPTER III

LITERATURE REVIEW

3.1 Introduction

Depth Estimation is a widely researched computer vision topic due to its value in the rising industries. Many research [31, 32, 33, 34, 35] have investigated how to extract depth information using stereo images and video sequences. Nevertheless, as discussed in Section 2, these methods have peculiar problems meanwhile introducing additional cost and complexity into the data collection process. Considering a human can easily estimate depth from a single image, learning methods conceptually must have the same capacity for achieving this task when an appropriate loss function and a dataset are provided. As outlined in Section 2.6, humans achieve this task by exploiting high-level information present in the image. Extracting this information called monocular cues is rather complicated since it involves general scene understanding and semantic knowledge of the world. Recent advances in deep learning for computer vision have shown to be capable of extracting those complicated relations in image scenes and dominated the monocular depth estimation research. In this section, we review previous works tackling the Monocular Depth Estimation problem in various strategies and substantial challenges of the task.

3.2 Deep Learning in Monocular Depth Estimation

As a consequence of GPUs becoming cheaper, the democratization of computing led deep learning to become a popular choice to tackle computer vision problems. In 2012, AlexNet [36], one of the most significant advances in image recognition for its time, has encouraged using deeply stacked CNNs. Consequently, using deep architectures has shown great promise for revealing this complicated scene information underlying a given image. Following this, VGGNet [27] and ResNet [3] further improved image classification performance and introduced new strategies for deep architecture structuring in vision problems.

Depth prediction, being another challenging vision task relying on scene understanding, studied applications of CNN extensively. Many of the studies [30, 37, 38, 14, 39, 40, 41, 42] have adopted proven classification architectures as a backbone to their depth prediction designs or followed their practices. The first adaptation of deep CNN on this task is performed by [18].

They proposed a two-stage model architecture, first for learning coarse-scale information and second for capturing fine-scale information. The coarse-scale network processes the RGB input initially by 96 large 11×11 kernel convolutions. The output features then pass through 4 additional convolution layers with smaller kernels. The last two layers are fully connected layers where the last one represents a flattened single-channel depth map. The coarse-scale network is designed to learn this depth map to be low resolution and capture the global depth information of the scene. Following that, the fine-scale network is structured to fuse this global depth information again with the RGB input to refine the details of the depth map. To achieve this, the fine-scale network stacks three feature extraction layers, where the second one fuses the low-resolution depth map to the input features via concatenation. Following the practice of [36], the feature maps are down-scaled with max-pooling operation between feature extraction layers, and training data is augmented with known transformations.

In contrast to well-known $\mathcal{L}_1$ and $\mathcal{L}_2$ losses, [18] came up with Scale-Invariant

Error for training the network. Their deep learning solution significantly outperformed former feature engineering methods for depth extraction since the architecture can extract those features from data for less controlled environments. The work compared their performance to recent methods in NYU Depth [43] and KITTI [17] datasets on various error metrics which were later adopted by following studies and set a milestone for the MDE benchmark of today. Along with straightforward image-to-image metrics such as Absolute Relative difference and Squared Relative difference, those metrics also involve a Threshold error and Root Mean Squared Error in several other scales.

In 2015, another study wanted to adopt the emerging success of learnable convolutions. Rather than feeding the whole image to the convolutions, Liu et al. [44] proposed fusing this approach into Radial fields [45] where the input image is over-segmented into superpixels. Each of those superpixels later is treated as centroids of image patches to be cropped, which are later processed through convolution layers connected to fully connected layers for expressing depth. Finally, the neighboring superpixels are compared with a pairwise network which enforces similar superpixels to have close depth estimations with the help of a custom energy-based loss function.

Subsequently in 2016, Laina et al. [30] proposed a novel fully convolutional architecture influenced by [3]. Besides having more layers than [18]., their model leveraged Batch normalization operations between each convolution block which eases convergence. Moreover, the skip connections preventing the vanishing gradient problem allowed the architecture to fully benefit from deeper layers in terms of accuracy. Another important contribution is the final depth map construction section of the network. The fully connected layer representation of a flattened depth map in the previous work is replaced by upsampling blocks, resulting in a higher regression performance in the image domain without the parameter expense of fully connected layers.

The model uses two kinds of upsampling blocks; Up-convolution Block and

Up-projection Block. The first one stacks 2×2 un-pooling layer with a regular convolution and a ReLU activation yielding a two times resolution increase at each block. The latter follows the idea of residual blocks of ResNet and introduces a skip connection with a convolution to the Up-convolution block. Additionally, their findings showed reverse Huber Loss [46], *i.e.*, berHu loss, performed better during training of the model.

Yin et al. [40] introduced an additional loss term to their supervised training pipeline, aiming to leverage geometric relations that exist in the depth map. As geometric supervision, they set an angular surface normal constraint. From the ground truth depth map, they reconstruct a 3D point cloud assuming a pinhole camera projection. Later, this point cloud is sampled for a surface normal supervision technique. A naive surface normal supervision strategy, where three 3D points surface normal vector is used for angular loss directly, can not be applied to the acquired point cloud since regular surface normals only provide local information and such a process is vulnerable to noise present in depth maps. Thus, their strategy samples triplet points with non-co-linearity and range restrictions and defines the normal to those surfaces as virtual normals. The study shows applying angular loss to randomly sampled virtual normals assures a noise-robust and effective learning scheme.

Gan et al. [47] applies a stereo matching algorithm as a pre-processing to the LIDAR dataset to generate dense ground truth depth maps.

JH Lee et al. [41] proposed local planer guidance layers on the decoding section of their architecture as an alternative to the Nearest Neighbourhood Upsampling strategy. Instead of upsampling a given feature map with a $k \times k$ kernel convolution, they proposed local planar guidance layers following a 4D plane representation assumption for reconstruction. Thus instead of k^2 parameters, it is subject to only 4 parameters for upsampling meanwhile expressing more information for the geometric needs of this task. Specifically, each pixel of the 4-channel input feature map is projected as a $k \times k$ local plane patch via the ray plane intersection

formula. Input channels represent a 3D unit vector and a perpendicular distance.

3.3 Unsupervised and Semi-Supervised Approaches

The most prevalent problem with deep learning solutions is how data-hungry the methods are. The strength of low bias expressivity comes with an abundance of hyper parameters to tune. Although many deep learning methods already apply some amount of data augmentation to improve prediction performance, it is not even closely compared to using samples from the natural distribution of the problem during training. Given the cost of acquiring depth maps with LIDAR, quality ground truth depth maps are limited. Thus, some work investigated the use of Semi-Supervised and Unsupervised setups to outperform state-of-the-art solutions.

Godard et al. [37] surpassed the existing supervised regression solutions on the KITTI dataset by utilizing stereo matches instead of ground truth LIDAR depths during training. Note that the stereo matches are only provided during training as guidance and inference are done with a single image, thus the study compares its performance on the Monocular Depth Estimation benchmark. Given that depth can be extracted from disparity maps with a known calibration, they proposed a model that learns disparity while transforming left-right correspondences from one to the other. The straightforward setup to achieve such a disparity model is to learn to transform the left image to the matching right image. However, this naive setup ends up learning the disparity transformation aligned with the right image, resulting in a depth prediction of the right image given left. To overcome this issue, the model is conditioned to output a disparity map that retrieves the initially provided image when applied to the right image. Although this reconstruction loss was achieved to retrieve depth aligned with the provided image up to a point, it resulted in texture copy artifacts on the output depth map, especially around object corners and depth discontinuities. They solved the issue by introducing a left-right consistency setup. Instead of learning a single

disparity map that recovers a given image, the model outputs disparity maps for left and right images distinctively. The left-right consistency loss guides the model to learn to transform both left images to right and right images to left. This approach is shown to improve overall depth estimation aligned with the given image significantly.

