Geometric and Semantic Understanding of Objects from a Single Image Using Deep Learning

vorgelegt von
M.Sc.
Ahmed J. M. Afifi
ORCID: 0000-0001-6782-6753

an der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
-Dr.-Ing.-

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Manfred Opper	-	TU Berlin
1. Gutachter: Prof. Dr. Olaf Hellwich	-	TU Berlin
2. Gutachter: Prof. Dr. Hamid Laga	-	Murdoch University
3. Gutachter: Prof. Dr. Klaus Obermayer	-	TU Berlin

Tag der wissenschaftlichen Aussprache: 28. Juli 2021

Berlin 2021

بِسْمِ اللهِ الرَّحْمَنِ الرَّحِيمِ

﴿وَعَلَّمَكَ مَا لَمْ تَكُن تَعْلَمُ وَكَانَ فَضْلُ اللهِ عَلَيْكَ عَظِيمًا﴾

سورة النساء – آية رقم ١١٣

In the Name of Allah, the Most Beneficent, the Most Merciful
"*Allah (The God) has taught you what you did not know before, and great is Allah's grace upon you.*"
The Holy Quran, Chapter 4, Verse 113.

Im Namen Allahs, des Allerbarmers, des Barmherzigen
"Allah hat dich gelehrt, was du (vorher) nicht wußtest. Und Allahs Huld gegen dich ist groß"
Der Heiliger Koran, Kapitel 4, Vers 113.

Dedicated to

JAMAL

The Father(1958-2004) & *The Son*(5-13 March, 2017)

I love my land “Palestine” and my family.

Acknowledgements

First of all, my ultimate Praise and thanks be to ALLAH for the strength and His blessing in completing this dissertation after all the challenges and difficulties.

My heartiest gratitude is to my supervisor Prof. Dr. Olaf Hellwich for his continuous encouragement, valuable assistance, and endless support. Without his guidance, it would not have been possible to keep this work on track and deliver it to the fullest. I appreciate the time he dedicated to me to supervise my dissertation and the long and fruitful discussions we spent in answering some research questions. Also, I would like to thank my dissertation committee members; Prof. Dr. Manfred Opper, Prof. Dr. Hamid Laga, and Prof. Dr. Klaus Obermayer for their feedback and comments. I really appreciate the helpful comments and the insightful suggestions on my work.

I would also like to thank Nader Al-deeb, my friend and my office mate. The time and the fruitful discussions we had during our research journey were priceless. Also, I would like to thank other colleagues in the department for being very kind and friendly with me. A special thank to Marion for her incredible support and patience with me to answer many administrative questions and requests.

Nearly last, but by no means least, I would like to thank Dr. Toufique Ahmed Soomro for the amazing collaboration and the scientific work we have achieved together in the field of medical imaging analysis. My special thanks and gratitude goes to my friend Eng. Mahmoud Al-Hoby for proofreading this dissertation. Also, I would like to thank Jannes Magnusson for his help in generating some results and finalizing the abstract in the German language. Thank you Hardik Jain for the technical support during the defense.

On top of all, my gratefulness is never-ending for: my mother, Jamilah, for her endless prayers to me and her limitless encouragement and patience. My thanks are to my sisters, brothers, aunts, and uncles. Finally, my heartfelt gratitude to my beloved wife Heba, and my children Mohammed, Sara, and Siwar. Heba, without your support, this work would not be what it is.

I am gratitude to Erasmus Mundus (AVEMPACE II Project) and DAAD (STIBET Program) for the financial support they provided to me to conduct my Ph.D. studies in Berlin, Germany.

Thank you so much!

Publications

During my Ph.D. research, I have published 11 research papers. Parts of this dissertation have been published in the following 4 reearch papers:

- **Afifi, Ahmed J.**, Jannes Magnusson, Toufique A. Soomro, and Olaf Hellwich. "Pixel2Point: 3D object reconstruction from a single image using CNN and initial sphere." IEEE Access (9), pp. 110-121, 2020.
- **Afifi, Ahmed J.**, Olaf Hellwich, and Toufique Ahmed Soomro. "Mini V-Net: Depth Estimation from Single Indoor-Outdoor Images using Strided-CNN." In VISIGRAPP (5: VISAPP), pp. 205-214, 2020.
- **Afifi, Ahmed J.**, Olaf Hellwich, and Toufique Ahmed Soomro. "Simultaneous Object Classification and Viewpoint Estimation using Deep Multi-task Convolutional Neural Network." In VISIGRAPP (5: VISAPP), pp. 177-184, 2018.
- **Afifi, Ahmed J.**, and Olaf Hellwich. "Object depth estimation from a single image using fully convolutional neural network." In 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pp. 1-7. IEEE, 2016.

Also, I have shared my knowledge of computer vision and deep learning with my colleagues in solving different attractive research issues in the medical imaging field. As a result, these publications are produced but not discussed in the dissertation :

- Magnusson, Jannes, **Ahmed J. Afifi**, Shengjia Zhang, Andreas Ley, and Olaf Hellwich. "Synthesizing Fundus Photographies for Training Segmentation Networks." In DeLTA, pp. 67-78, 2021.

- Soomro, Toufique A., Lihong Zheng, **Ahmed J. Afifi**, Ahmed Ali, Ming Yin, and Junbin Gao. "Artificial intelligence (AI) for medical imaging to combat coronavirus disease (COVID-19): a detailed review with direction for future research." Artificial Intelligence Review (2021): 1-31.

- Soomro, Toufique Ahmed, **Ahmed J. Afifi**, Ahmed Ali Shah, Shafiullah Soomro, Gulsher Ali Baloch, Lihong Zheng, Ming Yin, and Junbin Gao. "Impact of Image Enhancement Technique on CNN Model for Retinal Blood Vessels Segmentation." IEEE Access (7), pp. 158183-158197. 2019.

- Soomro, Toufique Ahmed, **Ahmed J. Afifi**, Lihong Zheng, Shafiullah Soomro, Junbin Gao, Olaf Hellwich, and Manoranjan Paul. "Deep learning models for retinal blood vessels segmentation: A review." IEEE Access (7), pp 71696-71717, 2019.

- Soomro, Toufique Ahmed, **Ahmed J. Afifi**, Junbin Gao, Olaf Hellwich, Lihong Zheng, and Manoranjan Paul. "Strided fully convolutional neural network for boosting the sensitivity of retinal blood vessels segmentation." Expert Systems with Applications (134), pp. 36-52, 2019.

- Soomro, Toufique A., **Ahmed J. Afifi**, Junbin Gao, Olaf Hellwich, Manoranjan Paul, and Lihong Zheng. "Strided U-Net model: Retinal vessels segmentation using dice loss." In 2018 Digital Image Computing: Techniques and Applications (DICTA), pp. 1-8. IEEE, 2018.

- Soomro, Toufique Ahmed, Junbin Gao, Zheng Lihong, **Ahmed J. Afifi**, Shafiullah Soomro, and Manoranjan Paul. "Retinal blood vessels extraction of challenging images." In Australasian Conference on Data Mining, pp. 347-359. Springer, Singapore, 2018.

- Soomro, Toufique A., **Ahmed J. Afifi**, Junbin Gao, Olaf Hellwich, Mohammad AU Khan, Manoranjan Paul, and Lihong Zheng. "Boosting sensitivity of a retinal vessel segmentation algorithm with convolutional neural network." In

2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pp. 1-8. IEEE, 2017.

Abstract

Since the dawn of computer vision, semantic and geometric scene understanding has been an essential problem. It impacted early works and led to numerous real-world applications. A scene is a place where an agent can act with or navigate. Scene understanding is the process of analyzing the semantic information inside the static 2D image and the relationships between the scene contents.

This dissertation presents the use of deep learning models and how they can be applied to recognize the surrounding environment given a single still image. The scene captured in an image contains different objects from various classes and they are looking in different directions. These objects appear in the image either small because they are far from the camera or large because they are very close to the camera. When the scene is complex, we find that some objects are occluded and it is hard sometimes to predict contextual information about the complete 3D object. Also, using a single view always leads to ill-posed problems as there is no enough information to reveal the ambiguity. So, to build a useful and complete understanding of a given scene, we have to answer the following questions: (1) what are the object classes that appear in the image? (2) from which direction do they look at the camera? (3) are the objects far or close to the camera? (4) if some object parts are hidden, how can the machine generate the 3D information of the object?

More specifically, this dissertation approaches scene understanding by answering the above-mentioned questions. First, we present a multi-task CNN model that performs object classification and viewpoint estimation simultaneously. These two problems seem to have opposite representative features. For object classification, the features should be powerful so that they are orientation-invariant features. For viewpoint estimation,

the features should preserve the orientation characteristic and the ability to describe the same object with different viewpoints. To this end, the proposed CNN has a shared network for both tasks and task-specific sub-networks. With this design, the mentioned problems can be solved using the same model without proposing parallel models and separating the tasks.

Second, we address the problem of depth estimation from a single image. We formulate this problem as a regression task and propose two different CNN models to solve this task. The first model uses a series of stacked convolutional layers that generate the depth map from a single image. This model is a fully convolutional network that regresses the depth information for each pixel. However, the generated depth images are smaller than the input images and they are blurry. To overcome this drawback, we propose a second model that has encoder-decoder architecture. This model has an encoder part to extract useful features and a decoder part to reconstruct the image and generate the final depth image. The useful information extracted in the encoder part is transferred to the decoder part using the skip-connections. This information helps in generating more accurate depth images with sharper objects' parts. For this task, we select a non-convex loss function to train the models and optimize the network weights. The no-convex loss function is robust against to the outliers and converges faster.

In the last part of the dissertation, we propose a CNN model that reconstructs 3D point clouds of objects form a single image. Single-view reconstruction is a hard task as the 3D structure can be inferred accurately for specific object classes given restricted assumptions. The proposed model is a simple, yet powerful, that utilizes an initial point cloud of a sphere shape to generate the final point cloud. For this problem, we use a large-scale 3D dataset to train the model. Many images are rendered from each 3D model and used for training. To generate the ground-truth point clouds, we sample the points from the object's surface. The proposed model takes a single image of an object and generates a point cloud that presents the desired object accurately. An initial sphere is used to spread the points evenly on the reconstructed object surface and prevents the points to be grouped in some parts of the objects.

This dissertation demonstrates the leverage of using deep learning technology to give a comprehensive understanding of a scene using a single image only. We conclude this dissertation with a general discussion and propose several potential future research directions for better deep learning understanding and interpretation in solving computer vision tasks.

Zusammenfassung

Seit dem Beginn von Computer Vision war das semantische und geometrische Verständnis von Szenen ein essentielles Problem. Es beeinflusste erste Arbeiten und führte zu zahlreichen Anwendungen. Eine Szene beschreibt einen Ort in welchem ein Agent agiert oder navigiert. Daher ist das Verständnis einer Szene der Prozess semantische Informationen in einem statischen 2D Bild zu analysieren und die Beziehungen zwischen den einzelnen Inhalten in diesem Bild zu verstehen.

Diese Dissertation präsentiert wie verschiedene Deep Learning Modelle genutzt und für ein Verständnis von Szenerien aus einem einzigen Bild eingesetzt werden können. Ein aufgenommenes Bild enthält verschiedene Objekte aus verschiedenen Klassen, die unterschiedlich ausgerichtet sind. Diese Objekte können entweder durch eine große Distanz zur Kamera sehr klein erscheinen oder groß, wenn sie sich direkt vor der Kamera befinden. Enthält eine Szene viele Objekte, verdecken einige Objekte andere und schränken somit die verfügbaren Informationen über das komplette 3D Objekt ein. Somit führt ein einzelnes Bild immer zu Problemen mit unzureichenden Informationen um Unklarheiten aufzulösen. Für das kreieren eines nützliches und komplettes Verständnis einer gegebenen Szene müssen folgende Fragen beantwortet werden: (1) Welche Objektklassen kommen in dem Bild vor? (2) Aus welcher Richtung betrachtet die Kamera die Szene? (3) Sind die Objekte nah an der Kamera oder weit weg? (4) Wenn Teile von Objekten verdeckt sind, wie kann eine Maschine diese 3D Informationen ergänzen?

Genauer, diese Dissertation nähert sich dem Problem des Szenenverständnisses indem sie die oben genannten Fragen beantwortet. Als erstes wird ein multi-task CNN Modell vorgestellt, welches sowohl Objektklassifizierung als auch den Betrachtungs-

standpunkt schätzt. Diese beiden Probleme scheinen gegensätzliche repräsentative Features zu benutzen. Für Objektklassifizierungen sollten die Features invariant zur Orientierung des Objektes sein. Für die Schätzung des Betrachtungsstandpunktes sollte jedoch die Orientierung erhalten bleiben. Zusätzlich sollte die Fähigkeit bestehen bleiben Objekte von verschiedenen Blickwinkeln aus zu beschreiben. Dennoch hat das vorgeschlagene Modell ein neuronales Netz, das für beide Probleme genutzt wird, und problemspezifische Subnetze. Mit diesem Design können beide Aufgaben mit dem gleichen Modell gelöst werden statt jedes Problem parallel und separat zu lösen.

Als zweites wird die Problematik der Tiefenbestimmung aus einem Bild behandelt. Hierfür werden zwei verschiedene CNNs vorgeschlagen um das Problem als Regression zu lösen. Das erste benutzt eine Reihe von aufeinanderfolgenden Faltungen, die eine Tiefenkarte aus einem einzelnen Bild generiert. Dieses ist ein reines Faltungsnetz, welches die Tiefe für jeden Pixel per Regression bestimmt. Allerdings sind diese generierten Bilder kleiner als das Ursprüngliche und verwaschen. Daher wird ein weiteres Modell vorgeschlagen, welches einer encoder-decoder Architektur folgt. Der Encoder extrahiert nützliche Features aus denen der Decoder dann das ürsprüngliche Bild rekonstruiert und das fertige Tiefenbild generiert. Die nützlichen Informationen werden im Encoder extrahiert und durch sogenannte Skip-Connections an den Decoder weiter gegeben. Diese Informationen verbessern die Genauigkeit der Tiefenkarte und schärferen Objektkanten. Für das Trainieren und Optimieren der Netzwerkgewichte wurde eine nicht konvexe Verlustfunktion gewählt. Die nicht konvexe Verlustfunktion führt zu weniger Einfluss von Ausreißern und konvergiert schneller.

Im letzten Teil der Dissertation wird ein CNN Modell eingeführt, das 3D Punktwolken aus einem einzigen Bild rekonstruiert. Rekonstruktionen durch einen einzigen Betrachtungswinkel ist ein schweres Problem, da die 3D Struktur für spezifische Objekte nur durch Annahmen genau geschlussfolgert werden kann. Das vorgeschlagene Modell is einfach und trotzdem wirkungsvoll. Es benutzt eine Punktwolke einer Kugel um daraus die Punktwolke des Objekts zu generieren. Der Trainingsprozess basiert auf einem großzügig angelegten 3D Datenset. Aus 3D Modellen wurden viele Bilder erzeugt, die als Trainingsgrundlage dienen. Als optimale Lösung wurden Punkte auf

der Oberfläche der 3D Geometrien extrahiert. Das vorgeschlagene Modell nutzt daher ein einzelnes Bild um daraus eine Punktwolke zu generieren, welche das gewünschte Objekt akkurat visualisiert und beschreibt. Die ursprüngliche Kugel wird genutzt um die Punkte gleichmäßig auf dem rekonstruierten Objekt zu verteilen. Sie verhindert, dass sich Punkte an einigen Stellen des Objekts gruppieren.

Es wird demonstriert welchen Wirkungsgrad Deep Learning Technologien haben können um ein umfassendes Verständnis von Szenen aus nur einem einzelnen Bild zu erlangen. Die Arbeit wird mit einer Diskussion beendet und schlägt verschiedene potentielle Forschungsrichtungen vor, die zu einem besseren Verständnis und Interpretation von Deep Learning in Computer Vision führen können.

Table of Contents

List of Figures

List of Tables

1

Introduction

A picture is worth a thousand words.

Arthur Brisbane

The main task of this dissertation is how we can make the computer understands the human-made objects in the 3D world from a single image. This understanding includes the geometry and the semantic information of different objects from a single image. The understanding process is performed through proposing and building intelligent models, namely deep learning models. These models can learn the 3D surrounding environment from the available labeled data and extract useful information of different objects that appear in the images.

1.1 Motivation

Our eyes are among the most sensitive and significant organs we use to discover the surrounding environment. They can capture ten gigabits of information per second from the surrounding world, and the brain can process the captured information within less than a second [7]. This information helps the brain to learn and understand the world around us. Computer vision is the science of granting computers the ability to learn and experience the captured visual data (images and videos). It is the process

of gaining high-level information to understand the presented objects in the world from images and videos. From a single image, we can identify a person in the airport, distinguish between objects' shapes and colors, judge if the presented objects are far or near, and complete the missing parts of partially occluded objects. This process of understanding the surrounding environment is known as ***Scene Understanding***. However, granting the computer the ability to understand the world from images is a challenging task, and many algorithms have been proposed to achieve human performance and robustness in understanding the surrounding 3D world.

Under the scene understanding task, many sub-tasks can lead to a complete understanding of a captured scene. An image contains one or more objects that have different sizes and shapes. These objects have some meaningful relationships that help to describe the captured scene. The classification task is one significant task related to scene understanding. It is the task of assigning an expressive label to each object found in the image. This label classifies the objects to one of the pre-defined groups. In this way, we can describe the scene by knowing the names of the objects. Moreover, localizing objects in the image and estimating their viewpoint leads to more scene understanding. Object depth information also is a useful feature that describes how far the objects are from the camera. It helps in distinguishing between different objects by the depth values. The reconstruction task of objects is the process of estimating the geometry of the object and capturing its shape and appearance. It enriches the scene with more detail for a better understanding. For example, if we have a navigation agent asked to perform a specific task in an indoor environment, this agent has to perform multiple tasks to accomplish the requested task. This includes: 1) understand the objects in the environment and their orientation, 2) determine the near and the far objects to control the motion speed and navigate through the environment safely, and 3) estimate the 3D shapes of the objects for accurate manipulation. In summary, the mentioned tasks are considered as separate tasks in computer vision. However, they may overlap and complement each other to achieve better scene understanding.