In addition to the scarcity of metric depth maps, LIDAR outputs are sparse. Due to the mechanical limitations of the device, it can only capture less than half of the scene pixels in the depth map. This translates to less than half of the dataset being utilized during training, providing weak supervision. Kuznietsov et al. [38], presented a solution to use binocular information like [37] but additionally incorporate metric depth maps as in a regular supervised setting. Their model outputs an inverse depth map, which can be also used to project left image pixels to right image pixels with a given baseline and focal length. During training, if metric depth information is available on the ground truth LIDAR depth map for the predicted pixel, supervised berHu loss applies. For all other pixel regions, the predicted depth map is judged by how well it warps binocular RGB image pairs into each other. Thus, the dense information available in the dataset is also utilized along with metric depth maps.

3.4 Discrete Depth

Fu et al. [39] on the other hand, has transformed this supervised problem from regression to classification and achieved significant improvement across all metrics. Motivated by the relative convergence advantage of classification over regression, the method discretized depth into predefined sub-intervals transforming the problem into an ordinal regression problem. As the concern with other studies, the larger depth values tend to dominate loss functions although they provide less rich information. Accordingly, the study chose to discretize the depth values with respect to log space, *i.e.*, space increasing discretization (SID), instead of straightforward uniform quantization. Thus, the strategy allows the loss function to pay

more attention to closer depth pixels and condense larger depth values into a lower number of classes. Although the problem at hand seems like a standard $softmax$ multi-class classification problem, such a solution discards a strong ordinal correlation of depth values to be predicted. Therefore, the study introduces an ordinal loss that recognizes depth intervals as a well-ordered set.

Just recently, Adabins [42] improved upon quantizing the depth space and became the state of the art across all quantitative metrics. The previous work can be slightly improved by learning an appropriate depth quantization for a given dataset. However, the paper discusses that optimized depth intervals deviate not only for a dataset but also for each image. Instead of representing depth in pre-determined intervals, referred to as bin widths in the paper, the method adapts those bin widths dynamically with respect to the input. Moreover, the output of the network expresses depth as a linear combination of bin center values for clearing wave-like artifacts occurring in [39]. Adabins is the first work to incorporate vision transformers (ViT) [48] into a pixel-wise depth prediction task. Since predicting bin widths according to the input ultimately relies on the receptive field, Adabins leverages a small version of ViT called miniViT module. One of the heads of that module outputs bin width predictions while others are used as 1×1 convolution kernels later to be applied as Range-Attention Maps. In order to train the bin width prediction head, the Chamfer Loss [49] is used for quantifying the difference of bin center distributions.

3.5 Real Time Monocular Depth Estimation

Most of the promising applications of monocular depth estimation involve interactive environments and thus rely on inference speed. Robots navigating in a room, autonomous cars driving on roads, and augmented reality devices running apps all require real-time feedback for functioning appropriately. Although the aforementioned studies achieve satisfying results on many quantitative metrics and plausible depth maps, they are not subject to any time constraints.

CReaM [50], is the first work to achieve real-time performance on depth prediction using a deep architecture that is deployable on a mobile device. Larger and deeper networks are known to perform better on vision-based tasks, however, they come up with a higher number of parameters to train and slower inference time. CReaM aims to combine best of the both worlds with a knowledge transfer training pipeline. The work follows a teacher-student setting, where a deep and large model supervises a condensed real-time model via minimizing intermediate feature distances. Another proposed knowledge transfer strategy of the work includes a direct transfer of the final layer to the condensed real-time model. Both models adopt encoder-decoder network architecture with a solver layer at the end that predicts a depth map from the features maps coming from the decoder. Depth predictions of the models are guided with $\mathcal{L}_2$ loss on ground truth depth maps coming from LIDAR. The knowledge transfer on the other hand occurs in the penultimate layers of the networks. The feature maps coming from the decoders are treated as n-dimensional embedding vectors where an $\mathcal{L}_2$ loss learns to minimize their distances. The same procedure also applies to the strategy of transferring the solver layer. Once embedding vectors of both models get closer in the n-dimensional space, the solver layer from the bigger model is plugged into the decoder of the condensed model, and additional training is applied for parameter fine-tuning. The study shows both strategies surpass the performance of the condensed model without knowledge transfer in terms of accuracy and the condensed model achieves up to 300 fps of inference time.

Atapour et al. [51] is another study achieving real-time performance by making use of domain transfer on synthetic depth data. They handle monocular depth estimation problem as an image-to-image translation task and utilize adversarial loss [52] along with cycle consistency loss [53] for unsupervised learning. A naive approach is to directly train a depth predictor network with synthetic scenes and their corresponding depth. However, the inference performance of such a predictor may suffer in real-world applications due to dataset bias of the emulated scenes. Indeed, synthetic scenes rendered with the gaming application are easily distinguishable from real-life scenes and may not generalize to real-world depth maps. In order to solve this data domain variation, they use the recently proposed style transfer technique [54] for training a generator that translates real-world images into the synthetic domain. Specifically, the inference pipeline involves two successive generators, first transforming a real RGB image into a synthetic RGB image, and the latter transforming the synthetic RGB image into the depth map. With the help of this initial generator, the second generator trained as a depth predictor only on synthetic data can be used. Proposed generators follow a lightweight U-Net architecture and benefit from the generous availability of synthetic data.

Nekrasov et al. [13] train their model jointly for segmentation and depth prediction. Their proposed architecture is a mix of [7] and Lightweight RefineNet [55], where the first one is the encoder and the last one is the decoder. Their model follows a U-Net architecture, but additionally, their model applies a slightly modified version of a recently proposed Chained Residual Pooling Block (CRP) to each skip connection. Their slight modification aims to reduce the number of operations significantly for the last CRP block processing the highest resolution feature maps by replacing a 1×1 convolution with its depth-wise equivalent. The objective of the CRP Blocks is to convey contextual information from encoder to decoder via fusing multiple residual features along with the information coming from the pooling layer.

Chiu et al. [56], also follows an encoder-decoder architecture and adopts [7]

as its encoder network due to its light-weight structure. As the decoding scheme, their network has 4 levels of decoding blocks each predicting both segmentation and depth maps on different scales. At each block, feature maps are up-scaled with pixel shuffle strategy and processed by two independent convolutional layers; one predicting a depth map and the other predicting segmentation info. The up-scaled feature maps are later passed to the next decoding block for doing the same task on a bigger scale. Their loss function evaluates the model on all of these 4 scales over both segmentation and depth performance. They showed that involving segmentation task as a teacher to depth estimation objective has overall improved their results in terms of RMSE.

MiniNet [14] is another work trying to reduce inference time with a lightweight model and achieve this with an unsupervised setting. Their pipeline involves three models, two PoseNets sharing the same weights and the proposed DepthNet. The pipeline proposed receives three frames of a video sequence of a scene where the first and the last frame are fed to PoseNets and the one in the middle is fed to DepthNet. The two PoseNets are guiding the DepthNet during the training by exploiting SfM information in an unsupervised setting. Since DepthNet receives a single image during training, it can be detached from the pipeline for monocular depth estimation inference. DepthNet model is lightweight due to the efficiency of Inverted Residual Blocks inspired by [7] used in the recurrent encoder module and their proposed upsampling blocks on the decoder section.

CHAPTER IV

EXPERIMENTAL STUDY

4.1 Introduction

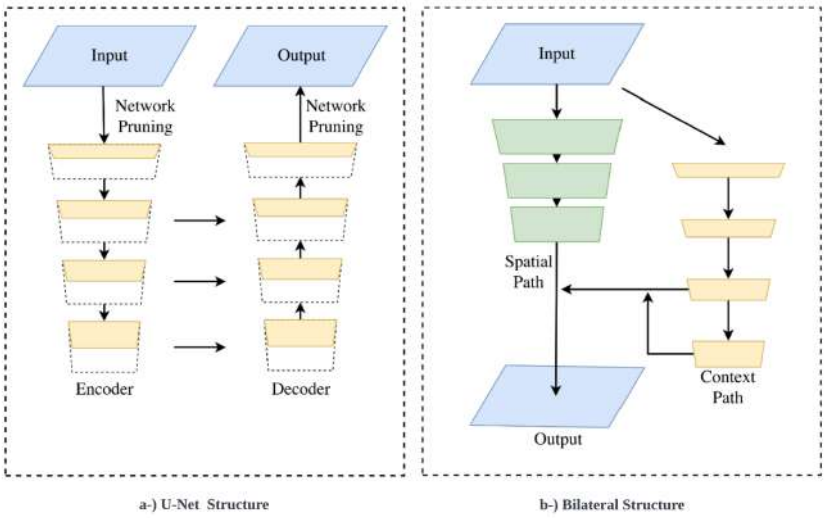

Figure 29: The structural differences of U-Net and Bilateral Networks taken from [11]

Monocular Depth Analysis is a fundamental Computer Vision field, which has been studied extensively on different supervision schemes and with different constraints, *i.e.,* inference time, size, and accuracy. The accuracy and throughput trade-off for this task resulted in two different benchmarks for the same dataset, namely Real-Time Monocular Depth Estimation on KITTI and Monocular Depth Estimation on KITTI.

Shortly after its success in biomedical segmentation, the U-Net structure influenced many fully convolutional networks related domains. Image to image translation and other segmentation networks followed their practice of using a balanced encoder-decoder structure with skip connections. Lately, BiSeNet [11] showed that using a bilateral network instead of upsampling with an additional decoder, speeds up the inference time without compromising the accuracy of the real-time

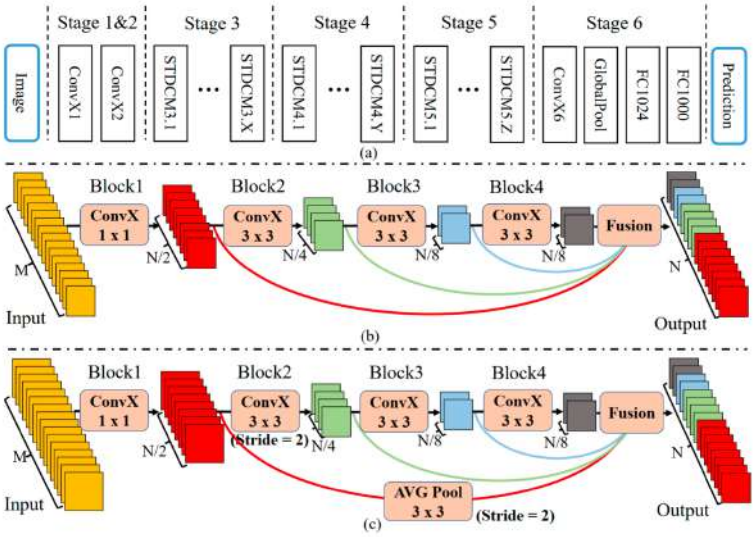

Figure 30: The structure of the Short-Term Dense Concatenation feature extractor taken from [12]

semantic segmentation domain. Figure 29 shows the comparison of U-Net and Bilateral structures.

In a regular U-Net-like structure, an encoder processes the image on multiple scales by deeply stacking convolutional layers. At each step, the information captured by the network goes from coarse to fine with the help of the increasing receptive field. The final feature map compression is later interpreted by the decoder with a very similar process but in reverse. A bilateral network aims to separate this process efficiently by using two paths addressing different needs. One of them, namely the Spatial path, processes low-level features, and the other, the Context path, provides high-level information.

The Spatial path does not need a large receptive field since it has to capture only the high-frequency details. This means the path can work shallow which allows it to leverage high-resolution feature maps without the computational burden of deep layer stacking. On the other hand, the context path needs this receptive field for recovering high-level information. Given that the spatial path takes care of high-frequency patterns and outputs high-resolution feature maps, the context path has the luxury to work on aggressively down-sampled feature maps. Processing down-sampled feature maps, allows the network to achieve this receptive field

without the need for a very deep CNN design resulting in further speed improvements. Finally, the high-resolution depth maps containing the spatial information are fused with context information by a Feature Fusion Module (FFM).

Short-Term Dense Concatenate Segmentation network (STDC-Seg) [12] builds on top of BiSeNet and improves upon its structural redundancies. Moreover, instead of using an Object Classification network as a backbone, they propose a Dense-Net [38] like feature extractor specifically designed for the segmentation network as shown in Figure 30. Each ConvX block in the figure denotes a succession of Convolution, Batch Normalization, and ReLU [39] activation respectively and the bottom indicators show the kernel size. Different than a typical classification network, the STDC feature extractor does not employ a high number of channels towards the end of the network.

This study aims to investigate a real-time model architecture that can maintain on-par quantitative results with high precision depth estimation models that are not subject to any time constraints. In order to achieve high accuracy while maintaining real-time performance, we investigate the structural advantages of above mentioned Bilateral Networks on the monocular depth estimation problem. We transform this real-time segmentation architecture STDC-Seg into a depth regression network, then we train the model on the KITTI dataset following the practice of Eigen split for comparing the performance of our approach with state-of-the-art models. Later, we tailor the backbone and attention layers of the architecture to better suit the needs of MDE. Finally, we compare our modified version both on standard and real-time constrained KITTI Benchmarks.

4.2 Baseline

The highest performance on evaluation metrics with real-time performance belongs to the architecture proposed by [35]. Similar to all other depth estimation networks, their architecture also follows a U-Net-like structure composed of balanced encoder and decoder networks. Similarly, the layers of the two networks

sharing the same dimensions are connected with skip connections. MobileNet being one of the most widely preferred feature extractors for real-time applications is adopted as an encoder in their architecture for high-speed inference. As the decoder structure, they use a lightweight version of the architecture proposed by [55]. As Figure 31 shows the regular skip connections are manipulated by using a module called Chained Residual Pooling. This module intends to ease context information transfer on skip connections.

The fastest inference speed among real-time models belongs to MiniNet. As shown in Figure 32, they also follow a U-Net-like structure. The Recurrent module on the encoder section of their network is also built with the inverted residual block and Squeeze-and-Excitation (SE) strategy adopted by [26]. For having a faster inference, they propose an efficient upsampling block inspired by the depth-wise separable convolutions of [28]. Their metric performance is low due to a very lightweight decoder architecture.

4.3 Proposed Method

Recent advances in real-time semantic segmentation showed that using a bilateral network in parallel instead of upsampling with an additional decoder speeds up the inference time without compromising the predictive power.

To adapt this STDC-Seg into the MDE domain, we modify the convolutional architecture of the STDC backbone concerning the needs of the depth regression. Moreover, we make adjustments to the convolutional architecture of Attention

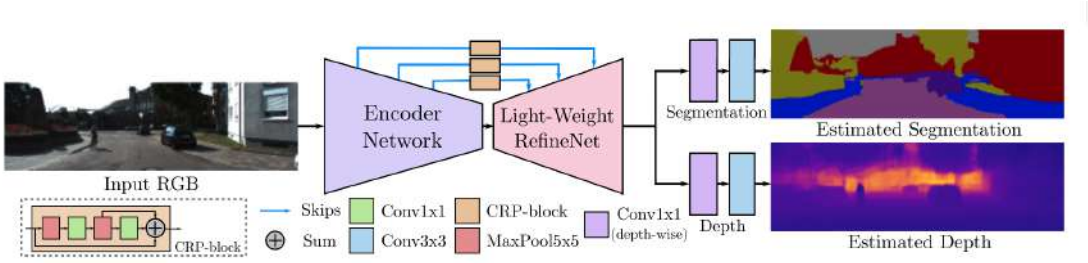

Figure 31: The U-Net like model structure of Nekrasov et al. from [13]

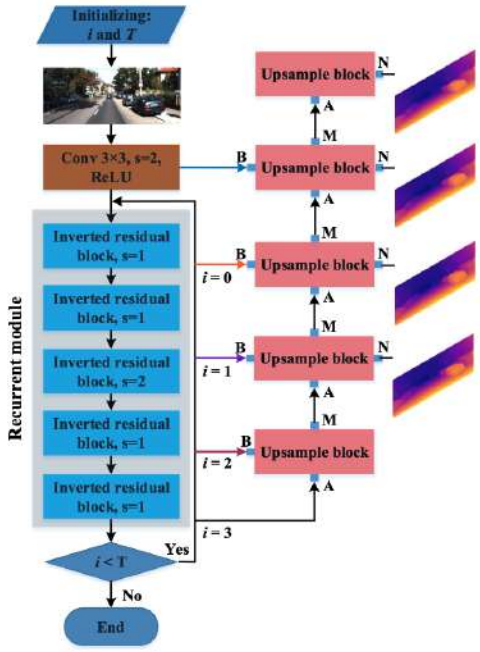

Figure 32: The U-Net like model structure of DepthNet used in MiniNet inference, taken from [14]

Refinement Modules and Feature Fusion Modules for balancing the accuracy-inference time trade-off.