Besides computer vision, Machine Learning (ML) plays a significant role in solving many computer vision tasks. After introducing deep learning in computer vision for

solving image recognition tasks [8], many researchers have formulated the problems as functions that can be learned from the target data. The contributions of this dissertation are to propose object-oriented deep learning-based models that solve different scene understanding tasks from a single image. More specifically, the proposed models tackle the problem of scene understanding from the geometric and semantic perspectives. From the images, we are trying to look at the objects and reconstruct them geometrically as well as recognize them semantically. By geometry, we are interested in the shape and size of the objects. It describes the shape and the structure of the objects in the 3D world. This information includes object pose, depth, and shape. Another aspect of understanding is using semantics. Semantics is the language used to describe the objects in the scene and the relationships among them. It includes object classification, localization, and segmentation. These properties help understand the surrounding environment and influence many practical applications such as medical imaging, augmented reality, and autonomous driving.

In this dissertation, we study how to formulate deep learning models for scene understanding from a single image. The understanding process is performed from an object-oriented perspective. It includes:

- Object classification and viewpoint estimation: it is a multi-task problem where we classify the objects that appeared in the image (classification) and where they look (viewpoint estimation).
- Monocular depth estimation: it is a task where we have to estimate how far away the objects are and try to generate high-quality depth map images.
- 3D Object Reconstruction: it is the task of estimating the shape of an object and visualizing it in the 3D world.

1.2 Challenges

We rely on deep learning models to propose the methods for solving the object-oriented scene understanding task. It is a powerful tool for extracting useful features from the

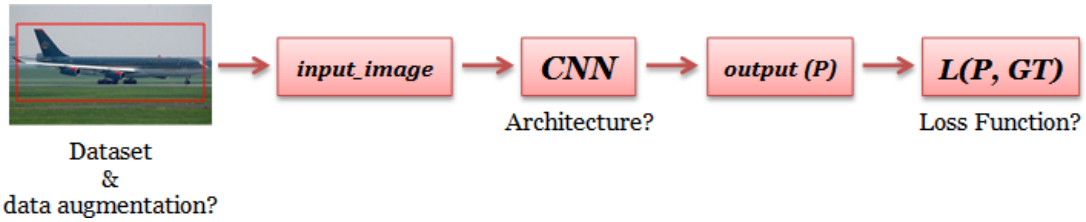


Figure 1.1: An overview of the challenges appeared when proposing a deep learning model. L is the loss function, P is the predicted output, and GT is the ground-truth.

available data. Mostly, the promising models were proposed to solve the classification tasks, *e.g.*, AlexNet [8], VGGNet (Visual Geometry Group Network, from Oxford University) [9], and Residual Net (ResNet) [10]. Applying these models to solve new tasks from different domains is not a straightforward process. Also, suggesting other models for solving different problems in computer vision is challenging. The challenges occur because of some factors such as the availablility of datasets for training, the output representations, the loss function, and the proposed model. Figure 1.1 demonstrates the mentioned challenges.

To approach a given problem and propose a suitable model to solve it, we focus on defining and formulating it through the following factors:

- **Problem Understanding:** The first step when solving a task using deep learning is to understand and define the given task. Task understanding includes defining the nature of the problem, such as a classification or a regression task. Understanding the given task helps in choosing the data representation for both the input and the output data. For example, when the task is considered as classification, the output can be represented as one-hot vector or vector of probability values. In Chapter 3, we treat the viewpoint estimation task as a classification problem where we sample the viewpoint angles of the objects into equal ranges and consider each range as a class.

- **Dataset:** Finding a suitable dataset to train a convolutional neural network (CNN) model impacts the accuracy of the solution. The dataset needs to have well-annotated in most cases and enough samples to ensure the model generalization. We can also apply different data augmentation techniques to overcome the lack

of data issue. Mixed data, containing real and synthetic samples, are useful to increase the training dataset and to improve the model performance. In Section 3.5, we explain how the synthetic data can be used to increase the training data and to improve the model performance.

- **CNN Architecture:** After defining the problem and proving the labeled dataset, we have to think about the suitable architecture to solve the given task. The architecture depends mostly on the nature of the task. For example, when the problem is considered a classification task, the model has to generate an output with probability values for each class, and the final result is the class with the highest probability value. When the task is a regression task, the model should generate an output with continuous values such as depth values. Additionally, the model may consist of many stages or sub-networks according to the definition of the problem. For example, a depth estimation model may contain an encoder part to extract the features and a decoder part to generate the depth map image. In Section 4.3, we propose two different models to solve the same task, and how one architecture outperforms the other one on the same conditions. So, it is crucial to think about the properties and the characteristics of the model when designing it for a given task.

- **Loss Function:** It is significant to select the suitable loss function depending on the given task. The purpose of the loss function is to optimize the network weights during the training process. It measures the error between the predicted output and the corresponding ground-truth. It should fulfill some conditions, such as it should be differentiable and robust against noise. In Section 4.3.3 and Section 5.3.2, we demonstrate different loss functions and report how they can be applied to optimize the model parameters and improve the model performance. The adequate loss function helps the model to converge very fast during training and improves the performance. Generally, we modify and update the selected loss function to overcome some issues, such as the generated noise in the output and the imbalanced data problem.

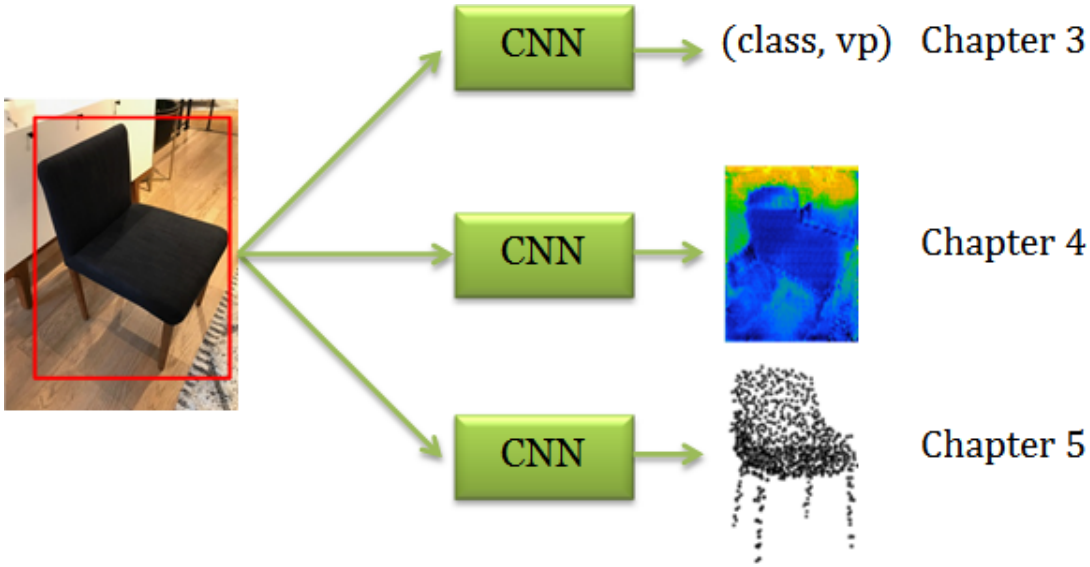


Figure 1.2: Dissertation contributions: different CNN models for scene understanding focusing on the objects. VP is the object viewpoint and class is the object class.

There are no clear answers to formulate these factors. Each task has its definition and the conditions that should be considered while solving it. We approach the given tasks by considering these factors and defining them to propose a suitable CNN model.

1.3 Dissertation Contributions and Organization

The contributions of this dissertation are in the field of deep learning. We explain how to formulate different computer vision tasks to solve them using CNNs. As illustrated in Figure 1.2, we start all tasks from a single image of an object detected in the given image. Then, we propose the CNN model and train it to solve a specific task and generate the desired output. We can summarize the main contributions of this dissertation in the area of object-oriented scene understanding as follows:

- A novel multi-task CNN model that classifies the detected object in an image and estimates its viewpoint angle simultaneously. The model was trained completely on synthetic data and tested on real data.

- Two different CNN models to estimate the depth information from a single image. Each model generates plausible depth map images but with different resolutions. We study the impact of using different loss functions to optimize the

models' parameters. We conclude that some loss functions can deal with noise and outliers.

- A novel and simple CNN model that can infer the 3D shape of an object from a single image. The model has a simple architecture and utilizes a spherical point cloud to generate the final 3D shape. The model was trained on synthetic data and tested on both synthetic and real data.

We now provide the outline of the dissertation and give a brief overview of each chapter. For each chapter, we review the related work of the chapter's topic.

Chapter 2 reviews the literature background of several related topics, *i.e.*, single-view reconstruction, deep learning for computer vision, and CNN-based model for multiple computer vision tasks.

In Chapter 3, we address the problem of joint object classification and viewpoint estimation. We propose a multi-task CNN architecture that performs object classification and viewpoint estimation simultaneously. The first part of the CNN is shared between the two tasks, and the second part is two subnetworks to solve each task separately. We use synthetic images to increase the training dataset to train the proposed model. To evaluate the proposed model, we use the PASCAL3D+ dataset [1] to test our proposed model, as it is a challenging dataset for object detection and viewpoint estimation.

In Chapter 4, we introduce the task of depth estimation from a single image using deep learning. Depth estimation plays a vital role in many computer vision tasks, including scene understanding and reconstruction. We propose two different CNN models to estimate the depth information from a single image. The first model is a fully-convolutional model that generates depth values with a resolution lower than the original input resolution. To solve this issue, we design a simple fully-convolutional encoder-decoder CNN that generates an output with a resolution equal to the input image resolution. For robustness, we leverage a non-convex loss function that is robust to the outliers while optimizing the network parameters.

Chapter 5 focuses on the problem of 3D object reconstruction from a single image. We propose a simple, yet powerful, CNN model that generates a point cloud of an

object from a single image. 3D data can be represented in different ways. Point clouds have proven to be a common and simple representation. The proposed model was trained end-to-end on synthetic data with 3D supervision. It takes a single image of an object and generates a point cloud with a fixed number of points. An initial point cloud of a sphere shape is used to improve the generated point cloud.

Chapter 6 concludes the dissertation and suggests some promising directions for future work.

2

Background

All things are difficult before they are easy.

Thomas Fuller

In this chapter, we will give an overview of the concepts and methods that are the most relevant to this dissertation.

We will start with a brief presentation of the field of 3D reconstruction from multi-view and single-view perspectives. After the introduction, we will present and review different types of image cues and priors that were used in the single-view reconstruction process to overcome the ill-posedness of this problem.

We will then present an intensive introduction to applying ML techniques in solving computer vision tasks. First, we will explain the supervised and unsupervised learning concepts and their application. Then, we will explain the main building layers of CNNs in detail. Finally, we will present the application of CNNs in solving scene understanding related tasks such as classification, detection, and regression problems.

2.1 Multi-view and Single-view 3D Reconstruction

Inferring 3D geometry of an object or a scene can be done using multiple images or a single image. Multi-view reconstruction is to estimate the 3D geometry of a model

from a set of images. The core of image creation is a projection from a 3D scene onto the 2D plane. 3D reconstruction is a reverse process of generating 2D images from 3D models or scenes. When two images are available, the position of a 3D point can be estimated by the intersection of two projection rays of two corresponding points, and it is referred to as triangulation.

Multi-view reconstruction from static images has been a hot topic in the field of computer vision for a long time. Many researchers proposed different methods to tackle this problem. **Silhouette-based method** is one of the earliest proposed methods for multi-view reconstruction. From the 2D silhouettes, 3D shapes can be reconstructed by projecting them back from the corresponding known viewpoints and intersecting the projection rays [11, 12]. In the case of multiple images with unknown or arbitrary-distributed viewpoints, voxel-based matching is used, and the photo hull is defined by iteratively carving the voxels with high photometric errors [13]. This method is known as **space carving**. However, it is not robust because of the high sensitivity to the outliers, and it does not work in convexities.

Other methods were proposed to generate **point clouds** or **meshes** to represent the 3D geometry. Given multiple images of the same scene, point clouds are computed based on image feature matching [14]. The process of computing the points' positions is repeated, and the number of points is increased. Then some points are filtered out due to inconsistencies in the feature matching. The generated dense point cloud is then used to create a mesh using Poisson surface reconstruction [15].

With the absence of multiple images from different views of a scene, inferring 3D geometry from a single image becomes an ill-posed problem. Given a single image, not enough information is available to recover the depth information or directly estimate the scene geometry. In the literature, various methods were proposed to infer 3D geometry with specific assumptions [16].

Next, we will review some image cues and priors that are used in different reconstruction methods to overcome the ambiguity and the ill-posedness of the problem. A detailed overview of single-view reconstruction approaches is presented in Section 5.2.2.

2.2 Image Cues and Priors for Single-View Reconstruction

Due to the ambiguity and the ill-posedness of the single-view reconstruction problem, an infinite number of candidates can potentially be the correct reconstruction of a given scene. To overcome this issue, local cues of the input image and prior knowledge are used to compensate for the lost information. Furthermore, the class-domain of the input image, the user-interaction, and the application-domain are used to narrow and restrict the generated results. Here, we discuss the image cues and priors that help to solve the ambiguity of the problem and generate plausible 3D geometry of the scene.

Shape from Shading (SfS) [17] is one of the earliest methods for single-view reconstruction. SfS infers a surface from a single gray level image by using the gradual variation of shading induced by the surface interaction of light. It also estimates the light conditions and reflection properties.

Contour Edges are salient structures in the image induced by discontinuities, object boundaries, or reflectance changes. They are used to give evidence for geometry and the position of objects. Corners and junction points are also important for single-view reconstruction. Shape from Contour [18] methods aim to estimate geometry information from the object contours alone. However, the reconstruction becomes difficult when the surfaces are smooth and don't exhibit adequate contour lines in the image.

Silhouette is a featureless projected image of an object or a scene from 3D to 2D that shows the general shape of the object. Silhouette cue is used to find the geometric reconstruction of an object such that the projection of the object into the image plane agrees with this silhouette. The disadvantage of the silhouette is that there are always many objects that are silhouette consistent if it is used alone for reconstruction. For general curved surfaces, silhouette-based methods are mostly used to recover the 3D models from a single view [19, 20].

Texture and Location of an object may be used as information for reconstruction. If an object is assumed to have a regular and known texture, it is possible to infer the

geometry of the object. The location of objects can allow inferring semantic knowledge about the objects. For example, the ground, the floor, or the sky can be identified by their location, and this can be helpful for the 3D reconstruction. In [21], texture and location cues are combined to reconstruct the objects by distinguishing their position and texture.

Besides the mentioned image cues, priors are important in single-view reconstruction to overcome the problem of ill-posedness. There are high-level and low-level priors, and depending on the reconstruction goals and the target object, different priors or a combination of them can be applied to estimate the 3D model.

Smoothness is the small partial change of some property. In single-view reconstruction methods, a dense reconstruction is difficult. Therefore, surfaces are chosen to be smooth and indicating the consistency between the object surfaces. This property plays a vital role in reconstructing curved objects [22].

Geometric Relations are very helpful before the reconstruction process, especially in man-made environments. By assuming planes to be parallel or perpendicular, one can derive some parameters, and the reconstruction process tends to be easier. Also, knowing that an object is symmetric can help us to determine the geometry of the object. It can be seen as a weak multi-view reconstruction that provides more information [23].

Semantic Relations allow inferring high-level information on the relative position and the structure of different objects in the image. For example, inferring occluded points based on semantic knowledge can improve the reconstruction, *e.g.*, the leaves are connected to the plant [24]. The knowledge of object location, *e.g.*, the sky and the ground, is an important feature that helps to start the reconstruction. And it ensures consistency between different objects in the reconstructed scene.

Shape Priors impose high-level knowledge on the objects to be reconstructed, and they are the most restrictive priors on objects. They limit the applicability of the proposed methods and usually work without the need for user input. They can be defined or learned. In [25], a shape grammar was defined for the reconstruction of facades. It limits the proposed method to reconstruct buildings. In contrast, the

proposed method in [26] used a database to learn the shape priors on some objects. So, the reconstructed objects are restricted to the classes in the pre-defined database.

2.3 Machine Learning in Computer Vision

ML is a field of Artificial Intelligence (AI) that extracts features from input data and uses statistics to perform a specific task. ML systems learn to extract features from examples implicitly. The learned features are extracted without controlling the types of features (edges or texture), and they don't follow hand-crafted rules. ML systems take data of input and the corresponding ground-truth and generate a model that can be used to generate similar output from new data.

To build an ML system, there are different choices concerning the available data and the training method. ML systems can be categorized into supervised, unsupervised, and reinforcement learning.

Supervised Learning is used when input examples and the corresponding labels are available. The model tries to learn a function that maps the inputs to the labels. This mapping function can then be used to estimate the label of unseen data. Artificial Neural networks (ANN) and random forests are examples of supervised learning methods.

In some cases, it is hard to collect or find a dataset that has both the inputs and the corresponding labels. **Unsupervised Learning** is the solution for this case. Given the input examples, and with the absence of the corresponding labels, the model attempts to automatically discover structure in the data by extracting useful features. Clustering algorithms are a good example of unsupervised learning models.

When the input data and the corresponding labels are absent, **Reinforcement Learning** is applied. In reinforcement learning, the model interacts with a dynamic environment that has a current state and action to perform a specific task. It receives feedback for each decision it takes and updates its next action to maximize the objective function that measures how well the model performs. This method is out of this dissertation' scope.

2.4 Convolutional Neural Networks

CNNs are well-known deep learning architectures biologically-inspired by the natural visual perception mechanism of living creatures. The history of CNNs started with the experiments conducted by Hubel and Wiesel in 1959 [27]. They found that the cells in animals' visual cortex are responsible for detecting light in the receptive fields. In 1990, LeCun *et al.* [28] published a paper where they described the modern framework of CNN. They introduced a neural network called LeNet-5 which is used to classify the handwriting digits. They used the backpropagation algorithm to train the neural network. However, due to the lack of data and the computation power at that time, the proposed networks couldn't perform well in large-scale problems. After that, many researchers have developed methods to overcome some problems encountered in training deep CNNs. In 2012, Krizhevsky *et al.* [8] proposed a deep CNN architecture to solve the problem of image classification. They showed significant improvements and outperformed results upon previous methods. They won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [29]. The proposed architecture was similar to LeNet-5 but with a deeper structure. It is known as AlexNet. With the success of AlexNet, many CNN architectures have been proposed to improve performance and get more accurate results such as ZFNet [30], VGGNet [9], GoogleNet [31], and ResNet [10]. ResNet is the deepest architecture. It is 20 and 8 times deeper than AlexNet and VGGNet, respectively. It is observed that deeper architectures are better for classification tasks. The reason behind this is that the network can approximate the output, because of the increasing nonlinearity allowing to extract more representative features. This section was published in [32].