As STDC-Seg, our architecture uses earlier stages of the backbone for processing spatial information and fuses those feature maps with the final outputs of the backbone. During our experiments, we examined that STDC output maps are low in terms of resolution for the depth estimation task, which results in low RMSE scores on predicted depth maps. Although the Spatial path helps to recover the resolution up to a point, it still needs an aggressive upsampling layer at the end of the network for achieving the dimensions of the input image. To address this issue, we decrease the stride on the initial convolution layers of the backbone inspired by [9], resulting in higher resolution feature maps at each stage. Higher-resolution feature maps, however, are known to slow down the convolutional operations significantly. To compensate for the inference time, we decrease the channel sizes on Attention Refinement Modules by taking into consideration the specific needs of the depth regression head. Furthermore, we change the Future Fusion Module accordingly for composing the information coming from the Context path and Stage

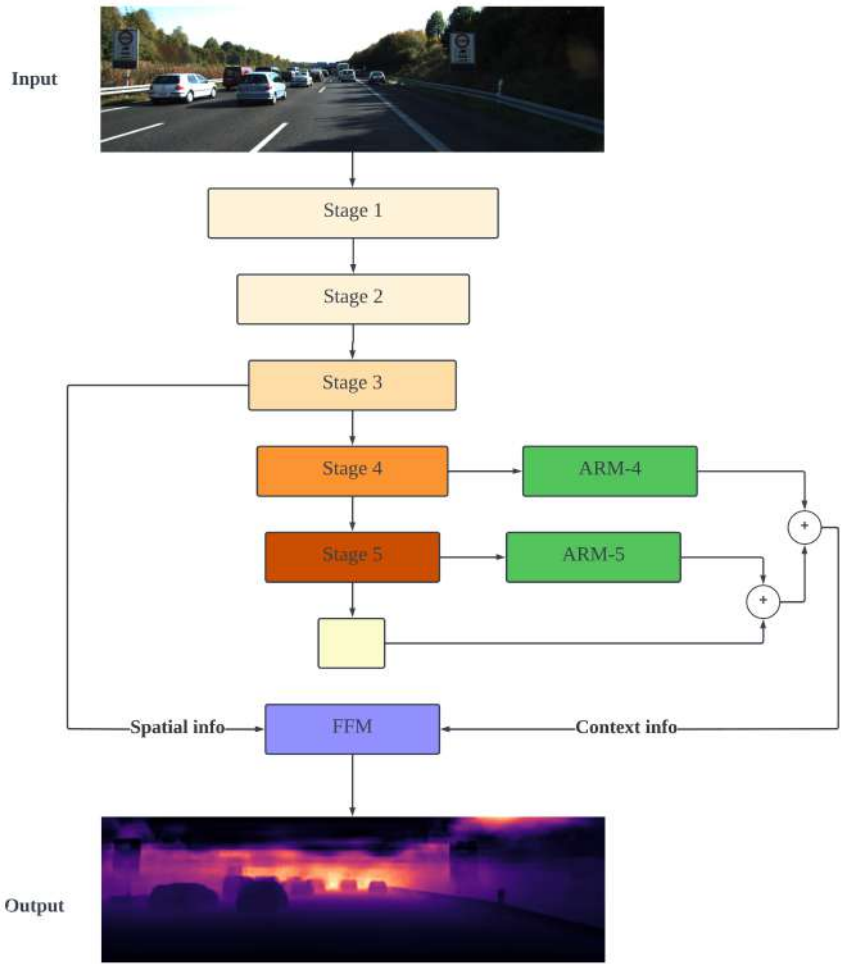

Figure 33: The overall structure of STDC-Depth

3. Table 2 shows the final channel sizes and resolutions of the proposed network at each block. Figure 33 shows how the mentioned blocks are connected and the overall flow.

For training the model, we follow the practice of the current MDE leader [42] and adopt their pixel-wise depth loss. Formally the loss we use is defined as,

$$\mathcal{L}_{pixel} = \alpha \sqrt{\frac{1}{T}\sum_{i} d_i^2 - \frac{\lambda}{T^2}(\sum_{i} d_i)^2} \tag{25}$$

, where $d_i = \log \hat{d}_i - \log d_i$ same as the case of the aforementioned scale invariant loss. To scale this equation, we adopt α and λ values of 10 and 0.85 respectively from [42].

Table 2: STDC-Depth(Final) Architecture Channel and Output Resolution Summary

Stages	Output size	Stride	Input Channels	Output Channels
Input	352, 1216	N/A	N/A	3
Stage1	176, 608	1	3	16
Stage2	176, 608	2	16	32
Stage3	88, 304	2	32	128
Stage4	44, 152	2	128	256
Stage5	22, 76	2	256	512
ARM 4	44, 152	N/A	256	64
ARM 5	22, 76	N/A	512	64
FFM	88, 304	N/A	[128,64]	128

Figure 34: The depth maps from LIDAR visualised as normalized heat-maps on right and matching RGB images on left. Samples are selected from KITTI training dataset

4.4 Dataset

KITTI is a well-known dataset both for semantic segmentation, depth prediction, and depth completion domains. For our case, we are interested in the depth prediction set which is formed by RGB and point cloud pairs of street scenes mainly composed of people and vehicles. The depth prediction dataset is arranged as various splits due to changing leader board evaluation preferences of Karlsruhe Institut fur Technologie (KIT). The test samples are not shared publicly with the

last version. Therefore, for evaluating our method with most of the prominent works in the literature, we also use the old Eigen Split which has 23k training and 697 testing samples respectively. Figure 34 shows the densified version of the point clouds acquired by Velodyne Lidar with corresponding RGB images. As can be seen, even with the additional processing the ground truth depth maps are sparse.

4.5 Implementation Details

All the experiments were made on an Ubuntu server that employs 2 NVIDIA Tesla V-100 32 GB, 126 GB of RAM, and an Intel Xeon Silver CPU with 2.20GHz frequency. As the development platform, we utilize PyTorch deep learning framework and we use WandB for visually tracking our experiments on the cloud.

During training, we used AdamW [57] as an optimizer with a learning rate of 1e-3. The weight decay is set to the default value of 1e-2 and the beta values are 0.9 and 0.999. We update our gradients with respect to the batches of 8 samples. Due to its popularity in the state-of-the-art Monocular Depth Estimation works, we use the KITTI dataset for training and evaluating our method. We train our network over 500 epochs on the Eigen Split and we use the range of 0 to 80 meters following the practice of the previous works. For calculating the loss, we mask our depth prediction according to available spots of the given ground truth depth map. For dataset augmentation, we adopt the regular pipeline of [42], where we use random rotations and Eigen crop. We choose not to use pre-trained weights and all the network weights are initiated using the Kaiming Normal method [58] at the beginning of the training.

The predicted depth maps are visualized by normalizing the pixel values and applying reverse magma and jet color palette from the MatPlotLib library.

CHAPTER V

RESULTS AND DISCUSSION

This study aims to explore the utilities of Bilateral Network structure on the well-studied Monocular Depth Estimation task. For measuring the benefits of our approach, we train our model on the KITTI dataset following the practice of Eigen Split. MDE is a pixel-wise regression problem, for evaluating our performance, we adopt the metrics of the previous studies.

The success of U-Net on biomedical segmentation tasks lead encoder-decoder architectures to become the default choice for domains needing outputs with image dimensions. MDE being no exception, most of the depth estimation networks followed the recent advances in classification models and adopted them into their architecture as the backbone. Although those architectures have achieved successful qualitative and quantitative results, using a decoder structure composed of heavy Up-convolution operations hinders inference time.

Influenced by the accuracy-speed balance of Bilateral Networks on the semantic segmentation domain, we modify the recently proposed STDC-Seg network for the MDE problem and evaluate our results compared to the standard and real-time models.

To achieve that, we first replaced the segmentation head of the STDC-Seg with the Ordinal Regression Layer. The study [39] has shown that quantizing the depth values into intervals and reframing the problem as estimating depth intervals boosts convergence. Their proposed model outputs multi-channel maps each representing a depth interval. Since this structure resembles a segmentation head, we have initially experimented with adopting their output strategy into STDC-Seg and trained the network on the KITTI dataset using the Ordinal Regression Loss from [39].

Table 3: Metric performances on KITTI Monocular Depth Estimation Benchmark (Eigen Split 80m)

Model name	**RMSE**	**Abs Rel.**	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	**inference**
Gan et al.	3.933	0.082	0.918	0.984	0.996	N.A.
Fu et al.	2.727	0.072	0.932	0.984	0.994	6.6
Ours	3.378	0.079	0.922	0.986	0.997	**0.6**
Yin et al.	3.258	0.072	0.938	0.990	0.998	7.9
BTS	2.750	0.059	0.956	0.993	0.998	7.1
Adabins	**2.360**	**0.058**	**0.964**	**0.995**	**0.999**	5.4

Our repetitive experiments show that a bilateral setting with STDC and an Ordinal Regression layer converges to non-competitive results despite having real-time performance. As an alternative, we replaced the segmentation head with a depth regression layer. This setup along with the scale-invariant loss has shown significant improvement during training. Moreover, during our experiments, we examined that training the network from scratch results in remarkably better convergence than using pre-trained backbone weights from the classification task.

We also experimented with different strategies for using the KITTI dataset. As we have shown earlier, the Velodyne LIDAR outputs are very sparse even though it has great metric precision. Initially, we used bi-linear interpolation techniques for having more supervision at each batch. However, our experience has shown that using a mask over predictions resulted in conspicuously higher accuracy on every metric.

At this point, STDC with a depth regression layer has a lower RMSE score than the best-performing real-time model of Nekrasov et al.. This is due to the output dimensions having a low resolution for the pixel-wise depth prediction task. Compared to the segmentation masks, a depth map has richer details. To cover that, we modified the STDC backbone to output higher resolution depth maps at each stage. As the work of EfficientNet has shown, augmenting the resolution of the feature maps remarkably slows down the inference, although it improves the overall accuracy. In order to maintain a decent accuracy-speed balance, we further decrease the number of channels on FFM and ARM blocks. The improvements have increased the RMSE performance of our model, ranking the best RMSE, Abs

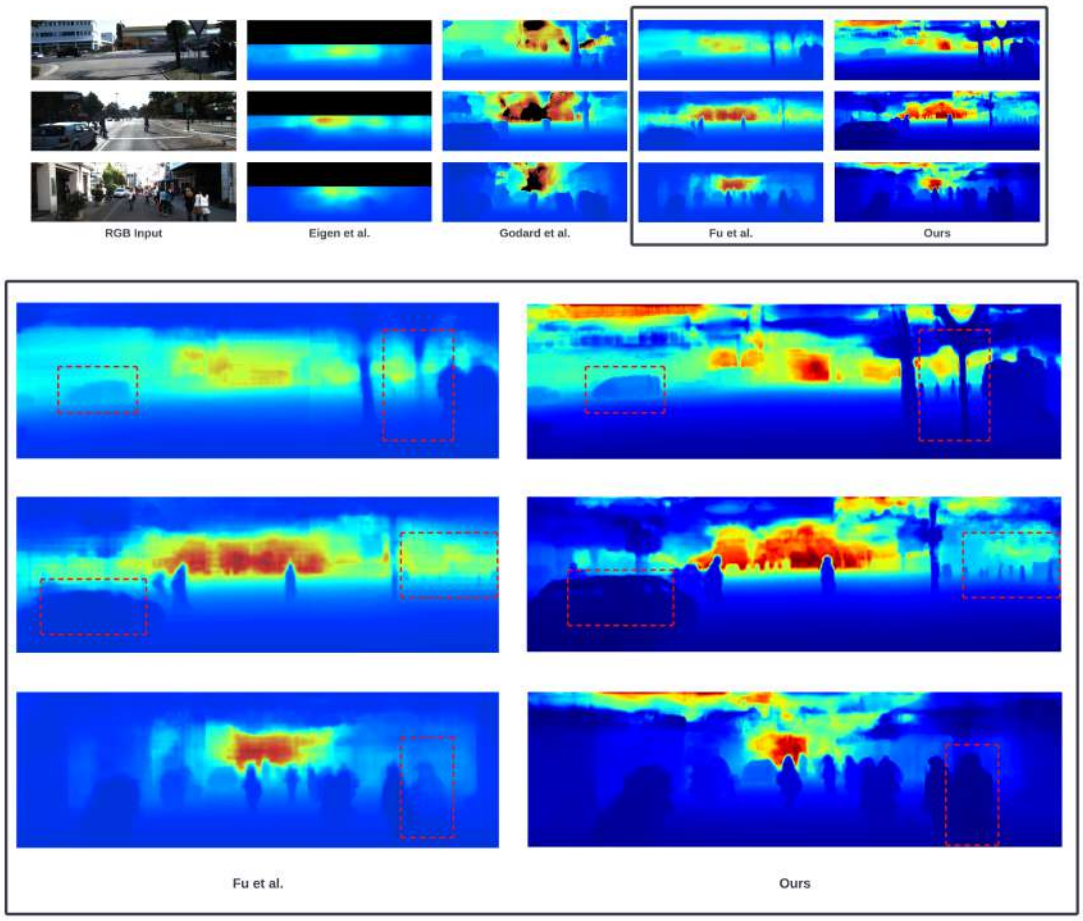

Figure 35: Our qualitative results compared with the closest non real-time competitor

Rel. and δ_1 score among real-time models. Table 4 shows the quantitative results of real-time models that have studied the KITTI dataset.

As Table 3 shows, our method has also achieved comparable results with the competitive KITTI MDE Benchmark models that are not subject to any time constraints. The inference column of the table shows the inference time of the method in seconds caluculated on Intel i7 CPU with 2.6 GHz of clock frequency. Our method achieves up to 11 times faster inference time while maintaining similar metric performance.

Qualitatively, our method detects the close objects better than the study [39] which has the closest metric performance to ours on Table 3. As can be seen in Figure 35, our model catches smaller details such as barriers and traffic cones.

Table 4: Metric performances of Real-time Monocular Depth Estimation models on KITTI dataset (Eigen Split 80m)

Model name	**RMSE**	**Abs Rel.**	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	**inference**
MiniNet	5.264	0.141	0.825	0.941	0.976	**0.027**
Atapour et al.	4.726	0.110	0.903	0.988	0.992	3.650
CReaM	4.367	0.156	0.818	0.940	0.977	N.A.
Chiu et al.	3.971	0.106	0.897	**0.998**	**0.999**	N.A.
STDC-Depth (low res)	3.520	0.082	0.918	0.984	0.996	0.210
Nekrasov et al.	3.452	N.A.	N.A.	N.A.	N.A.	2.760
STDC-Depth (Final)	**3.378**	**0.079**	**0.922**	0.986	0.997	0.603

Moreover, our model is more consistent with stick-like vertical objects. Examining the leaders of the Table 3, our method has a close qualitative performance compared to the studies [42] and [41], whereas our method can infer depth-maps in real-time. As Figure 36 and Figure 37 shows our model is also capable of constructing table like structures consistently comparable to the leader case [42].