2.4.1 General CNN Architecture Blocks

Many CNN architectures have been proposed for solving different computer vision tasks. The general building blocks are similar, and they are different in some layers depending on the tasks they want to solve. For example, LeNet-5 [28] and AlexNet [8] are almost similar in the general building blocks, but AlexNet is deeper than

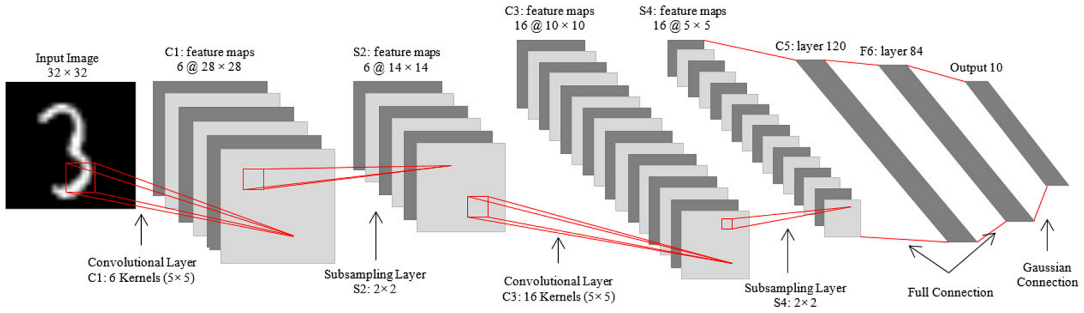


Figure 2.1: LeNet-5 Architecture.

LeNet-5. The main blocks of the CNNs are convolutional layers, pooling layers, and fully connected layers (FC). Each block has its own specifications and features, and it can vary from one architecture to another according to the problem to be solved. In the following, we will explain each building block of the CNNs in detail. Figure 2.1 shows LeNet-5 architecture [28].

Convolutional Layer. Convolutional layers are the main building blocks of the CNNs. They learn the feature representation of the input data (images in our case) by performing convolutions over the inputs. The convolutional layer consists of several kernels that are used to compute different features from the input images. They ensure that the local connectivity-neurons are connected to a small region of the input which is known as the receptive field. The extracted feature maps are calculated by convolving the input with the kernels and then add the bias parameters to the feature. The convolutional layer has many kernels, and they are applied to the input image to calculate the output feature map. Each kernel is shared by all special locations of the input. The advantage of weight sharing is to reduce the complexity of the model and the training process of the network becomes easier. Figure 2.2 illustrates the convolution operation. Mathematically, consider x as the input image, W is the kernel, and b is the bias for the convolutional layer. The feature map z generated from this layer is calculated as:

$$z = Wx + b \tag{2.1}$$

Many researchers proposed different types of convolutional layers to improve feature representation and to learn some kind of invariance. Tiled CNN [33] is one kind of the

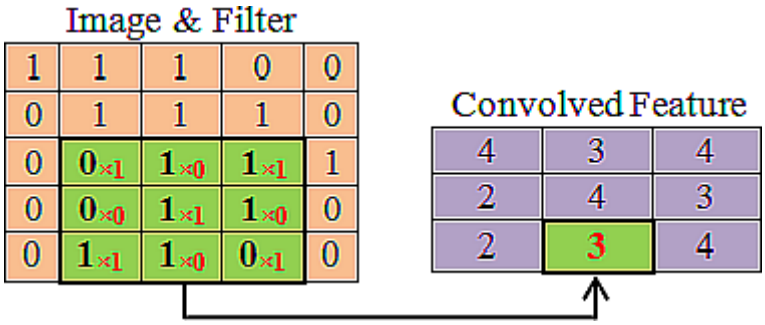


Figure 2.2: A convolutional operation with a 3×3 filter.

enhanced convolutional layers that tiles and multiplies different feature maps to learn rotation and scale-invariant features. Dilated CNN [34] is another recent development of the standard CNN where more hyper-parameters are introduced to the convolutional layer. This strategy enhances the performance for tasks that need a large receptive field when they make the prediction, such as scene segmentation and speech synthesis and recognition. Also, there are other improved convolutional layers such as Network in Network (NIN) [35] for classification task which replaces the normal convolutional layer with a multilayer perceptron convolution layer (mlpconv). It is like a micro neural network with more complex structures rather than a conventional convolutional layer that uses linear filters followed by a nonlinear activation function. NIN enhances the model discriminability for local patches within the receptive field for the classification task. Transposed Convolution [30] is another type of convolutional layer that is used to reconstruct the input again in regression tasks such as image segmentation, depth estimation, visualization, and image super-resolution. In the literature, this layer is called the deconvolutional layer. The deconvolutional layer first upsamples the input to a specified factor and then performs normal convolution operation on the upsampled result.

Activation Function. CNNs have some linear components and nonlinear components. The activation functions are the nonlinear components that follow the convolutional layers to introduce the nonlinearities to the CNN to detect the nonlinear features and to improve the CNN performance. Rectified Linear Unit (ReLU) is one of the most popular activation functions used in CNNs. It has been shown that CNNs

can be trained efficiently using ReLU [36]. ReLU is defined as:

$$a = \max(z, 0). \tag{2.2}$$

where z is the input to the activation function and a is the output. ReLU keeps the positive part of the input and prunes the negative part to zero. Another version of ReLU is leaky ReLU (LReLU) [37] that defines a parameter λ in range $(0, 1)$ to compress the negative part rather than mapping it to zero. Mathematically, LReLU is defined as:

$$a = \max(z, 0) + \lambda \min(z, 0). \tag{2.3}$$

This makes a small and non-zero gradient when the unit is not active (negative value).

Exponential Linear Unit (ELU) is another activation function that enables faster learning of CNN and improves the accuracy of the classification task. Like ReLU and LReLU, ELU [38] sets the positive values to identity to avoid the vanishing gradient problem, and the negative part is used for fast learning and it is robust to noise. Mathematically, ELU is defined as:

$$a = \max(z, 0) + \min(\lambda(e^z - 1), 0). \tag{2.4}$$

where λ is a controlling parameter to saturate ELU for negative inputs.

A different activation function with respect to the output is the sigmoid function (σ) [39]. Sigmoid function is used often in the ANNs to introduce the nonlinearity in the model. It takes real numbers and squashes them into range $(0, 1)$.

Mathematically, the sigmoid function is defined as:

$$a = \sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.5}$$

In general, the activation functions are applied after the convolutional layers in CNN to add the nonlinearity and project the values from one range to the desired range. It means that the CNN can approximate functions that don't follow linearity and successfully predict the class of a function which has a non-linear decision

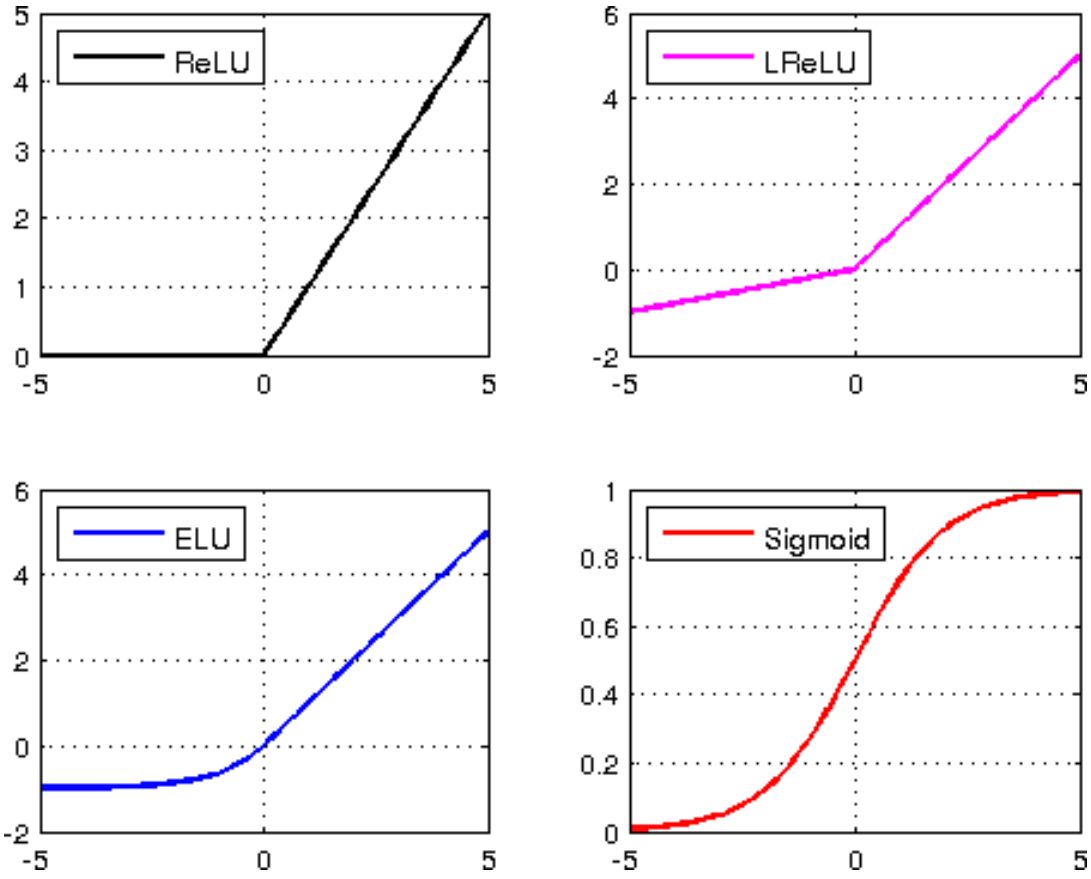


Figure 2.3: Different activation functions.

boundary. In fact, it is hard to find a physical world phenomenon that follows linearity straightforwardly. Therefore, the non-linear functions help us to understand and approximate the non-linear phenomenon. The "**Activation Function**" term is biologically inspired. In the real brain, neurons get signals from other neurons and decide whether or not to fire by taking the cumulative input into account. This decision, based on the cumulative input, is the output from the activation function. Figure 2.3 shows the activation function we have discussed.

Pooling Layer. The purpose of the pooling layer is to ensure the shift-invariance and lowers the computational burden by reducing the resolution of the feature maps. It is usually placed after the convolutional layer. It takes the feature map which is generated from the convolutional layer and outputs a single value for each receptive field (pooling window) according to the pooling operation. The pooling layer performs max-pooling [40] or mean (average) pooling [41]. Figure 2.4 shows different pooling

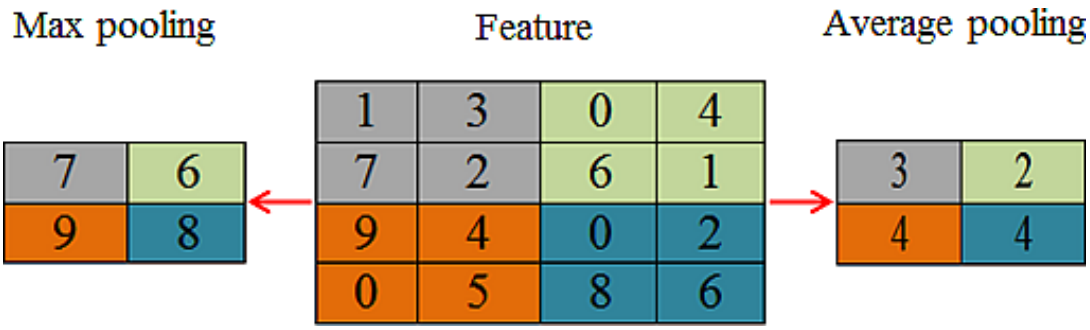


Figure 2.4: Different Pooling Operations with 2×2 filters and stride 2.

operations. Also, there are other versions of pooling layers proposed for some tasks, such as Spatial Pyramid Pooling (SPP) [42] that can generate a fixed length of features regardless of the input size.

Fully Connected Layer (FC). In classification tasks, FC layers [8] are used at the end of the CNN after the convolutional layers and the pooling layers. FC layers aim to generate specific semantic information. The neurons in the FC layers have a full connection to all neurons in the previous layer. It can be considered as a special case of a convolutional layer where the receptive field size is equal to one. Usually, dropout is used after the FC layers to avoid CNN from overfitting.

Regularization. One of the most problematic issues regarding CNN training is overfitting. Overfitting happens when the model fits too well to the training dataset, and it cannot generalize to new examples that are not in the training dataset. So, overfitting is an unneglectable problem in deep CNNs [43]. There are many proposed solutions to reduce the overfitting effectively, such as Dropout [44], $L1$ regularization, and $L2$ regularization [43].

In deep learning, Dropout is widely used as regularization after the FC layers. It deletes or deactivates some neurons so that not all connections between the layers are activated at that time during training. It is not preferable to add them in the first layers because dropout causes information to get lost. And if the information is lost in the first layers, it will be also lost for the whole network and this will affect the performance of the network. During testing time, dropout layers are bypassed and they are not active.

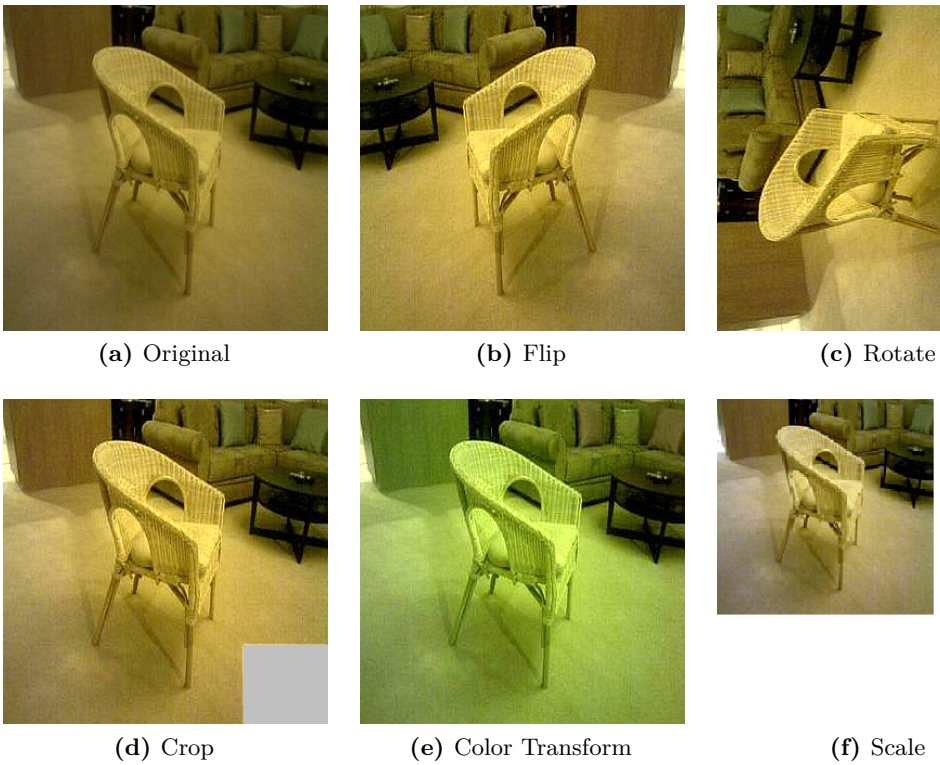
2.4.2 Data Augmentation and Optimization

To train a CNN, many steps should be done such as preparing the training dataset, designing the CNN and initializing the weights, choosing the loss function, and training the CNN using backpropagation to optimize and update the network weights.

It is well known that training a deep CNN mostly depends on the availability of the training data. The more training data we have, the better-trained model we can get. However, in some applications such as medical problems, the available datasets are small and we cannot rely on them to train a deep CNN. To overcome this issue, data augmentation [45] is applied to the training dataset. The purpose of the data augmentation is to increase the number of training samples by transforming the images into new data without altering the nature of the content. The commonly used data augmentation methods include simple geometric transformations such as mirroring, shifting, scaling, rotating, and spatial cropping [8]. This method not only helps to overcome the problem of training data scarcity, but also trains the network and updates its weights more accurately. Figure 2.5 illustrates an example of applying data augmentation on an image.

After preparing the training data, global data normalization is usually applied on the training dataset to transform the data to zero-mean and unit variance. However, during training and as the data flow deeper through the network, the distribution of the input data changes which affects the training process and the network accuracy. Batch Normalization (BN) is applied to the input data in some layers to avoid this problem [46]. It fixes the means and the variances of the layer inputs by computing them after each mini-batch rather than the entire training set.

Deep CNN has numerous parameters to be optimized. A proper initializing of these parameters is important for fast convergence and to avoid the vanishing gradient problem. If the weights start with small values, the input signal shrinks as it passes through each layer and it drops to a very low value, and becomes no longer be useful. If the weights start with large values, the input signal starts to increase and it becomes too large and useless. Initializing the network with the right weights can help the



(a) Original **(b)** Flip **(c)** Rotate

(d) Crop **(e)** Color Transform **(f)** Scale

Figure 2.5: Applying different data augmentation techniques to an image.

network to converge in a reasonable amount of time. Many methods have been proposed to initialize the weights. In [8], the network weights were initialized from a zero-mean Gaussian distribution with a standard deviation of 0.01. Another weight initialization method is Xavier [47]. The idea in the Xavier method is to initialize the weights from a Gaussian distribution with zero mean and a variance of $\frac{2}{(n_{in}+n_{out})}$, where n_{in} is the number of neurons feeding into the layer and n_{out} is the number of the neurons that the result is fed to. Xavier method and its improved version allow deep networks to be trained and they converge fast. The motivation for Xavier initialization is to initialize the weights with values so that the activation functions can generate meaningful values that are not saturated or begin from dead regions. In other words, the weights are initialized with random values that are neither large nor small.

To optimize the CNN, the backpropagation algorithm is used to train the CNN which uses gradient descent to update the CNN parameters. Among the optimization methods, Stochastic Gradient Descent (SGD) [48] is commonly used to estimate the gradient on the basis of a single randomly picked example (x^i, y^j) from the training

dataset:

$$\theta_{i+1} = \theta_i - \eta_i \nabla_\theta \mathcal{L}(\theta_i; x^i, y^j). \tag{2.6}$$

where θ is the network parameters, x^i is the input, and y^j is the output. $\mathcal{L}$ is the objective function that is used to train the network, and η is the learning rate. In practice, the network parameters are updated with respect to mini-batch. To guarantee the convergence and speed up the learning process, momentum is proposed to make the current gradient depends on historical batches [49]. The classical momentum update accumulates a velocity vector in the relevant direction, and it is given by:

$$\begin{aligned} \nu_{i+1=} &\gamma \nu_i - \eta_i \nabla_\theta \mathcal{L}(\theta_i; x^i, y^j); \\ \theta_{i+1} &= \theta_i + \nu_{i+1}. \end{aligned} \tag{2.7}$$

where ν_{i+1} is the current velocity vector and γ is the momentum term which is usually set to 0.9. Nesterov momentum [50] is another way of using momentum in gradient descent optimization that moves in the direction of the previously accumulated gradient, calculates the gradient, and then updates the parameters.

2.4.3 CNNs in general Computer Vision Applications (Classification, Detection, Segmentation)

CNNs were proposed widely in computer vision to solve many problems. These problems include image classification, object localization and detection, depth estimation, semantic segmentation, and action recognition. These models were proposed to achieve state-of-the-art performance, method generalization, and to outperform traditional methods that depend on hand-crafted features. Following, we introduce recent CNN models that were applied to solve different computer vision tasks.

Many CNN models have been applied to solve image classification problem. The breakthrough results achieved by AlexNet [8] encouraged researchers to propose different models that achieved significant improvements in classification accuracy. These models have either reduced the filter size or proposed deeper networks (*e.g.* VGG and ResNet).

Moreover, the publicly available datasets, such as ImageNet and Pascal VOC, helped in improving the models' accuracy.

Moreover, object detection has been studied in deep learning. In [51], Girshick *et al.* has shown that the learned features from classification problem can be transferred and used to solve the problem of object detection. Region-based CNN (R-CNN) was proposed where candidates were extracted from the input image using Selective Search (SS) [52] to narrow the search space of the objects in the image. The candidates were fed to the network and classified between the background or one of the target classes. After that, Faster R-CNN [53] was proposed to overcome the problem of generating the candidates. In the proposed model, a region proposal network (RPN) was introduced for object proposal generation and achieves further speed-up. Some models were proposed to perform the object detection learning problem in one shot. YOLO (You only look once) [54] dealt with the problem of object detection as a regression task. It generates the bounding boxes and predicts the object class in a single network. This speeds up the detection process as the model detects, localizes, and classifies the objects directly from the input image.

Also, CNN was applied to solve the problem of image segmentation. Image segmentation is the task of dividing the input image into multiple consistent regions. In contrast to object classification or detection, image segmentation is a pixel-level task and each pixel is labeled and the pixels with shared properties, such as color, texture, intensity, *etc.*, are then merged to create one region. One of the leading network architectures proposed for image segmentation and dense prediction was Fully Convolutional Network (FCN) [55] in 2014. From its name, one can see that this architecture is composed of convolutional layers without using FC layers. This allows the segmentation image to be generated from any input image with arbitrary size. After that, different architectures have been proposed for image segmentation. U-Net [56] is an encoder-decoder architecture that was proposed to segment medical images. The encoder part extracts the features from the input image and reduces the resolution of the input image through the pooling layers, and the decoder part reconstructs the image and recovers the object details with the help of the skip connections between the

encoder and the decoder layers. SegNet [57] is another architecture proposed for image segmentation. Authors introduced the transferring of max pooling indices instead of the whole values from the encoder to the decoder using skip connections to improve the segmentation resolution for faster training.

In summary, CNN has proved its ability to solve different computer vision tasks and outperformed traditional methods that depend on hand-crafted features. Also, it has been applied to 2D data and extended to 3D data to perform various tasks.

3

Object Classification and Viewpoint Estimation

A complex system that works is invariably found to have evolved from a simple system that works.

John Gall

3.1 Introduction

One fundamental task in computer vision aims to understand, describe, and extract useful information from an input image of a scene. Formally, this task is known as **Scene Understanding**. It can be tackled by solving sub-tasks such as object classification and viewpoint estimation. Object classification and viewpoint estimation have become popular research topics in computer vision field because of their wide applications. Addressing these two tasks at the same time is beneficial to describe an object under the general object recognition task [58]. For example, a robot that manipulates with objects and carries them from one place to another needs to know the object category and its viewpoint. This information can help it to correctly execute a target task without human supervision. Also, selecting a piece of furniture from a

massive database to design an office or a living room needs to infer the correct object pose and the object category. **Object Classification** is the problem of assigning the correct label to the object in an image. This problem concerns many object classes with different visual instances. For better object understanding, viewpoint estimation is an important step in many applications, such as image retrieval and model matching. **Viewpoint Estimation** is the problem of estimating the view angle with respect to the camera.

Human visual system can recognize different objects of the same class with different viewpoints easily, and it can differentiate between various classes by matching these objects with the correct classes. However, some computerized vision systems can recognize specific objects, but they have trouble in learning and understanding more object categories. Even among some learned classes, these systems find difficulties in recognizing and classifying some objects because of the changes in lighting conditions, occurrence in different poses, or occurrence in cluttered or occluded environment [59].

From a given image, we are interested in detecting and localizing the objects in the image using R-CNN [60]. The detected objects are then categorized and the pose information is predicted. Object pose can either be a relative pose or an absolute pose. The relative pose is the rigid transformation between the object and the camera. The absolute pose is the rigid transformation with respect to the canonical coordinate system. The 3D pose is about object translation and orientation in R^3. Viewpoint estimation is the prediction of the azimuth angle, the elevation angle, and the distance between the object and the camera (as shown in Figure 3.1). In our case, we are interested mostly in estimating the azimuth angle of the object.

Object classification and viewpoint estimation problems have been studied as separate problems intensively. However, finding a standalone system that is capable of performing both tasks simultaneously is difficult because these tasks have opposite directions in terms of feature representation. For object classification, the system has to learn invariant features with respect to object viewpoint. So, it can easily classify the same object that appears in different poses. With regard to estimating the viewpoint of an object, the system has to learn a representation that preserves the geometric and

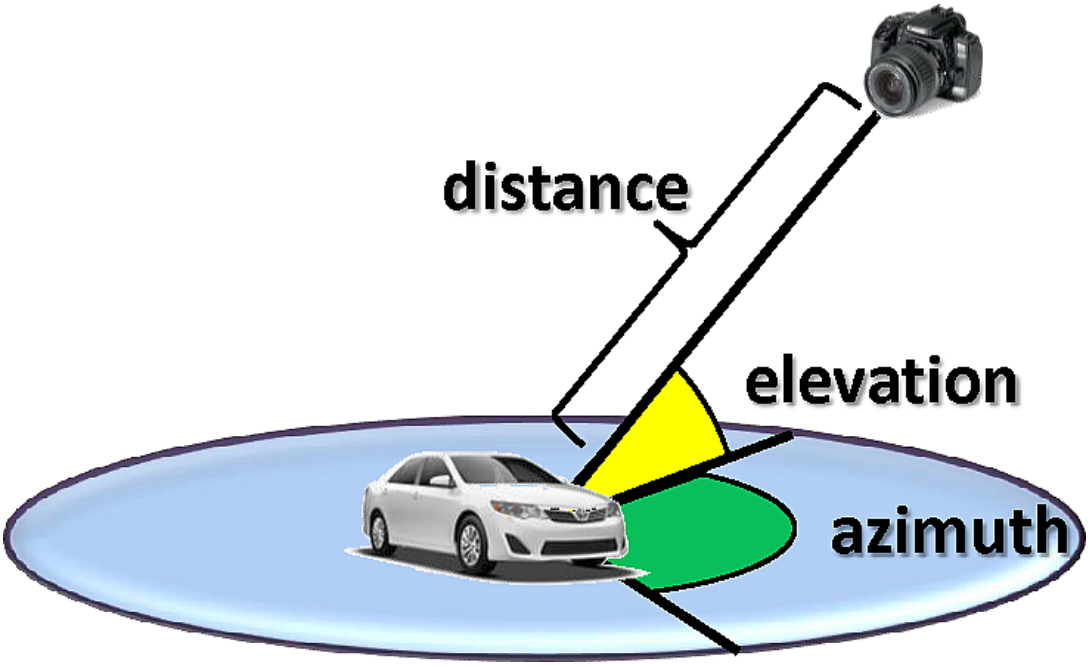


Figure 3.1: 3D pose of an object. Figure from [1] website.

the visual information in order to distinguish between different viewpoints of the same object [61].

Classical object classification methods extract hand-crafted features for different categories firstly, and then train a classifier to classify objects of different categories. Such features used are Histogram of Oriented Gradients (HOG) [62], Scale-Invariant Feature Transform (SIFT) [63], and Speeded-Up Robust Feature (SURF) [64]. With the rise of deep learning models, many computer vision tasks have been solved using CNNs such as object recognition and detection [51], segmentation [55], and object depth estimation [65]. These problems have been considered either classification problems or regression problems. The models have been proposed to solve a single task, and they have shown impressive results. They were pre-trained to perform a specific task and then fine-tuned to perform another task, which is known as transfer learning [66]. Extending these architectures to solve multiple tasks at the same time can be done, but careful design is needed. This means that some layers of the network will be shared for both tasks and other layers will be separated.

In deep learning, collecting or finding a sufficient dataset to train a model for a specific task is a challenging issue. Training a CNN model needs a huge number of

images. One solution is to fine-tune a pre-trained CNN. In practice, very few people train a CNN from scratch nowadays. This is because training a CNN from scratch needs a dataset of sufficient size. Instead, it is common to use the weights of a pre-trained CNN, that was trained on a very large dataset such as ImageNet, and fine-tune them on the new task. To transfer the knowledge from an existing CNN to a new one, there are two options. **The first option** is to use the pre-trained CNN as a feature extractor for the new task. That means, taking a pre-trained CNN, removing the last FC layers, freezing the weights of the remaining network, and using it as a feature extractor. On top of this model, one can train a classifier for the new dataset to perform the new task. **The second option** is not to train a classifier on top of the feature extractor model, but also to fine-tune the whole model end-to-end for the new dataset to perform the new task. Also, it is possible to select and fine-tune some layers and freeze the remaining ones. In [66], it is explained that earlier layers extract generic features that are applicable to many datasets and tasks. So, freezing them can boost the training process when fine-tuning the model. What we discussed is called ***Transfer Learning***. It is the idea of utilizing the learned knowledge of a model that solves a task and using it to solve related ones.

With the availability of large-scale online 3D models repositories (*e.g.* ModelNet [67] and ShapeNet [68]), a huge number of images with known viewpoints can be rendered which can be used for training. In order to make the synthesized images as real ones, the synthesized images can be overlaid with real images as a background image. This step helps the CNN to train on synthesized images, similar to the real images, and to overcome the lack of data issue [3].

The contributions of this chapter are summarized as follows. We propose a multi-task CNN model that jointly solves object classification and viewpoint estimation tasks. We use a complete synthesized dataset rendered from 3D objects with rich annotations to increase the training set and to train the CNN for both tasks. Also, we build a class-dependent subnetwork for the viewpoint estimation task that takes care of estimating the viewpoint and depends on the object class. Our proposed model showed

impressive results in both tasks and is comparable to the state-of-the-art results. The results reported in this chapter were published in [69].

3.2 Related Work

Object classification and viewpoint estimation have been studied recently, especially with the evolution of deep learning methods in solving computer vision tasks.

3.2.1 Object Classification

In [8], the authors proposed a CNN model, now it is known as AlexNet, to solve the problem of object classification. They submitted their results to the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [29]. They achieved the top results in the competition. The model was deep consisting of successive convolutional layers, with activation functions and max-pooling layers, and FC layers.

After the impressive results of AlexNet, several networks have shown remarkable improvements and scored higher classification accuracy by modifying the layers' filter size [30] or going deeper with the network [9, 10]. In [31], the authors proposed a deeper model where they used a well-designed convolutional block named the Inception block. The proposed model, named GoogleNet, achieved outstanding results on the image classification task. In [10], He *et al.* proposed a very deep model consisting of 152 layers. To overcome the difficulties of training very deep models, they reformulated and arranged the layers and forced them to learn residual function. Since that, the new structure of this arrangement is called residual blocks.

Moreover, many proposed work solved the classification task with the help of object part annotation. They benefit object part information to improve classification accuracy. Barnson *et al.* [70] proposed a method to detect object parts and extract features using CNN from multiple pose-normalized regions. The proposed method extracted features from different image regions and integrated lower-level and higher-level features to improve classification accuracy. Lin *et al.* [71] proposed a recognition system composed of part localization, alignment, and classification and named it Deep

LAC. The model comprises three sub-networks to perform each task. The output of each sub-network is forwarded to the next sub-network. First, object part locations are estimated. Then, the alignment sub-network takes the localization results and performs template alignment. Finally, the classification sub-network predicts the object class depending on the aligned template. The drawbacks of the above-mentioned methods are the difficulties of finding well-annotated datasets and that they are applicable for specific categories and cannot be generalized for more classes.

In [72], the authors fine-tuned a CNN pre-trained on ImageNet dataset to compute mid-level image representation from images different from ImageNet dataset and performed object classification on Pascal VOC dataset [73]. Also, [42] obtained state-of-the-art results in object detection and classification by training new FC layers on the top of convolutional layers of a network previously trained on ImageNet dataset. They introduced the Spatial Pyramid Pooling layer (SPP) that is flexible enough to handle different scales, sizes, and aspect ratios of the image.

3.2.2 Viewpoint Estimation

Object orientation is an important geometric feature of the objects in images that can be used for 3D reconstruction. While some previous works dealt with the problem of object viewpoint estimation as a regression problem, we consider this problem as a classification task using CNN. The main limitation of considering the viewpoint estimation task as a regression problem is the ambiguous representation of the different viewpoints for some objects. For example, the table has almost a symmetrical shape that makes the estimation of the viewpoint unclear. Moreover, estimating the viewpoint of a bottle is almost unattainable as it looks the same from all directions.

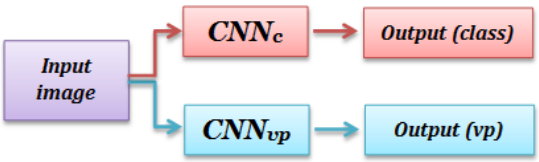
Previous methods focused on estimating the object viewpoint of a single object class. They considered simple models of objects without considering the large intra-class variations. Also, they didn't generalize their methods to handle different object categories because the dataset annotation was insufficient.

To estimate the object pose, many proposed methods predicted 2D keypoints and used them to estimate the pose. Pavlakos *et al.* in [74] used 2D keypoints that are

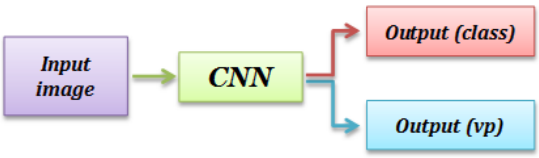
semantically meaningful projected points of the 3D models. From the input image, the proposed method predicted a probability map of the 2D keypoints and estimated the 3D pose by comparing it with some pre-defined models. Instead of using the semantic 2D keypoints, other methods used 2D keypoints that are the projection of the 8 corner points of the 3D bounding box surrounding the target object. These methods trained a network to find the correct 2D keypoints locations and compared them with the corresponding points projected from the 3D keypoints of the ground-truth to estimate the pose [75]. The estimated pose is used then as a prior to retrieving the closed 3D model that accurately represents the object geometry that appeared in the input image. However, these methods require 3D models with the annotated 3D keypoints and the corresponding 2D projection of them to estimate the pose.

Recently, PASCAL3D+ dataset [1] has been introduced as a challenging dataset for object detection and pose estimation. It augments 12 rigid objects with 3D annotations. Most related works used this dataset to evaluate their models on object detection and viewpoint estimation. The output is considered to be correct if the object detection part is correct (the bounding box overlap is larger than 50%) **AND** the viewpoint estimation part is correct. R-CNN [51] and Fast R-CNN [76] are mostly used as ready detectors to detect the objects first and then estimate the viewpoint. In [5], the authors considered the viewpoint estimation as a classification problem and trained CNN to predict the viewpoint. To evaluate their model, they used R-CNN to detect the objects and the detected regions were used to estimate the viewpoint. Also, [6] solved the problem of detection and pose estimation in a single fast shot. They combined the detection and the pose estimation at the same level by extending the fast SSD detector [77] to estimate object pose at the same time. To overcome the scarcity of training data to train the CNN for solving the problem of viewpoint estimation, [3] proposed to use synthetic images in training. They rendered images from 3D online model repositories and mixed them with real images for training. We adopt this method to train our model for both object classification and viewpoint estimation.

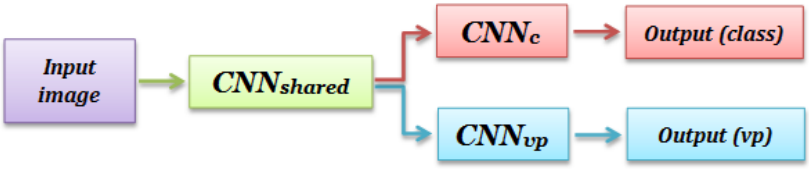
In sum, object classification and viewpoint estimation tasks have been studied and the proposed methods have attained good results. Mostly, the two tasks are considered



(a) Parallel Models



(b) Shared Model



(c) Multi-Task Model

Figure 3.2: Different network setups for joint object classification and viewpoint estimation. (a) Independent and parallel models for each task. (b) A shared model for both tasks and the output layer splits into two layers; one for each task. (c) A multi-task model that has a shared network and branches into two networks; one for each task.

separately and CNN models are created for each task. Conversely, we consider the two tasks by proposing a new multi-task CNN architecture that performs object classification and viewpoint estimation simultaneously. We also train our model using a synthesized image dataset and test it using a real image dataset.

3.3 Problem Statement

Designing a multi-task model is not a straightforward process. The tasks' representation and the extracted features influence the proposed model. Looking deeper into the targeted problems, the object classification task and viewpoint estimation task have opposite feature representations. For the object classification task, the extracted features should be viewpoint invariant to classify the object correctly. For the viewpoint estimation task, the learned features should preserve the visual features and learn to distinguish different viewpoints of the same object.

One solution to solve the given tasks is to train independent and parallel models for each task, as shown in Figure 3.2a. The advantage of this choice is that there is flexibility in selecting and designing the models for each task. The models are independent and different for each task. However, the drawback of this setup is that it doesn't leverage the fact that the same image representation can be shared with different tasks [78].