In Figure 38, we examine that although [41] is capable of constructing more defined depth maps, most of the objects have bubble-like structures due to some object connecting behavior. The two close objects merged into each other on the prediction, whereas our model was capable of distinguishing the two structures although it presents a lower resolution. Additionally, our model carefully reconstructed the roof of the car straight like the leader [42].

Ablation Study: In our study, we experiment with different architectural configurations in order to examine the effect of the modules. Our repetitive experiments with STDC-Depth architecture show that the model can not converge without the ARM-5 module. Table 5 refers to this architecture as ARM-4 Only. On the other hand, STDC-Depth without the ARM-4 module has shown comparable performance to the configurations having both modules. Therefore we conclude that although the ARM-4 module improves our prediction accuracy, the ARM-5 is the essential module for our model.

Although removing the block Stage 5 and ARM-5 module speeds up the inference time, STDC-Depth requires a deeper architecture for recovering enough

Table 5: STDC-Depth with different architecture configurations on KITTI dataset (Eigen Split 80m)

Model name	**RMSE**	**Abs Rel.**	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
ARM-4 Only	20.289	0.999	0	0	0
ARM-5 Only	3.486	0.085	0.914	0.984	0.996
Final	**3.27**	**0.082**	**0.924**	**0.986**	**0.997**

receptive field to converge. For validating this convergence issue, we have experimented with different batch sizes and also lower learning rates. We further tested Adam [59], AdamW, and SGD optimizers and confirmed that ARM-4 Only architecture has stability issues and tends not to converge any competitive result.

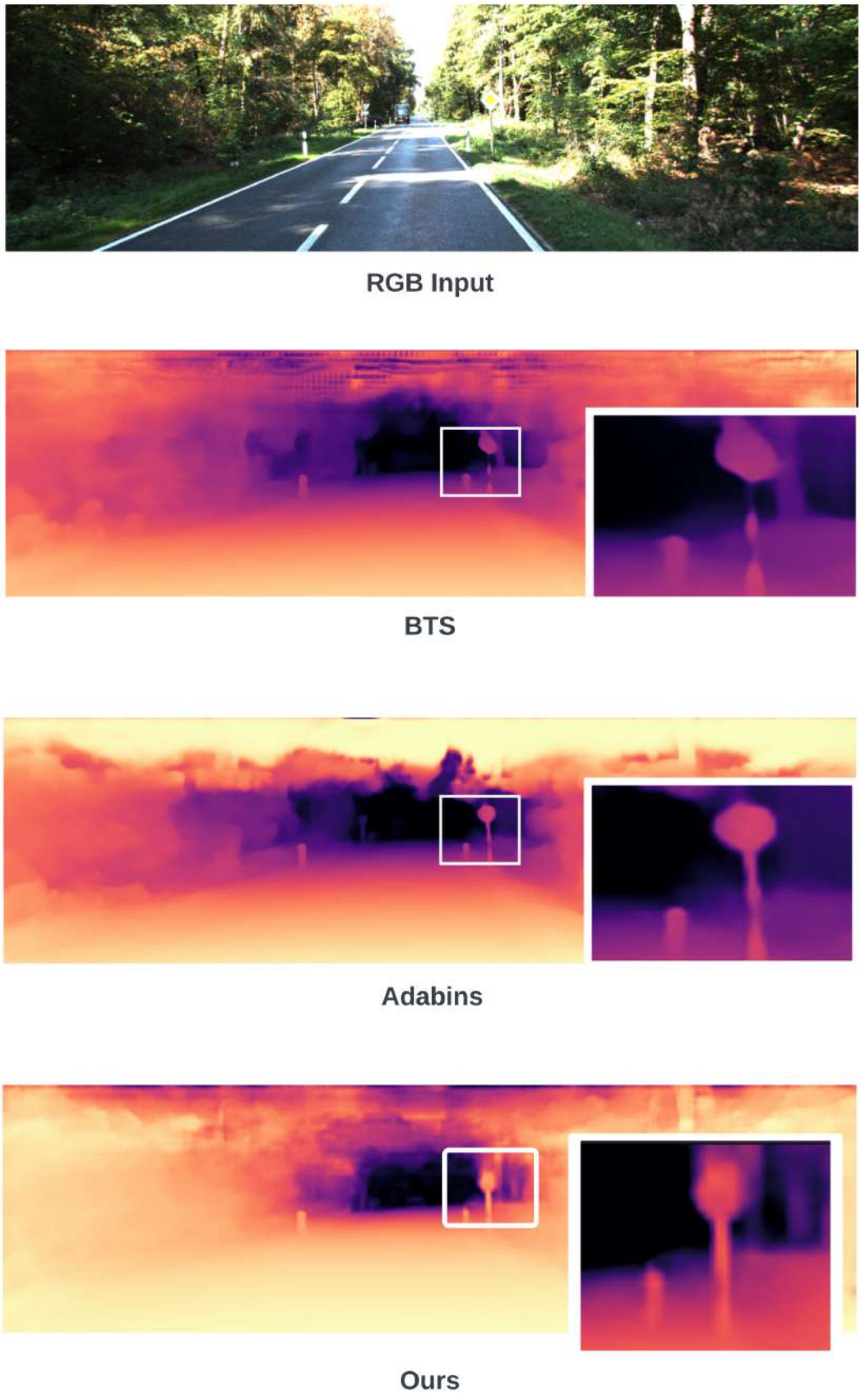

Figure 36: Our qualitative results compared with the leader SOTA works without any time constraint, the forest scene with the sign.