To overcome this limitation, a multi-task model can be proposed to solve the two tasks by splitting the last layer into two layers; one for each task. The model consists of a shared network to extract common features for both tasks and branches at the end of the network into two task-specific layers, as illustrated in Figure 3.2b. A serious issue with this setup is that one task-specific layer is not enough to learn useful features for each task [79]. Moreover, the viewpoint is an intrinsic object feature, and a single layer estimating the viewpoint of different object classes is theoretically inaccurate.

To address the limitations of the above-mentioned setups, we propose a multi-tasks model that has a shared network and task-specific networks as shown in Figure 3.2c. The shared network extracts general features that can be shared across multiple tasks. The task-specific network is trained to extract specific features to solve the target task.

3.4 Multi-Task CNN Architecture

We adopt the well-known CNN architecture introduced by [8] and extend it to solve our problem. The original architecture consists of five consecutive convolutional layers followed by three FC layers and it was trained on ImageNet dataset to classify 1000 object classes.

Figure 3.3 shows the proposed multi-task model. This architecture is divided into two parts; a shared part and a task-specific part. The shared part consists of five convolutional layers and it acts as a feature extractor for both tasks. After the fifth convolutional block, the model branches into two sub-networks, one for each task. Each branch consists of three FC layers.

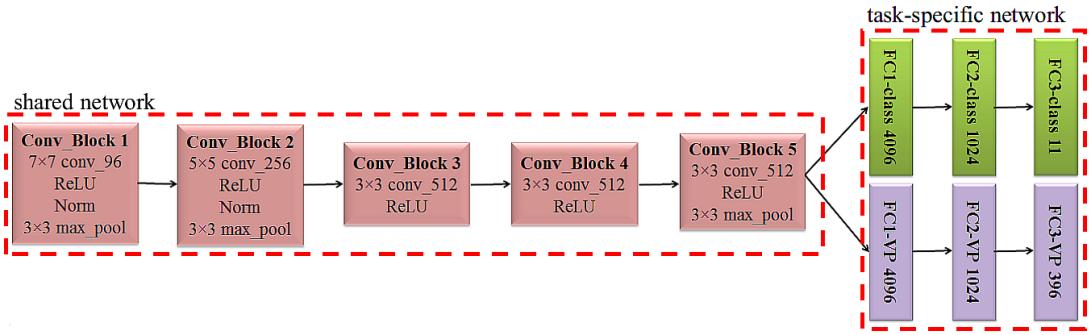


Figure 3.3: The proposed CNN architecture. ReLU is the activation function we use for the nonlinearity. Max-pooling is used in the first, second, and fifth convolutional blocks. The FC viewpoint output layer is an object category dependent layer. FC layers are followed by a dropout layer with a rate of 0.5.

The loss function L that is used to train the proposed model has a classification term and a viewpoint term. Formally, it is defined as:

$$L(W) = \lambda_c \cdot loss_c(x, l^c) + \lambda_{vp} \cdot loss_{vp}(x, l^{vp}) \tag{3.1}$$

where the ourput is generated from a softmax function and $loss_c$ and $loss_{vp}$ are a cross-entorpy loss function of the object classification task and the viewpoint estimation task, respectively. x is the input image, and l^c and l^{vp} are the class label and the viewpoint label, respectively. λ_c and λ_{vp} are parameters to balance the training process between the two tasks. They are set to 1 as we have separated task-specific networks after the shared network that they have completely independent parameters (the FC layers). W is the CNN weights to be learned and optimized. We apply max-pooling after the first, second, and fifth convolutional layers. Max-pooling layers are used to reduce the computation time and to control overfitting.

In [66], the authors demonstrated that early layers in CNNs extract generic features, while the last ones are original-dataset-specific layers. Features from earlier layers can be utilized as general features for different tasks. The last layers extract specific features that help in solving a specific task. In our proposed model, the convolutional blocks extract the generic features and the FC layers are task-specific layers. So, the network branches after the fifth convolutional block. The new sub-networks contain

rich and specific features that represent the objects for a specific task as shown in Figure 3.3.

3.5 Implementation Details and Training Dataset Preparation

We used MatConvNet [80], a MATLAB toolbox implementing CNNs for computer vision application, to implement and evaluate our proposed model. The weights of the shared layers are initialized using the corresponding weights in VGG-m model [81], which was pre-trained on the ILSVRC data for image classification. The weights of the classification subnetwork layers are initialized from the same network while the viewpoint subnetwork weights were initialized randomly. We fine-tuned the subnetworks using backpropagation. Stochastic gradient descent (SGD) algorithm was used to optimize the network weights with the following settings: the momentum is set to 0.9, and the weight decay is set to 10^{-5}. The learning rate is initialized to 10^{-3} and is decreased by 10 when the validation error doesn't change.

With reference to the object classification and viewpoint estimation, we train our model on 11 object classes which are introduced in the PASCAL3D+ dataset [1]. For the viewpoint estimation task, it is known that the nature of the viewpoint is continuous, and many research works deal with this problem as a regression problem [82, 4]. However, we consider the task as a classification problem. More specifically, we focus on estimating the azimuth angle and we divide the viewpoint range into 36 classes (10 angles in each class). We excluded the bottle class from our experiments as it is very hard to estimate the viewpoint. It has always a symmetrical shape, and the viewpoint solution would belong to an infinite set.

Regarding the training data, CNNs are always hungry and need a massive amount of data for training. The publicly available datasets don't have accurate and rich annotations for both tasks (classification and viewpoint estimation). Also, collecting and annotating images for both tasks is a hard and expensive task. With the availability of large-scale online 3D model repositories [68], it is possible to utilize them to generate

a complete and well-annotated dataset under full control. We use these 3D models to render object images with different orientations. The rendering process helps to introduce more images to train the CNN for object classification and viewpoint estimation. We can control the rendering process and generate a huge number of images. Rendered images provide a reasonable number of images for CNN to be trained well [3]. Also, we performed data augmentation on the rendered images which resulted in increasing the training data. To make the rendered images more realistic, we overlaid the rendered images with real images as a background to guide the network towards convergence and to avoid the wrong classification. Figure 3.4 shows the process of preparing the training set. The proposed model was only trained on a complete synthetic images. We didn't use any real images for training. In Section 3.6.2 and 3.6.3, we show how the proposed model performs on the real images.

3.6 Experimental Results

In this section, we will introduce the testing datasets that we used to evaluate the proposed model. Also, we will present the experimental results for object classification and viewpoint estimation. Moreover, we will compare our results against other methods that considered the same problems.

3.6.1 Datasets

ImageNet is a large-scale dataset of over $15M$ labeled high-resolution images belonging to roughly 22,000 categories [29]. The images were collected from the web and labeled by humans. ILSVRC is an annual challenge that uses a subset of ImageNet with roughly $1.2M$ training images, $50K$ validation images, and $150K$ testing images. All these images are belonging to 1000 categories, each contains almost 1000 images.

ImageNet consists of variable-resolution images. As our proposed model requires a constant image dimensionality, the images were resized to the desired resolution of 224×224. Following the traditional method in resizing [8], the images were rescaled

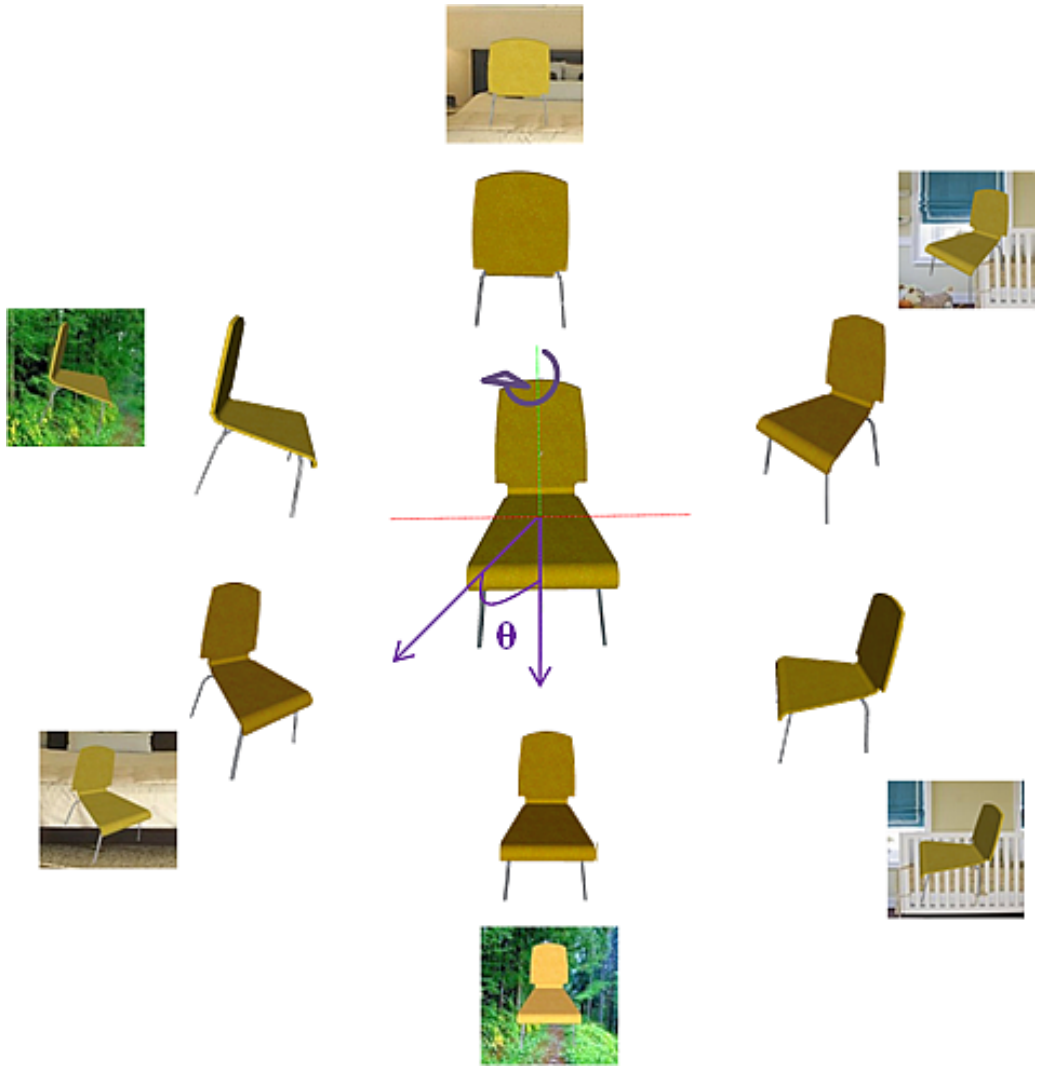


Figure 3.4: Preparing the training data. We render many images from the synthetic 3D models from different orientations. The rendered images are overlaid with real background to mimic the real data.

so that the shorter dimension was of length 224, and then a centered rectangle was cropped from the image with a size of 224×224.

PASCAL3D+ [1] is a well-known challenging dataset for object detection and viewpoint estimation captured in the wild. It contains 12 rigid categories of PASCAL VOC 2012 [73] with rich 3D annotations. That is, each object is annotated with its viewpoint (azimuth angle, elevation angle, and distance from the camera) and bounding box values surrounding the object in the image. Furthermore, more images were added from ImageNet [29] for each category with rich annotations, and we use them to evaluate our model on the object classification task.

Figure 3.5: Examples of ImageNet (first two columns) and PASCAL3D+ (last two columns) datasets.

Table 3.1: Object Classification Performance on ImageNet dataset and PASCAL VOC 2012 val dataset.

Dataset	aero	bicycle	boat	bus	car	chair	d.table	mbike	sofa	train	tv	mAP
ImageNet	99.0	94.3	98.4	94.5	97.4	72.1	97.7	90.8	86.3	95.1	97.9	**93.1**
VOC 12 val	89.6	81.7	79.1	81.0	77.5	76.1	65.6	78.9	58.0	78.6	85.8	**77.5**

Figure 3.5 shows some example images of the ImageNet dataset and the PASCAL3D+ dataset. From the visualized images, we can notice that the PASCAL3D+ images are more challenging than ImageNet images. In ImageNet images, the objects are clearly seen and the image almost contains one object. In PASCAL3D+ images, the images usually contain many objects from different classes (Figure 3.5, fourth row, last column), and they are occluded or some parts of the object are missing (Figure 3.5, eighth row, last column).

3.6.2 Object Classification Results and Comparisons

We tested our model on the ImageNet dataset for object classification on the same object classes that are introduced in the PASCAL3D+ dataset (11 object classes). Also, we test it on Pascal VOC 12 (the validation set) on the same object classes. Table 3.1 reports the quantitative results of our proposed model. We use the mean Average Precision (mAP) as a metric to evaluate our model.

From Table 3.1, we notice that our proposed model can classify the objects accurately. The proposed model was completely trained on synthesized images and tested on real images. The results show that the synthesized images can be used to train CNNs and improve the performance of the trained model. Also, the trained model generalizes very well to the real images. We got a 93.1% classification mAP when we consider the maximum output value from CNN on the ImageNet test set.

Moreover, we conducted another experiment to evaluate the performance of the proposed model using the PASCAL VOC 2012 val dataset on the same object classes introduced in the PASCAL3D+ dataset. This dataset is challenging because the images were captured in the wild, and each image contains many objects of different classes. We use the ground-truth bounding box to extract the objects from the input image and resize them to fit the CNN model input size. Table 3.1 reports the performance of

our model on the PASCAL VOC 2012 val dataset. We notice that the chair and the table classes record low accuracy. The reason behind this is that they appear mostly together in the same image. Also, it appears as a cluttered or occluded object when we extract the object. We can conclude that the proposed model performs well for object classification.

To validate our proposed model and the training procedure, we compare it against other methods. We use the PASCAL VOC 2007 test set as most of the literature uses this dataset for comparison. Our proposed model solves the object classification and viewpoint estimation problems simultaneously. We compared our results against [83], [72], and [84]. In [83], the authors proposed a deep learning framework in weakly supervised settings that can classify multiple objects in a single image and perform image annotation. The authors in [72] proposed a method to exploit the image representations learned by CNNs, trained on the large-scale annotated dataset, to other recognition tasks. They used the layers trained on the ImageNet dataset to extract mid-level features from the PASCAL VOC dataset and trained new layers for object classification problem. That is, they have applied the transfer learning concept to exploit the pre-trained layers to extract generic features and train new layers on a different dataset for the same task. Razavian *et al.* [84] used the features extracted from the OverFeat network [85] as generic features to solve many object recognition tasks, such as object classification and scene recognition. For each task, they selected a suitable dataset according to the task. After that, a linear SVM classifier was applied to the extracted features from the network. We have to point out that the previously mentioned works used the whole 20 object classes in PASCAL VOC 2007 test dataset to test their proposed methods. In our work, we just train and test the proposed model on the object classes introduced in the PASCAL3D+ dataset, which are 11 object classes. Also, the images we used in training are synthetic images to overcome the lack of training data problem. Table 3.2 demonstrates the comparison between the results reported in the previous works and our results. The proposed model scored 80.9% mean average precision (mAP), and it outperformed the previous works in the object classification task on most classes.

Table 3.2: Object Classification results and comparison with other methods on PASCAL VOC 2007 test dataset.

Category	Wu *et al.*, 2015 [83]	Oquab *et al.*, 2014 [72]	Razavian *et al.*, 2014 [84]	**Ours**
aero	**93.5**	88.5	90.1	90.4
bicycle	83.4	81.5	84.4	**86.7**
boat	83.6	82.0	**84.4**	76.9
bus	81.6	75.5	73.4	**84.3**
car	86.6	**90.1**	86.7	87.5
chair	54.5	61.6	61.3	**77.2**
d.table	53.8	67.3	69.6	**73.3**
m.bike	79.0	80.0	80.0	**81.0**
sofa	63.7	58.0	67.3	**67.4**
train	**91.1**	90.4	89.1	77.1
tv	80.4	77.9	74.9	**87.5**
mAP	77.4	77.5	78.3	**80.9**

3.6.3 Viewpoint Estimation Results and Comparisons

To evaluate the proposed model on the viewpoint estimation task, we first checked the best configuration of representing the viewpoint classes. We compared two different setups of the proposed model for viewpoint estimation with respect to the output layer. The first setup of the model was the same proposed model with respect to the viewpoint sub-network as shown in Figure 3.3 where the last FC layer is a class-specific layer. That is, we divided the last FC layer into class-based viewpoint bins. Each group of bins is reserved for the viewpoint of a specific class. We denoted this model by *specific_model*. The second model shared the same output layer between all classes. So, it didn't care about the object class when the model estimates the object viewpoint. We denoted this model by *general_model*. We trained both proposed models on synthetic images and tested them on PASCAL VOC 2012 val dataset. We used the ground-truth bounding box to extract the object from the image during testing, and we used the Average Precision (AP) as a metric to compare the performance of the models. We performed the experiment on 4 different viewpoint categorizations as introduced in [1].

Figure 3.6 shows the results of both models, and we can clearly notice that the class-specific layer at the end of the sub-network performs better and more accurately than when using the same FC layer without embedding the object class. This is because of the variety of geometry information between different classes and even between the objects in the same class.

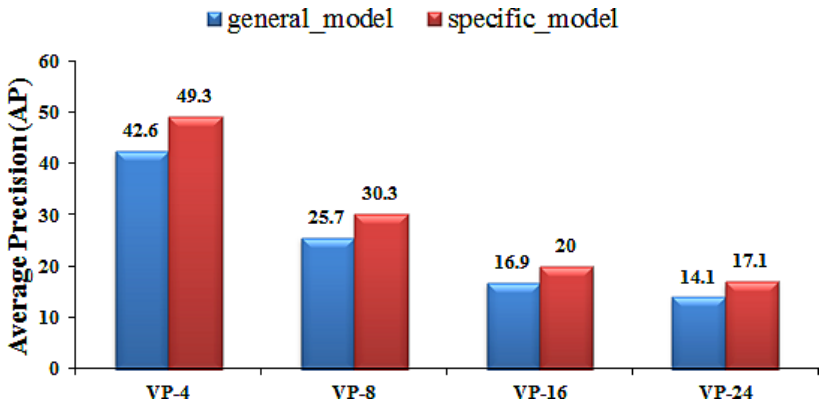


Figure 3.6: Comparison of Different Viewpoint Discretization between different Viewpoint Estimation Models. The viewpoints are discretized into 4 (VP-4), 8 (VP-8), 16 (VP-16), and 24 (VP-24) bins.