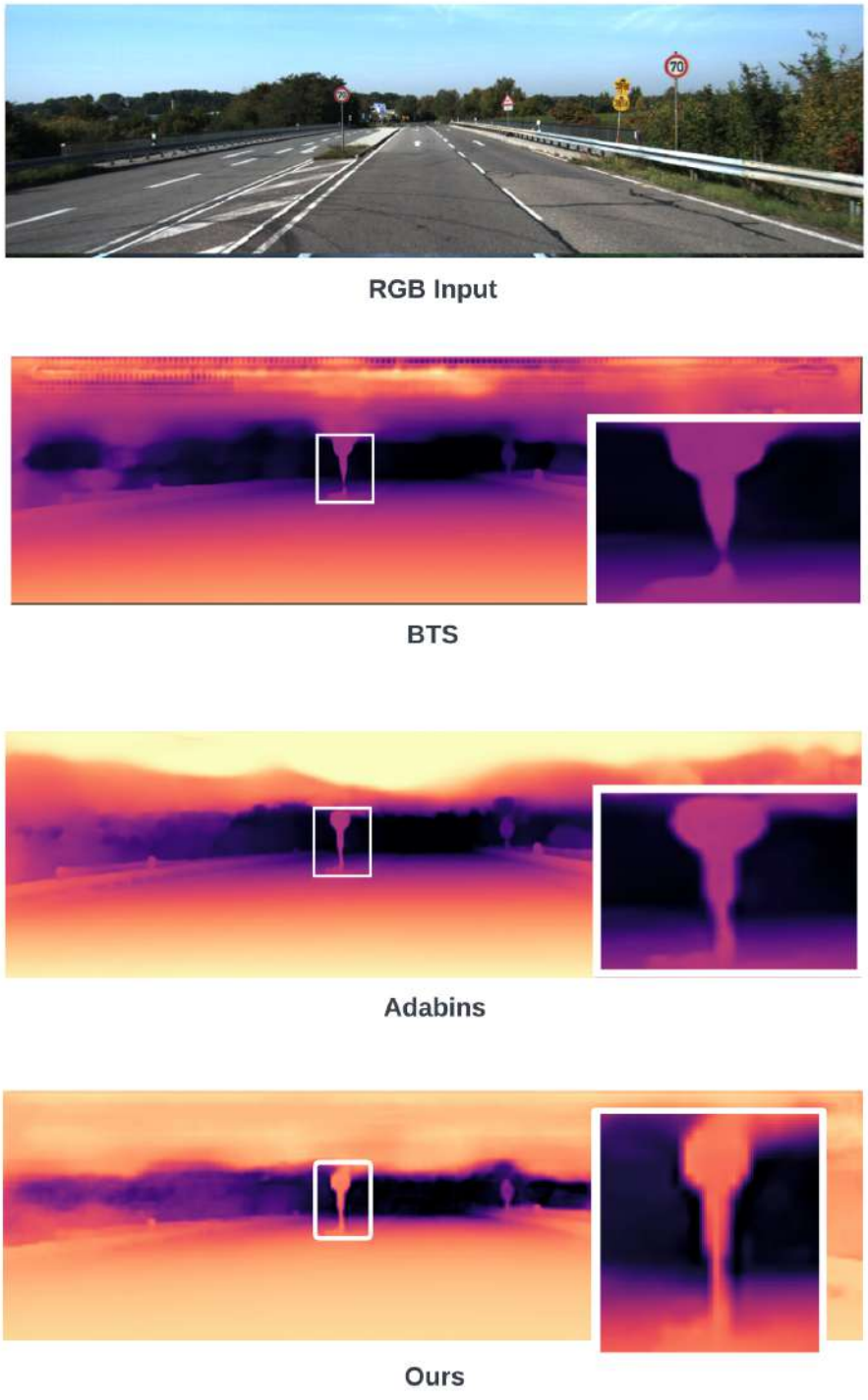

Figure 37: Our qualitative results compared with the leader SOTA works without any time constraint, the scene with 4 lane highway.

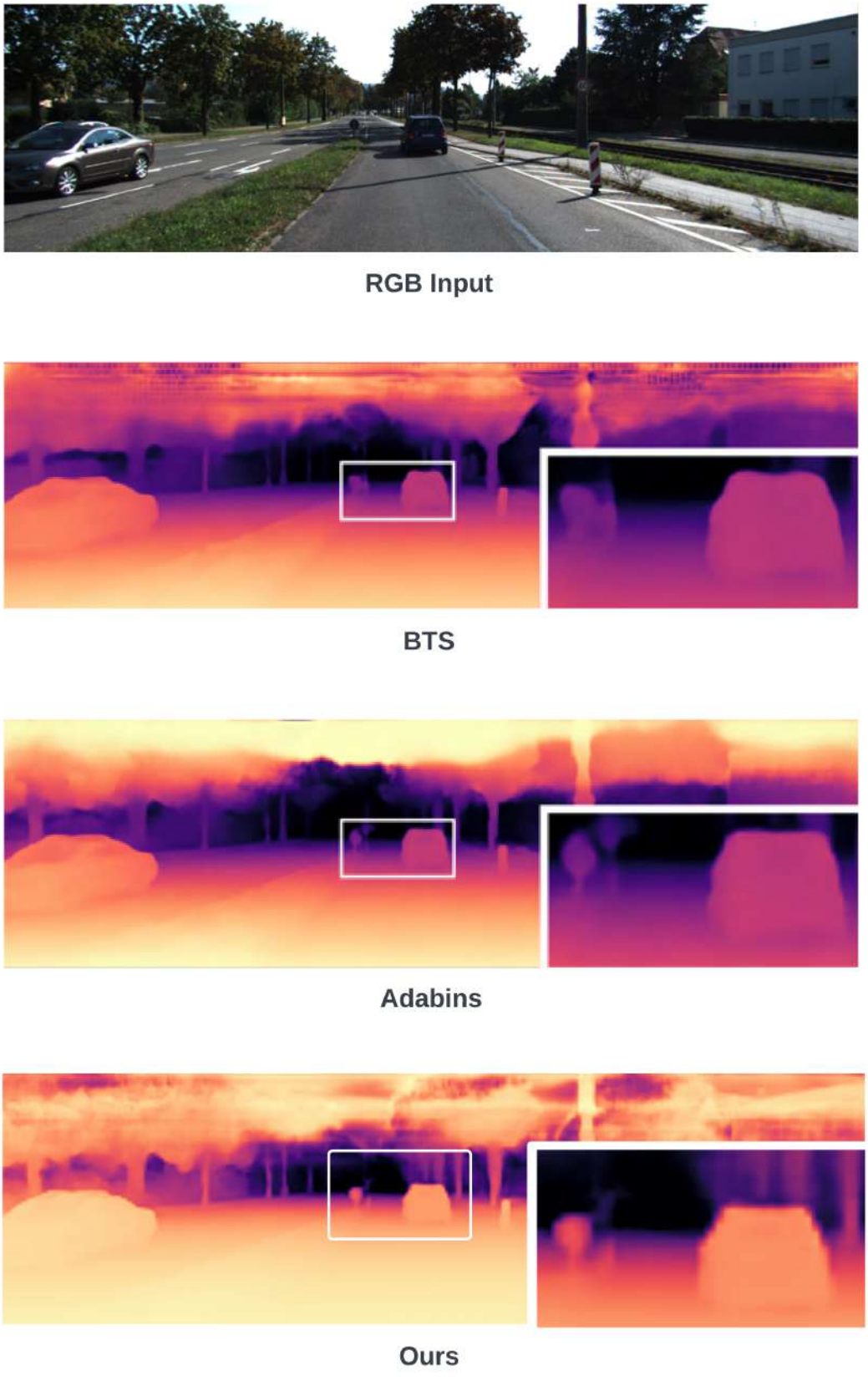

Figure 38: Our qualitative results compared with the leader SOTA works without any time constraint, the scene with the car.

CHAPTER VI

CONCLUSION

Depth estimation is a challenging but attractive problem as a consequence of its industrial value. The logistic advantages of using single images for depth prediction over using image pairs or expensive active range sensors have led to an interest in the field of MDE. The advances in CNNs and the success of deep learning in computer vision have also affected the domain of MDE. The success of U-Net architecture has influenced many computer vision fields that require image shape outputs, including MDE. However, the problem of depth estimation is subject to time constraints for the inference time due to its potential product applications.

In our study, we conclude that Bilateral structures can be adapted to the MDE domain for managing the speed-accuracy trade-off. A mixture of encoder and attention mechanisms used to bridge high-resolution and low-resolution feature maps is an efficient structure that can be used as an alternative to a decoder network. Our experiments have shown that using the feature fusion strategy on the encoder generates accurate depth maps while achieving real-time performance. Our proposed method still leverages a simple interpolation as an upsampling method at the end of the network. This may hinder the quantitative and qualitative performances of our method. The future study might investigate potential alternatives to a better approach for upsampling without revealing computational bottlenecks. Moreover, similar to [12], an early layer supervision strategy can be employed for guiding Spatial feature maps to convey higher quality depth information.