Moreover, we conduct an experiment to validate the performance of our model on object viewpoint estimation and compare it against other models. We use PASCAL VOC 2012 val dataset as most of the previous methods used it to evaluate their models.

As introduced in PASCAL3D+, Average Viewpoint Precision (AVP) metric [1] was used to evaluate object detection and viewpoint estimation jointly. In computing AVP, the output from the model is considered to be correct if and only if the bounding box overlap that detects the object is larger than 50% **AND** the estimated viewpoint is correct. As the proposed model was trained to solve only the viewpoint estimation task without object detection, we use R-CNN [51] detector to generate the bounding box and detect the objects in the input image. Other methods used either their own detectors or other proposed detectors like Fast R-CNN [76]. Table 3.3 shows a detailed comparison between our method and other previous methods handling the same problem.

We compared our model against the following models. DPM-VOC+VP [2] used a modified version of DPM to predict viewpoint. It was trained on real images from PASCAL3D+ VOC 2012 train set. Render for CNN [3] used R-CNN for object detection and it was trained on synthetic and real images. Viewpoints & Keypoints [5] proposed a model to estimate the object viewpoint and leverage it for object keypoint prediction. The authors in [6] proposed a fast model that detects the object using the proposed SSD detector and estimates its pose, and they achieved comparable results with the state-of-the-art. In [4], the authors considered the viewpoint estimation as a regression

problem as the pose space is continuous. They trained VGG-m model to estimate the viewpoint. As reported in Table 3.3, our model outperforms the other methods on four different quantization cases of 360-degree views (the viewpoint is divided into 4, 8, 16, and 24 bins respectively, with increasing difficulty). The proposed model was trained on synthetic data to perform two different tasks simultaneously and utilized earlier layers to extract generic features for both tasks. Pascal3D+ dataset is a challenging dataset that contains images captured in the wild. Our proposed model succeeded to generalize on real images and it performed accurately on the real images.

3.7 Conclusion

In this chapter, we considered object classification and viewpoint estimation tasks. Solving both tasks simultaneously improves the performance of many applications such as in robotics and for better scene understanding. We presented a new multi-task CNN architecture that has shared layers performing as features extraction layers for both tasks and separated sub-networks for each task. Owing to the opposite nature of the two tasks, the branching is necessary. Object classification task requires viewpoint invariant features while viewpoint estimation task requires capturing the variations of the viewpoint for different objects of different classes. We also trained our network on synthesized images and this helped us in solving the problem of the lack of data. Our results in both tasks showed that the proposed model has high accuracy on both tasks and is comparable to the state-of-the-art methods.

Table 3.3: Viewpoint estimation results and comparisons against other methods, considering viewpoint estimation problem, on PASCAL VOC 2012 val dataset. The methods are referenced as follow: DPM-VOC+VP [2], Render for CNN [3], VP-Regression [4], Vps & Kps [5], and Fast SSD [6].

Mthods	aero	bicycle	boat	bus	car	chair	d.table	mbike	sofa	train	tv	**Avg**
Joint Object Detection and Viewpoint Estimation (**4 View AVP**)												
DPM-VOC+VP	37.4	43.9	0.3	48.6	36.9	6.1	2.1	31.8	11.8	11.1	32.2	23.8
VP-Regression	52.43	50.80	19.74	61.24	46.82	20.85	20.31	50.60	42.01	53.42	53.11	42.85
Render for CNN	54.0	50.5	15.1	57.1	41.8	15.7	18.6	50.8	28.4	46.1	58.2	39.7
Vps & Kps	63.1	59.4	23.0	69.8	55.2	25.1	24.3	61.1	43.8	59.4	55.4	49.1
Fast SSD	64.6	62.1	26.8	70.0	51.4	11.3	40.7	62.7	40.6	65.9	61.3	50.6
Ours	58.4	60.8	29.1	62.1	50.3	37.6	41.5	59.1	55.6	55.9	51.3	**51.1**
Joint Object Detection and Viewpoint Estimation (**8 View AVP**)												
DPM-VOC+VP	28.6	40.3	0.2	38.0	36.6	9.4	2.6	32.0	11.0	9.8	28.6	21.5
VP-Regression	42.98	37.96	13.18	41.59	38.66	16.13	12.55	37.94	33.19	43.00	40.43	32.51
Render for CNN	44.5	41.4	10.1	48.0	36.6	13.7	15.1	39.9	26.8	39.1	46.5	32.9
Vps & Kps	57.5	54.8	18.9	59.4	51.5	24.7	20.5	59.5	43.7	53.3	45.6	44.5
Fast SSD	58.7	56.4	19.9	62.4	45.2	10.6	34.7	58.6	38.8	61.2	49.7	45.1
Ours	49.6	55.9	22.2	60.8	44.7	32.2	31.2	55.4	46.1	53.1	50.3	**45.6**
Joint Object Detection and Viewpoint Estimation (**16 View AVP**)												
DPM-VOC+VP	15.9	22.9	0.3	49.0	29.6	6.1	2.3	16.7	7.1	20.2	19.9	17.3
VP-Regression	29.90	24.37	7.73	38.75	29.23	12.18	10.32	25.62	24.82	29.50	25.16	23.42
Render for CNN	27.5	25.8	6.5	45.5	29.7	8.5	12.0	31.4	17.7	29.7	31.4	24.2
Vps & Kps	46.6	42.0	12.7	64.6	42.7	20.8	18.5	38.8	33.5	42.5	32.9	36.0
Fast SSD	46.1	39.6	13.6	56.0	36.8	6.4	23.5	41.8	27.0	38.8	36.4	33.3
Ours	31.9	40.3	13.5	55.9	37.8	25.8	24.6	41.7	41.0	47.2	44.2	**36.7**
Joint Object Detection and Viewpoint Estimation (**24 View AVP**)												
DPM-VOC+VP	9.7	16.7	2.2	42.1	24.6	4.2	2.1	10.5	4.1	20.7	12.9	13.6
VP-Regression	21.71	14.21	5.62	29.16	25.15	9.16	6.98	18.94	15.47	26.38	17.97	17.34
Render for CNN	21.5	22.0	4.1	38.6	25.5	7.4	11.0	24.4	15.0	28.0	19.8	19.8
Vps & Kps	37.0	33.4	10.0	54.1	40.0	17.5	19.9	34.3	28.9	43.9	22.7	31.1
Fast SSD	43.2	29.4	9.2	54.7	35.7	5.5	23.0	30.3	27.6	44.1	34.3	28.8
Ours	26.4	30.7	11.2	53.9	34.1	23.2	23.3	33.3	37.3	45.0	40.2	**32.6**

4

Object Depth Estimation from a Single Image Using CNN

Life can only be understood backwards;

but it must be lived forwards.

Soren Kierkegaard

4.1 Introduction

Estimating depth information from images is a significant problem in many computer vision fields. Depth information is an important component for a better understanding of the 3D geometry of a scene. Many computer vision problems have proven to benefit from using depth information such as semantic labeling [86], pose estimation [87], scenes modeling [88], robotics [89], and virtual reality. Combined with the RGB images, depth information adds a significant meaning to understand the scene and provides a richer representation for the objects and scenes. Normally, RGB-D data are obtained by depth sensors such as Microsoft Kinect for indoor scenes or LiDAR sensor for outdoor applications. The data, collected using these sensors, were used to investigate the problem of estimating the depth from RGB images and incorporate the outcomes

to solve other related problems. However, it is an ill-posed problem and inherently ambiguous. Using one captured image of an object, it is difficult to map an intensity or color measurement into a depth value [90]. Prior knowledge and external information are required to formalize the problem and extract useful information for depth prediction. The importance of solving this problem with accurate prediction helps to improve many computer vision tasks, such as reconstruction [91] and recognition [92].

While for humans, inferring depth information and the 3D structure from a single image is effortless, it is a challenging task for computer vision systems due to lack of knowledge and information. Depth estimation from multi-view images has been extensively researched compared to the single-view scenario. One of the reasons is in the multi-view scenario, provided the accurate local image correspondences, depth information can be recovered. Structure-from-Motion (SfM) [93] is one of the promising methods that uses multiple images to estimate the camera poses, the local correspondences, and the depth. It takes multiple RGB images ($n > 1$) and estimates the depth map by matching the local correspondences in the images. In contrast, single-view depth estimation is inherently ambiguous and it requires the use of prior knowledge and cues such as object sizes and positions, line angels, *etc.* Prior knowledge and cues can restrict the scene environment such as parallel lines for the indoor scenes, the sky and the ground for the outdoor scenes, or assuming a box model for room scenes. Also, object position and size play an important role in depth estimation from a single view. These assumptions and cues restrict the considered scenarios and cannot be generalized for further data or even new tasks. Other non-parametric methods depend on retrieving similar models that try to align them with the input scene to infer depth information [94]. In recent years, researchers have incorporated different sources of information such as user annotations and labeling to perform depth estimation. Still, the mentioned methods depend on hand-crafted features to solve the problem of depth estimation from a single image and they generate many physically implausible results.

Recently, CNNs have shown a breakthrough performance in solving the problem of object classification. This success inspired many researchers to apply deep learning to solve different computer vision tasks. Starting from AlexNet [8] as a base network for

object classification, many deeper networks have been proposed to solve the same task such as VGGNet [9], GoogLeNet [31], and deep ResNet [10]. Moreover, U-Net model [56] and its different variations have mostly been used to solve the regression tasks. They are employed to learn implicit relations between RGB images and some semantic information, such as object detection and localization, scene segmentation, and depth estimation. In general, deep learning models outperform the traditional hand-crafted features methods (*e.g.* SIFT [63], HOG [62], and Fisher kernel (FK) [95]) in solving these problems as the CNNs learned useful features directly from the images. They scored higher records for a wide variety of computer vision tasks.

In this chapter, we present two different CNN models to estimate the depth information from a single RGB image. The first model is a Fully-CNN (F-CNN) where we directly predict the depth values using it. The resolution of the generated output from this model is smaller than the input image. The second model is a fully convolutional, encoder-decoder model inspired by V-Net [96], proposed for 3D medical image segmentation, with skip connections between the encoder and the decoder to generate a more accurate depth image. To target the complete meaning of fully-convolutional, the pooling layers were replaced with strided convolutional layers [97] in the encoder. The decoder network is mirrored from the encoder with additional layers; the upsampling layers (deconvolutional layers) and the concatenation layers. The skip concatenation layers, which perform as fusion layers, fuse features from the encoder part with the features from the upsampling layers in the decoder. In this way, the fine-grained features are transferred and fused with the decoder features, where the decoder features lose some information due to the upsampling operation. As a consequence, this step improves the quality of the predicted depth image. Both models were trained using a non-convex loss function named Tukey's biweight loss function [98] which is robust in regression tasks. The results presented in this chapter were published in [65] and [99].

4.2 Related Work

The first work on depth estimation was originally based on stereo vision where pairs of images of the same scene are used for 3D shape reconstruction. Most approaches for single-view depth estimation depend on different shooting conditions; such as Shape-from-Shading (SfS) [100] and Shape-from-Defocus (SfD) [101]. Generally, these methods utilize geometric or triangulation differences to estimate the depth information. With the absence of these cues in the single-view scenarios, estimating depth becomes a challenging task. Following, we review the related work of single-view depth estimation using both classical and deep learning methods.

Traditional methods, which rely on geometric information, impose hard assumptions such as modeling a scene of a room as a box model. And suppose that the objects are located on the floor and the walls have the properties of parallel planes in the indoor scenes [102]. For the outdoor scenes, the global geometric constraints are like the sky should be on top of the scene and the ground is on the bottom. Also, the size of the objects is large when they are near the bottom of the scene and very small when they are on the top. These methods and assumptions restrict the target applications and cannot be generalized in different scenarios. Another group of work comprises non-parametric approaches, *e.g.*, textons, SIFT [63], HOG [62], where features between the query image and the images in a database are matched to find the nearest neighbors and then perform depth transfer between the matched parts. Karsch *et al.* [94] used a kNN depth transfer utilizing SIFT Flow with a global optimization method to estimate the depth map. Liu *et al.* [103] defined the problem as a discrete-continuous Conditional Random Field (CRF) to learn the depth of different segmented superpixels. They assumed that the similar regions in the retrieved RGB images have similar depth cues. These approaches are well-scaled to the pre-defined database, but the inferred depth maps depend on the depth quality of the training images.

Moreover, supervised learning methods based on probabilistic graphical models have been investigated to infer the depth information. An early work by Saxena *et al.* [104] used a Markov Random Field (MRF) for predicting depth from a single image using

strong assumptions about scene geometry. The authors used superpixels to enforce the consistency between neighboring regions and improve their approach. Following this work, Liu *et al.* [105] combined the problem of semantic segmentation with depth estimation, where the predicted segmentation is used as a source of information to predict depth in order to guide the 3D reconstruction.

After the success of CNNs in classification tasks [8], researchers applied deep learning in solving the depth estimation problem. Eigen *et al.* [90] proposed a CNN to directly predict the depth from a single image. The model was multi-stage where a coarse depth was predicted from the first stage of the network and was combined with the output of the first convolutional layer in the second stage to infer the final depth map. The authors extended their model to a multi-task model that estimates depth, normals, and semantic labels from a single image [106]. In both studies, the first stage was initialized based on AlexNet [8] or VGG [9], and the second and the third stages were initialized as fully CNNs. More recently, Liu *et al.* [107] proposed a model that combines a deep CNN and CRFs to predict depth from a single image. The model learns the unary and pairwise potentials during training with CRF to model the relations of neighboring superpixels. The authors further improved their method by proposing an efficient training method based on a fully convolutional network and a superpixel pooling method [108]. Whereas most of the researchers defined the problem of depth estimation as a regression task, other researchers considered this problem as a classification problem. The success of the deep residual networks [10] in solving object classification problem has inspired Cao *et al.* [109] to define the depth estimation as a pixel-wise classification task. They first discretized the continuous depth values into multiple categories and label each category according to the depth range. They trained a fully convolutional deep residual network to estimate the depth label of each image pixel. To improve the output depth map, fully connected CRFs were applied as a post-processing step to enforce local smoothness interactions.

4.3 Methodology

The best practice when designing a CNN model for solving a problem is to consider these factors. ***The first factor*** is the nature of the problem. In our case, the depth estimation problem is considered as a regression problem. So, the proposed model should generate an image with continuous values for each image pixel. It is an image-to-image problem. ***The second factor*** is the training data and the values that should be generated from the output layer. For example, if the desired values are positive and bounded between 0 and 1, sigmoid activation function is the best to infer the output values. ***The third factor*** is choosing a suitable loss function. In deep learning, the loss functions are used to train and optimize the network parameters. Selecting a suitable loss function with respect to the problem and the training data helps in fast and better convergence. So, these factors affect how to design and shape the best model to solve the target problem. Following, we present the two proposed models in more detail.

4.3.1 Fully-CNN Architecture (F-CNN)

Most CNN architectures consist of consecutive blocks that perform linear and non-linear operations. These blocks extract useful features and decrease the input image resolution through the convolutions and the pooling operations, respectively. The proposed network is a fully CNN model as shown in Figure 4.1. It takes an RGB image as input and generates an image of 1/4-resolution of the original input image with continuous depth values. The proposed model is composed of five convolutional blocks. The first four blocks consist of two convolutional layers followed by ReLU as a non-linear activation function. The pooling layers are used in the first two blocks to decrease the feature resolution and to minimize the computational cost. The last block of the network consists of a convolutional layer followed by a sigmoid function to generate the depth image. Here, the sigmoid function is used to generate continuous values between 0 and 1. The kernel size of the convolutional layers is 3×3. We didn't use FC layers as they are useful in the classification tasks but our task is a regression

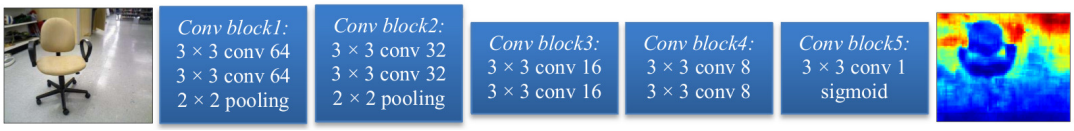


Figure 4.1: The proposed F-CNN Architecture. All convolutional layers have a filter size of 3 × 3. Pooling layers are used after the first and the second convolutional blocks. All activation functions after the convolutional layers are ReLU except in the last block, the activation function is sigmoid.

task. Although max-pooling layers affect the accuracy of the regression problems, it prevents the network from the overfitting.

As experienced, regression tasks require fewer filters with small kernel size and almost no need to use FC layers compared to classification tasks [110]. This reduces the number of parameters and the computational cost and extracts more powerful features from the input image.

As mentioned above, the output resolution is 1/4-resolution of the original input image. This is due to the using of the pooling layers in the first two convolutional blocks. This will affect the quality of the inferred depth image and generate some noise on the borders. To overcome this drawback, we propose a second model that can generate more accurate depth images and have the same input image resolution as well.

4.3.2 mini V-Net Architecture

The second model we proposed is inspired by U-Net [111] and V-Net [96] that were proposed for medical image segmentation. The proposed CNN model is an encoder-decoder model to infer the depth information from a single 2D image. The model is trained end-to-end from scratch and is a fully convolutional model in the encoder and decoder as shown in Figure 4.2.

The encoder consists of three fully consecutive convolutional blocks with feature sizes of 16, 32, and 64, respectively. In the first two blocks, we use two convolutional layers, and in the third block, we use three convolutional layers. The convolutional layers have a kernel of size 3 × 3 and a stride of 1. Each convolutional layer is followed by leaky-ReLU [112] as an activation function. We use the strided convolutional layers

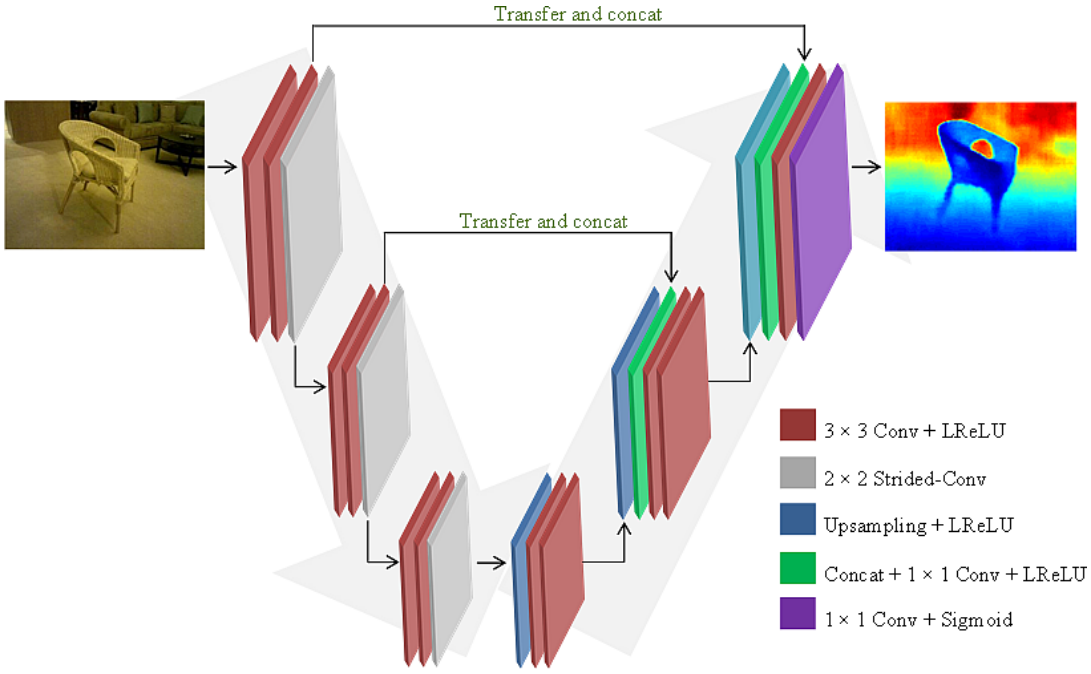


Figure 4.2: mini strided V-Net architecture. Pooling layers are replaced by strided-convolutional layers to generate an output of the same resolution as that of the input.

instead of the pooling layers to decrease the features' size. In [97], authors proved that the pooling layers can simply be replaced by convolutional layers with increased stride. The advantage of using the strided convolutional layers is that they can be easily reversed, trained, and tuned rather than fixing the layers to max or average operations. They are trainable layers and extract useful features. In the proposed model, the kernel size of the strided convolutional layers is 2×2 with a stride of 2. This decreases the size of the feature maps to a half.

The decoder has the same structure of the encoder, but with some additional layers that reconstruct the image again and generate the depth image with the same resolution as that of the input image. The sizes of the features for each convolutional block in the decoder are 64, 32, and 16, respectively. In particular, we add the upsampling (upconvolution) layers of 3×3 kernel size in the decoder part to reconstruct the features and generate the final depth image with the same resolution of the input image. LReLU is used as an activation function after each convolutional layer in each block. To generate a better depth image with more details, we concatenate the output of some layers from the encoder part with the corresponding output of the upsampling layers

from the decoder part as shown in Figure 4.2 (the concatenation links between the encoder part and the decoder part). After the concatenation, we apply a convolutional operation of 1×1 kernel size to fuse the concatenated feature maps. We notice that the concatenation layers (or as they are called the skip connections) add more details, and as a consequence, the objects in the output images have sharper edges and hence are less blurry. The depth image is generated using a sigmoid layer as the output range is between the interval of $(0, 1)$.

In summary, the encoder-decoder models have been introduced into many computer vision tasks such as semantic segmentation, image reconstruction, and optical flow estimation. They have significantly outperformed other models in solving the same tasks. In the results section, we will prove that the results generated from the encoder-decoder model are more accurate than the results generated from the F-CNN.

4.3.3 Loss Function

Selecting a suitable loss function plays a critical step in training a CNN. In our case, the loss function measures the error between the generated depth image and the ground-truth image to optimize and update the model weights. It should fulfill some constraints regarding the task and the nature of the training dataset. Our depth estimation problem is considered as a regression task and a straightforward loss function like L2 norm can be used to compute the error between the estimated values $\hat{y}$ and the ground-truth y [113].

For depth estimation, L2 norm (Eq. 4.1) is not robust to outliers (the large error calculated between the predicted depth and the ground-truth) [113]. Optimizing the model using L2 norm biases the training process towards the outliers because small errors (differences between the ground-truth and the predicted depth values) have little influence on the CNN weight modifications, while the large errors (outliers) incur a large penalty.

$$\ell_2 = \|\hat{y} - y\|_2 \tag{4.1}$$

To overcome this issue, we propose to use a non-convex loss function that is robust in regression tasks, namely, Tukey's biweight loss function (Eq. 4.3) [98]. The advantage of using this loss function is that the small residual values (the difference between the predicted depth and the ground-truth depth) influence the training process and it is robust to the outliers. During the training process, the loss function suppresses the influence of the outliers and sets the magnitude of the outlier gradients close to zero.

Formally, the difference between the ground-truth depth y and the estimated depth value $\hat{y}$ (*i.e.* the residual r) is calculated as:

$$r = \hat{y} - y \tag{4.2}$$

Given the residual r (Eq. 4.2), Tukey's biweight loss function is defined by:

$$\rho(r) = \begin{cases} \frac{c^2}{6}\left[1 - \left(1 - \frac{r^2}{c^2}\right)^3\right] & \text{if } |r| \leq c \\ \frac{c^2}{6} & \text{if } |r| > c \end{cases} \tag{4.3}$$

The first-order derivative of Tukey's biweight loss with respect to r is defined as:

$$\acute{\rho}(r) = \begin{cases} r\left(1 - \frac{r^2}{c^2}\right)^2 & \text{if } |r| \leq c \\ 0 & \text{if } |r| > c \end{cases} \tag{4.4}$$

In our work, the value of c is not constant during training. It changes in each gradient step according to the maximum error in the respective step. During training, c is set to be at 20% of the maximal residual in that gradient step. Another benefit of using Tukey's biweight loss is that it is differentiable and the training process converges better when the depth values are represented on a log scale. Transforming depth values to log space can minimize the contribution of regions with large depth values. It benefits the training because the regions with large depth values (far regions in the image) have less rich information for depth estimation. Figure 4.3 shows Tukey's biweight loss and its first derivative when $c = 1$.

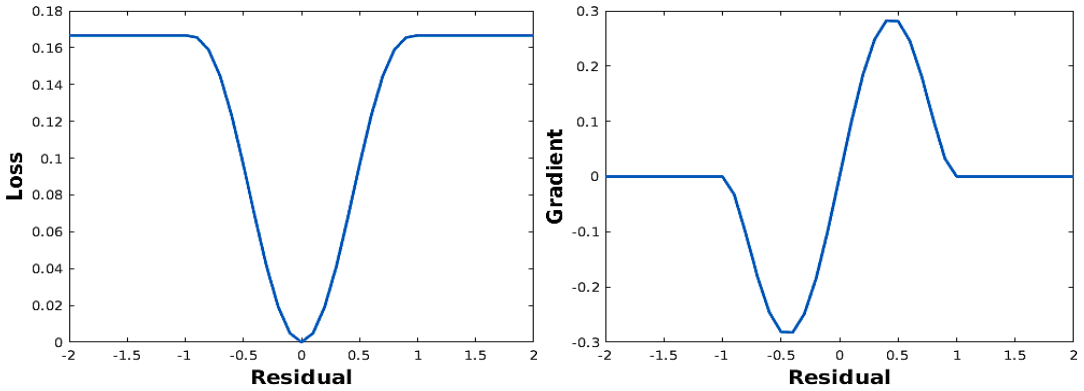


Figure 4.3: Tukey's biweight loss function (left) and its first-order derivative (right) where $c = 1$.

4.4 Evaluation

In this section, we present the implementation details of the proposed models and the dataset used for training. We will also discuss the evaluation metrics that will be used to evaluate the performance of the proposed models.

4.4.1 Implementation Details

The following implementation details were applied to both models. We used MatConvNet [80], a MATLAB toolbox implementing CNNs for computer vision applications, to train and evaluate the proposed models. The weights of the layers were initialized using Xavier initialization method [47]. The model was trained end-to-end from scratch using backpropagation. Stochastic Gradient Descent (SGD) was used to optimize and update the network weights with the following settings: the momentum was set to 0.9 and the weight decay was set to 10^{-5}. The learning rate was initialized to 10^{-3} and was divided by 10 when the validation error didn't change. The training process was repeated until the validation accuracy stopped increasing.

4.4.2 Dataset Preparation

The proposed models were trained on real images in order to predict the depth of objects from a single image. A Large Dataset of Object Scans [114] is a publicly available dataset that contains tens of thousands of 3D scans of different real objects

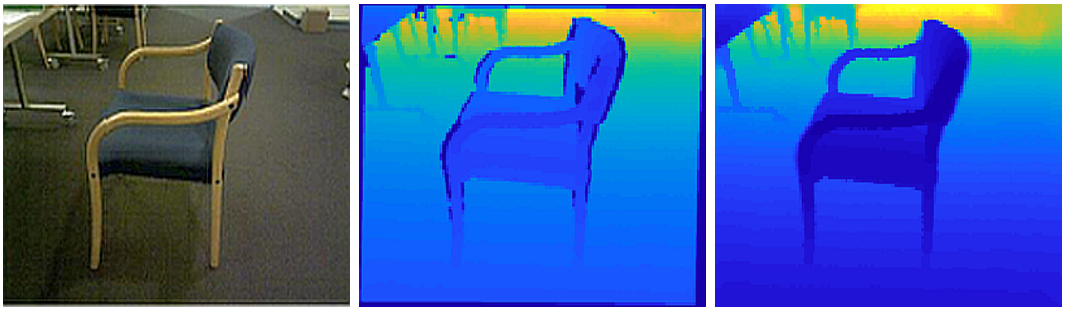


Figure 4.4: A sample of a training RGB image with the corresponding depth image. From left to right: RGB image, original depth image, and preprocessed depth image.

captured at a resolution of 640×480. It includes RGB images and their corresponding depth images of real objects. We collected different shapes of the class chair from it. The collected dataset was split into a training set and a testing set with a respective distribution of 80% and 20%, respectively. We selected the chair object because the dataset has a massive number of images with diversity in shapes text (the proposed methods can easily be applied for other objects). The proposed models were trained on almost 10 different shapes of chairs. Each chair shape was between $1K$ and $2K$ images of different viewpoints and distances in both indoor and outdoor scenes. We applied a pre-processing step on the depth images to fill the missing depth values in the images using the NYU Depth Dataset toolbox [91] as shown in Figure 4.4.

We also augmented the training set to reduce the overfitting during training and for better generalization performance. Horizontal flipping (mirroring) of images is applied at a probability of 0.5. Vertical flipping on indoor scene images will not help during training. Also, we applied photo-metric transformation, *i.e.* swapping the color channels of the RGB images, to increase the performance.

4.4.3 Evaluation Metrics

We evaluated our experimental results using several error and accuracy measures which have been used in prior works:

Mean Absolute Relative Error (rel):

$$\frac{1}{|T|} \sum_{\hat{y} \in |T|} \frac{|y - \hat{y}|}{y} \tag{4.5}$$

Root Mean Square Error (rms):

$$\sqrt{\frac{1}{|T|}\sum_{\hat{y}\in|T|}(y-\hat{y})^2} \tag{4.6}$$

Average $\log_{10}$ Error (log_{10}):

$$\frac{1}{|T|}\sum_{\hat{y}\in|T|}|\log_{10} y - \log_{10}\hat{y}| \tag{4.7}$$

Accuracy with threshold: percentage (%) of $\hat{y}$, s.t.:

$$\max(\frac{y}{\hat{y}}, \frac{\hat{y}}{y}) < \delta_i \quad \text{where } \delta_i = 1.25^i \quad i = 1, 2, 3 \tag{4.8}$$

where y and $\hat{y}$ are the ground-truth of the RGB input image and the estimated depth, respectively. $|T|$ is the number of valid pixels in all images in the validation set.

4.5 Experimental Results and Discussions

In this section, we report thorough analyses and results of the proposed models on single-view depth estimation in indoor and outdoor scenes. Moreover, we performed ablation studies to analyze the impact of the loss functions on the encoder-decoder model. Finally, we compared the proposed models and discuss their results and the performance quantitatively and qualitatively. The quantitative and qualitative results showed that the encoder-decoder model with non-convex loss function performed better than the F-CNN model trained on the same loss function. For the quantitative evaluation and comparison, we will use the metrics stated in Section 4.4.3.

4.5.1 Evaluation Performance of F-CNN

Figure 4.5 shows the depth estimation results that were predicted using the proposed F-CNN model. The results were inferred from the same architecture but trained on two different loss functions (L2 norm and Tukey's biweight loss). As visualized, the

predicted depth images from the model trained on Tukey's biweight loss are better and more accurate than the predicted depth images from the same model trained on the L2 norm. The objects in the scene have better details and they are not fused with the background. However, the inferred depth images from the model trained on the L2 norm are not accurate. It is clearly seen that the object parts are missing and in some results (the last image in the last row), the chair details are completely missed.

However, the proposed F-CNN has some weaknesses. The generated depth image is 1/4-resolution of the original input image. As a consequence, many fine details are missed and the depth information isn't inferred accurately. Also, the generated images are blurry and the objects' details are fused with the background. Moreover, some border areas are lost. A trivial solution for this problem is to apply a post-processing step to the inferred depth image. Applying bilinear interpolation to enlarge the generated depth image can improve the quality of the output. However, this will

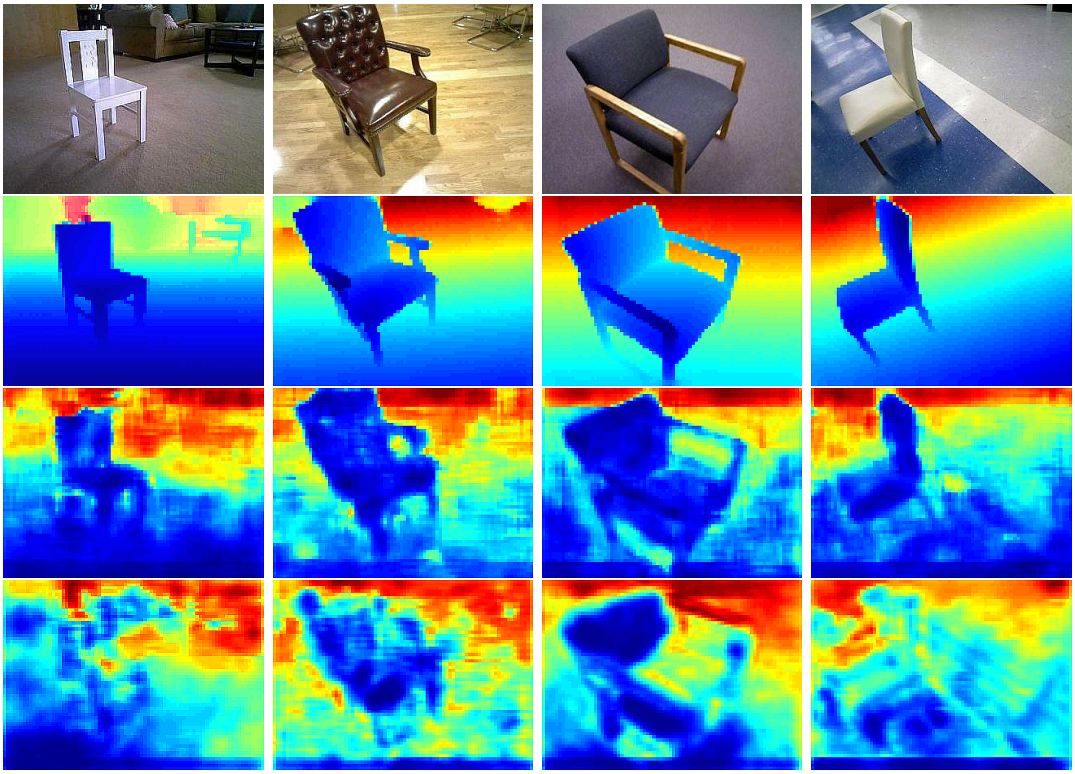


Figure 4.5: Qualitative results from F-CNN on Large Dataset of Object Scans (chosen from the testing set). From top to bottom: input RGB image, ground-truth, results from the model trained on Tukey's biweight loss, and results from the model trained on L2 norm. Depths are shown in log scale and in color (blue is close, red is far).

generate a smooth depth image and still, the details will not be recovered well. Another solution is to modify the CNN model to generate an output that has the same resolution as the input. We will investigate this solution in the next section.

4.5.2 Evaluation Performance of mini V-Net

The proposed F-CNN model shows some limitations such as the output resolution and the blurry inference of the information. The proposed encoder-decoder model overcomes these limitations. Figure 4.6 demonstrates the inferred depth results from the proposed encoder-decoder model. The predicted depth images have preserved the object details and can be easily distinguished from the background. The fine details of the objects such as the holes in the back of the chairs are predicted accurately and the model has succeeded to estimate the objects' parts. Interestingly, the proposed model predicts directly depth information from a single input image without any further post-processing steps. Moreover, the generated depth image has the same resolution as the input image. The model is trained end-to-end to estimate the depth of an image in a single shot. On the other hand, other previously described methods improved the predicted depth images through many steps. One method used a multi-stage model and combines the coarse depth image generated in one stage and the original RGB input image to generate the final depth image. This may introduce noise and reduce global scale depth information. Also, the output resolution is usually smaller than the input resolution and many details may be missed. Other methods used CRF as a post-processing step to generate a more detailed depth image. As a result, the predicted depth image cannot be estimated directly from CNN. However, our model differs from these models in being a single-stage model whereby no post-processing steps are required to generate the output.

Moreover, we reconstruct the images in 3D using the predicted depth values as shown in Figure 4.6 (the last row). It clearly is shown that the depth values were predicted very well and there are different levels of depth values that can distinguish between object parts, the floor, and the background.