APPENDIX A

MORE QUALITATIVE RESULTS

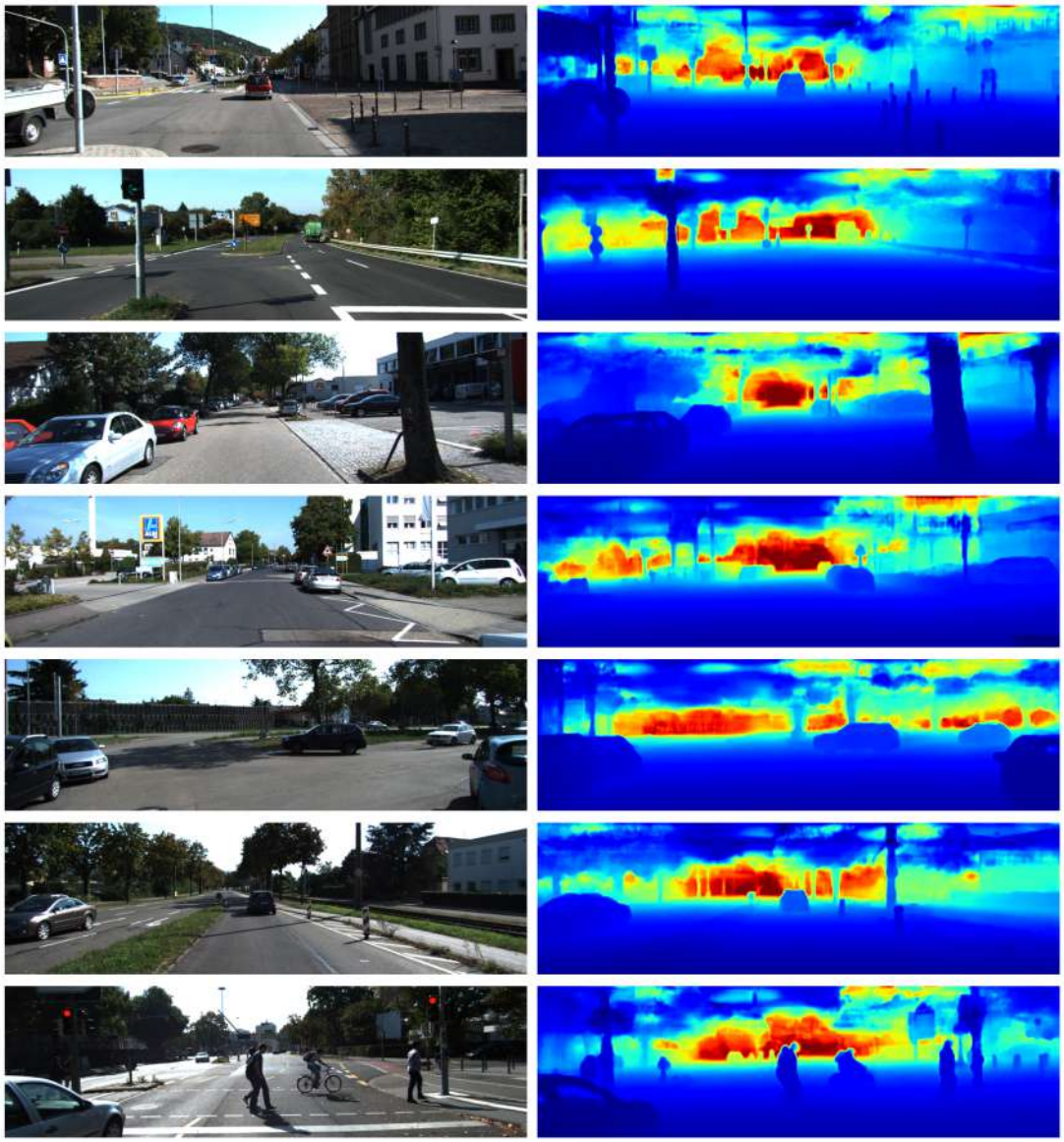

Figure 39: More of our qualitative results on KITTI Eigen Split Test set.

REFERENCES

[1] P. Sinha and E. Adelson, "Recovering reflectance and illumination in a world of painted polyhedra," in *(4th) International Conference on Computer Vision*, pp. 156–163, IEEE, 1993.

[2] J. J. Fei-Fei Li and S. Yeung, "Lecture 7: Training neural networks, part 2." http://cs231n.stanford.edu/slides/2018/cs231n$_2$018$_l ecture$07.*pdf*, 2018. *Accessed* : 2022–03 − 06.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1492–1500, 2017.

[5] K. Bai, "A comprehensive introduction to different types of convolutions in deep learning." https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215, 2019. Accessed: 2022–04-03.

[6] I. N. Junejo and N. Ahmed, "Depthwise separable convolutional neural networks for pedestrian attribute recognition," *SN Computer Science*, vol. 2, no. 2, pp. 1–11, 2021.

[7] 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*, pp. 4510–4520, 2018.

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks cvpr presentation." shorturl.at/pwEG9, 2017. Accessed: 2022–04-03.

[9] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, pp. 6105–6114, PMLR, 2019.

[10] 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*, pp. 234–241, Springer, 2015.

[11] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 325–341, 2018.

[12] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking bisenet for real-time semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9716–9725, 2021.

[13] V. Nekrasov, T. Dharmasiri, A. Spek, T. Drummond, C. Shen, and I. Reid, "Real-time joint semantic segmentation and depth estimation using asymmetric annotations," in *International Conference on Robotics and Automation (ICRA)*, pp. 7101–7107, IEEE, 2019.

[14] J. Liu, Q. Li, R. Cao, W. Tang, and G. Qiu, "Mininet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 166, pp. 255–267, 2020.

[15] "Mobile augmented reality (ar) market revenue worldwide from 2019 to 2025." https://www.statista.com/statistics/282453/mobile-augmented-reality-market-size/. Accessed: 2022-04-30.

[16] "Global robotics market - growth, trends, covid-19 impact, and forecasts (2022 - 2027)." https://www.mordorintelligence.com/industry-reports/robotics-market. Accessed: 2022-04-30.

[17] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[18] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[19] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International Workshop on Artificial Neural Networks*, pp. 195–201, Springer, 1995.

[20] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, p. 3, Citeseer, 2013.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, 1988.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.

[24] I. Goodfellow, Y. Bengio, and A. Courville, "Softmax units for multinoulli output distributions. deep learning," 2016.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

[26] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

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

[29] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *International Conference on Computer Vision*, pp. 2018–2025, IEEE, 2011.

[30] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Fourth International Conference on 3D Vision (3DV)*, pp. 239–248, IEEE, 2016.

[31] S. Ullman, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979.

[32] S. Palmisano, B. Gillam, D. G. Govan, R. S. Allison, and J. M. Harris, "Stereoscopic perception of real depths at large distances," *Journal of Vision*, vol. 10, no. 6, pp. 19–19, 2010.

[33] C. H. Hung, L. Xu, and J. Jia, "Consistent binocular depth and scene flow with chained temporal profiles," *International Journal of Computer Vision*, vol. 102, no. 1, pp. 271–292, 2013.

[34] Y.-S. Chen and B.-T. Chen, "Measuring of a three-dimensional surface by use of a spatial distance computation," *Applied Optics*, vol. 42, no. 11, pp. 1958–1972, 2003.

[35] J. J. Koenderink and A. J. Van Doorn, "Affine structure from motion," *JOSA A*, vol. 8, no. 2, pp. 377–385, 1991.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[37] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 270–279, 2017.

[38] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6647–6655, 2017.

[39] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2002–2011, 2018.

[40] W. Yin, Y. Liu, C. Shen, and Y. Yan, "Enforcing geometric constraints of virtual normal for depth prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5684–5693, 2019.

[41] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[42] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4009–4018, 2021.

[43] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision*, pp. 746–760, Springer, 2012.

[44] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2024–2039, 2015.

[45] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[46] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics*, pp. 492–518, Springer, 1992.

[47] Y. Gan, X. Xu, W. Sun, and L. Lin, "Monocular depth estimation with affinity, vertical pooling, and label enhancement," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 224–239, 2018.

[48] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[49] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 605–613, 2017.

[50] A. Spek, T. Dharmasiri, and T. Drummond, "Cream: Condensed real-time models for depth prediction using convolutional neural networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 540–547, IEEE, 2018.

[51] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2800–2810, 2018.

[52] I. Goodfellow *et al.*, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[53] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2223–2232, 2017.

[54] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2414–2423, 2016.

[55] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1925–1934, 2017.

[56] M.-J. Chiu, W.-C. Chiu, H.-T. Chen, and J.-H. Chuang, "Real-time monocular depth estimation with extremely light-weight neural network," in *25th International Conference on Pattern Recognition (ICPR)*, pp. 7050–7057, IEEE, 2021.

[57] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[58] J. He, M. Lan, C.-L. Tan, S.-Y. Sung, and H.-B. Low, "Initialization of cluster refinement algorithms: A review and comparative study," in *IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 1, pp. 297–302, IEEE, 2004.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.