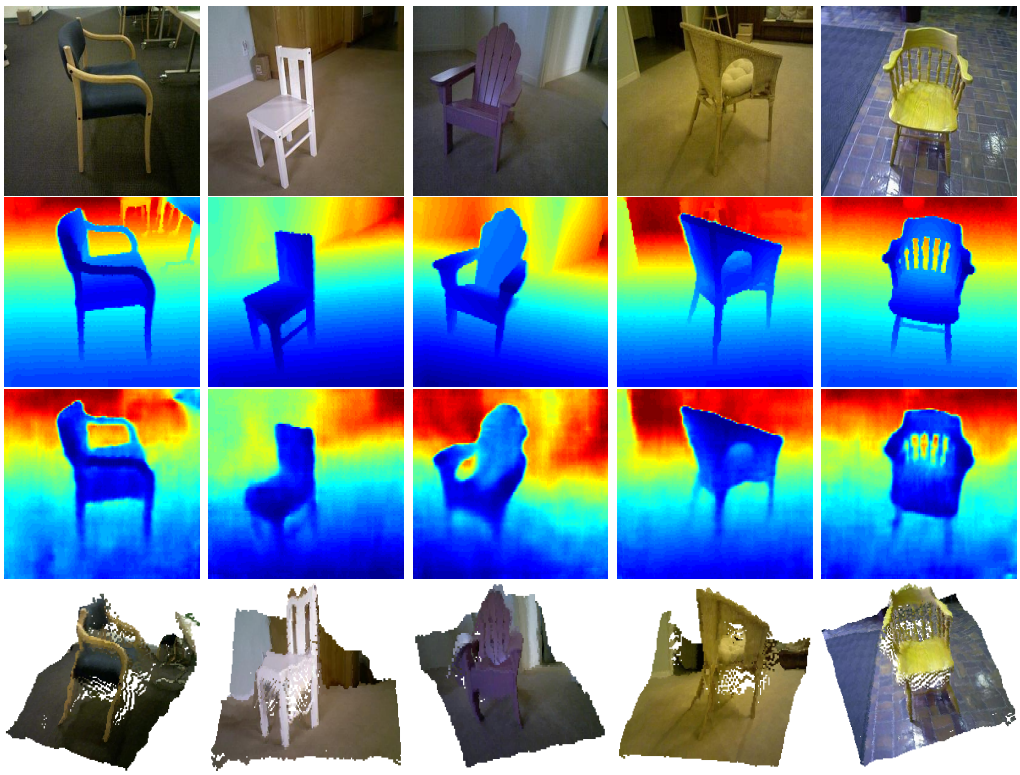


Figure 4.6: Qualitative results from mini V-Net on Large Dataset of Object Scans (chosen from the testing set). From top to bottom: input RGB image, ground-truth, predicted depth image, and the reconstructed images using the predicted depth values. Depths are shown in log scale and in color (blue is close, red is far).

4.5.3 Analysis of Different Loss Functions

To test the importance and the efficiency of choosing a suitable loss function for better training, we evaluated the performance of the mini V-Net model on two different loss functions quantitatively and qualitatively. We trained the model using two different loss functions; L2 norm and Tukey's biweight loss. The training environment was fixed and the same training images were used to train both models. In depth estimation task, the small difference between the depth values is important to update the network weights because these values highlight the basic features of the object and differentiate it from other objects in the scene. We compared the error and the accuracy of the estimated depth from Tukey's biweight loss with the one from L2 norm loss quantitatively. Figure 4.7 (left) shows that the error computed from Tukey's biweight loss model is smaller than the model trained on the L2 norm with a large margin. Also, the accuracy

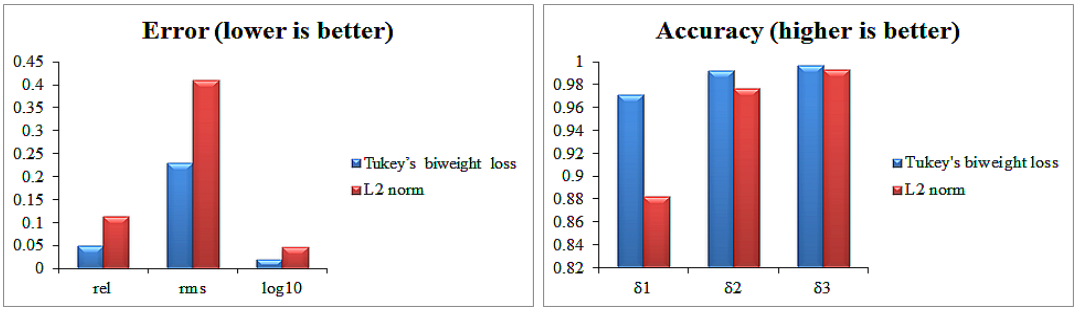


Figure 4.7: Error and Accuracy results of the mini V-Net model using different loss functions.

(right) when using the non-convex loss function is better for training the model in the regression problems.

Figure 4.8 elucidates that the model trained using Tukey's biweight loss outperforms the model trained using the L2 norm. In detail, the pixels with smaller distances are sensitive to smaller errors. This influences the relative error to be higher and results in larger gradients of Tukey's biweight loss over L2 norm. Consequently, the non-convex loss function is more robust to the outliers and takes care of the small errors between distances such that the output is estimated with finer details compared to the L2 norm model. Figure 4.8 also shows that the results predicted by the model trained using L2 norm are relatively blurry depth images and the object details are almost missed (the fourth row in Figure 4.8). Some object parts are fused with the background and the object details are not visible. On the other hand, the depth images generated by Tukey's biweight loss have captured finer details and the object inside the images can be recognized easily from the background (the third row in Figure 4.8). In addition, the network learns to preserve some details related to object shapes such as the holes in the chair's arms and the empty space between the back of the chair and seat.

4.5.4 Comparison of the proposed models

So far, we have proposed two different models to estimate the depth information from a single image. To show the influence of the network design, we compared both models quantitatively and qualitatively. The first model (F-CNN) is a fully convolutional model that uses the pooling layer to decrease the feature size and the computation

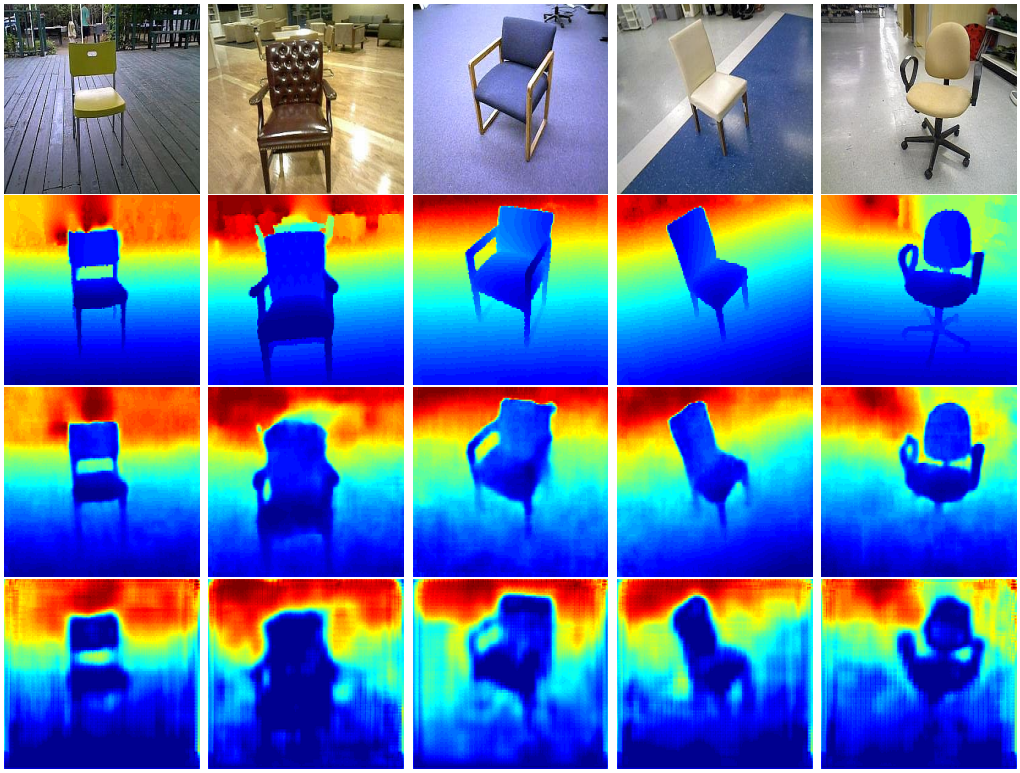


Figure 4.8: Qualitative comparison results on Large Dataset of Object Scans using different loss functions on mini V-Net model. From top to bottom: input RGB image, ground-truth, model trained using Tukey's biweight loss, model trained using L2 norm. Depths are shown in log scale and in color (blue is close, red is far).

cost. The second model (mini V-Net) is an encoder-decoder model that uses strided convolutional layers instead. F-CNN was trained with two different configurations. First, we trained the F-CNN model, and the output resolution was fixed (the generated the depth image that had a size of 1/4-resolution of the input image size). In the second configuration, the generated output from the F-CNN model was upsampled to meet the resolution of the input image. All models were trained end-to-end using the same loss function. The output of F-CNN has a smaller resolution than the input image, but the second model generates a depth image with the same input image resolution. Table 4.1 shows the quantitative comparison with respect to errors and accuracy. Figure 4.9 shows the qualitative results generated by both models.

As reported in Table 4.1, the F-CNN model trained to generate images without any post-processing step outperforms the same model when the output is upsampled. This is because the upsampling step generates blurry images and many details are

Table 4.1: Performance Comparison of different methods trained using different loss functions on Large Dataset of Object Scans (↓ lower is better, ↑ higher is better).

Architecture	rel↓	rms↓	log10↓	$\delta 1 \uparrow$	$\delta 2 \uparrow$	$\delta 3 \uparrow$
Models trained using Tukey's biweight loss						
F-CNN (upsampled)	0.2940	0.9516	0.1264	0.4895	0.7958	0.9205
F-CNN	0.2341	0.7644	0.0970	0.5971	0.8940	0.9720
mini V-Net	**0.0507**	**0.2314**	**0.0218**	**0.9713**	**0.9927**	**0.9972**
Models trained using L2 norm						
F-CNN (upsampled)	0.3047	1.2146	0.1661	0.3771	0.6662	0.8344
F-CNN	0.2571	0.9976	0.1317	0.4453	0.7794	0.9202
mini V-Net	**0.1150**	**0.4104**	**0.0479**	**0.8825**	**0.9772**	**0.9935**

lost as they are an extension to the neighboring regions. However, the mini V-Net model outperforms both configurations of the F-CNN model when using different loss functions with a large margin. The measured error is very small and the accuracy is very high comparing with F-CNN. Figure 4.9 validates this result as the generated depth images from the mini V-Net model are more accurate than the ones generated from F-CNN. This is because the F-CNN model uses two pooling layers which decreases the image resolution twice and some features are lost from these layers. However, mini V-Net is an encoder-decoder model, where deconvolutional layers (upconvolutional layers) are used to reconstruct the image again to its original resolution. Using these layers allows us to solve the issues related to the output resolution. Furthermore, we used skip connections between the encoder and the decoder to transfer useful information that has been extracted from the encoder part and utilized them when predicting the depth information in the decoder part. These connections improve the quality of the generated images and make the objects' parts sharper. The object parts and the holes in the chairs can be easily recognized from the depth images generated by our encoder-decoder model. To decrease the feature dimensions, strided convolutional layers are used that have features to be learned and can extract useful features, unlike the pooling layers.

Figure 4.9 demonstrates that the encoder-decoder models perform better than normal F-CNN models in solving regression problems (where the problem is an image-to-image problem). Moreover, the encoder-decoder model predicts the depth at a higher

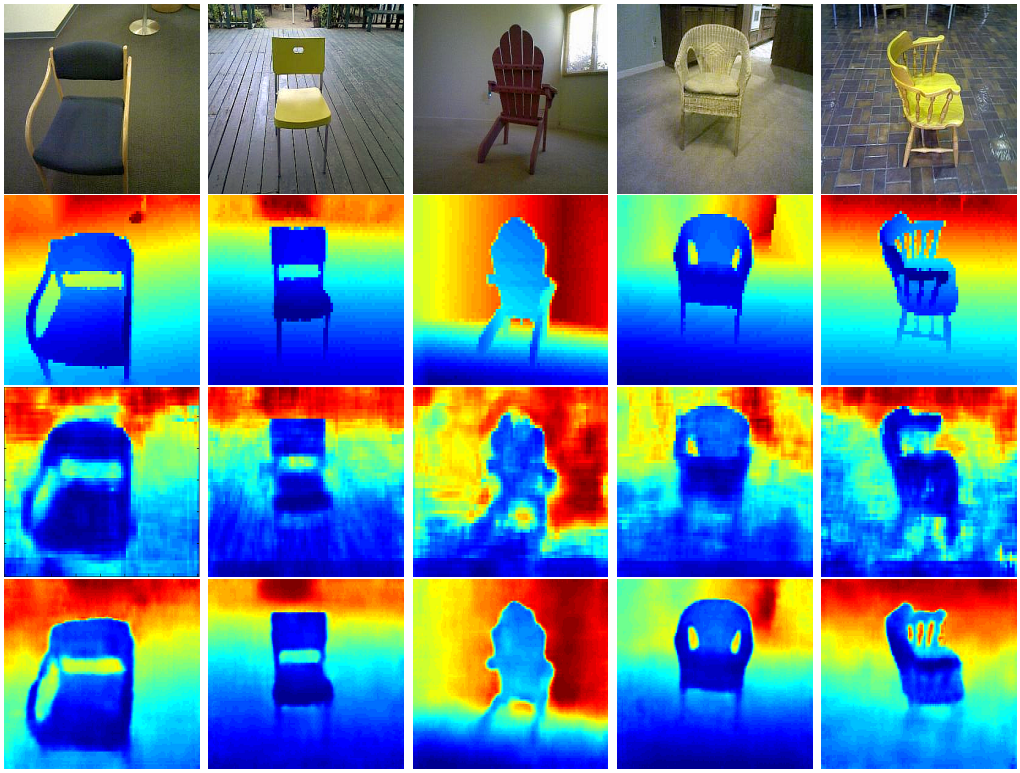


Figure 4.9: Qualitative comparison results on Large Dataset of Object Scans. From top to bottom: input image, ground-truth, F-CNN trained on Tukey's biweight loss, and mini V-Net trained on Tukey's biweight loss. Depths are shown in log scale and in color (blue is close, red is far).

quality where the edges and the holes almost match the ground-truth images with fewer artifacts.

4.6 Conclusion

Depth estimation from a single image is an extremely challenging problem. In this chapter, we have presented two different CNN models to estimate the depth information from a single image. The first model was a fully convolutional network (F-CNN) that generates plausible depth images from a single image. However, the generated images are blurry and resolution is smaller than the input image. To overcome the first model's limitations, we proposed a fully-convolutional encoder-decoder model to solve the same task. Unlike the traditional models that require multi-stage or post-processing steps to predict the depth, our model is a simple single-stage model that predicts the depth images directly without any further post-processing steps. By contrast to

other methods, that struggle to generate high-resolution images, the generated depth images using our model have the same resolution as that of the input images. We experimentally demonstrated that the loss function influences the final output, and for our specific problem the non-convex loss functions are more suitable for regression tasks because they are robust to the outliers. We showed that our simple and well-designed model outperforms other models on the same datasets and using the same loss functions during training. Our work generates high-quality depth images that capture the boundaries and reveal finer object parts such as the holes in the back.

5

3D Object Reconstruction from a Single Image

Simplicity is the ultimate sophistication.

Leonardo da Vinci

5.1 Introduction

The goal of this chapter is to infer the 3D geometry and the structure of an object from a single image only. This is a long standing ill-posed problem and fundamental to many applications such as object recognition, scene understanding, and 3D reconstruction. Single-view 3D reconstruction means using a single input image of an object and the reconstructed output can be viewed from all directions.

For multiple input images, many methods have been proposed which are able to present high-quality reconstruction results. The challenge appears when a single input image is just available to be used for the reconstruction process. Some methods were proposed with special assumptions on both the input image and the object class to be reconstructed. The term **class**, here, means the object nature with its properties like shape, color, topology, *etc.* Single view 3D reconstruction is a hard problem and it

mainly depends on the available information and the restricted assumptions about the object. This information or cues provide prior knowledge that helps in generating 3D shapes with plausible precision [16].

Multi-view reconstruction for objects has been investigated extensively [115]. With multiple cameras, methods such as triangulation can estimate the depth information and help to generate the object's surface. However, applying the triangulation method to a single image will generate an infinite number of models that the projection of the generated shapes could be similar to the input object image. To solve this ill-posed problem, object cues such as shading and silhouette can be used as a condition to estimate the 3D shape. Many methods were proposed to utilize object cues to solve this problem. The drawback of these methods is the assumptions they impose regarding the nature of the objects and the need sometimes for user inputs.

As humans, we can easily extract the 3D geometry of images using our visual system and leveraging the prior knowledge we have about the scenes. Moreover, humans can generate and infer the 3D models of any object from either a single image or using one eye only. For unseen objects, we can guess and generate plausible 3D shapes with approximated size by utilizing the prior knowledge and can infer the geometry of unseen objects from different viewpoints. This can easily be done by humans because the brain has the ability to understand surrounding scenes and build a strong prior information about it. Then, the brain utilizes this information to recognize the new 3D geometry of this scene depending on learned knowledge from real world. Thanks to our complex visual system, this ill-posed problem can be solved for any class of the object without imposing any prior assumptions regarding the scene.

With the astonishing results obtained by applying deep learning on different computer vision problems, many researchers have formulated single-view reconstruction problem to be solved using deep learning and exploiting the available large-scale datasets [68] for training. The inferred shape is represented in different ways. The easiest and the most common representation is volumetric as the CNN can be generalized from 2D to 3D [116]. The output resolution is small as in the volumetric representation, the computational and memory costs increase cubically when increasing the object

resolution. Other methods proposed to use meshes, multi-view, and point clouds to represent the object.

3D reconstruction from single or multiple images has many useful applications. In the entertainment industry, it has been applied to aid in the process of movie-making. For example, a scene with a large number of horses galloping in a field can be achieved in a simple way. First, reconstructing the dynamics of a horse galloping in the field is made. Then, the new views of the horse can be synthesized from the obtained 3D horse model. Finally, the scene can be created by inserting as many horse models into the scene as desired. This simple example shows that without the aid of 3D reconstruction, computer graphics artists would spend many tedious hours of modeling when the objects are rendered. Also, many times, they may face many problems to make the realism of the object or the scene. Other techniques have been explored and used in recreating 3D models such as attacking markers. This technique is often inconvenient and uncomfortable to be used. The main reason is that markers disrupt the performance of the user and photo-realistic 3D models based on markers are difficult. To make the process attractive, the ability to create 3D reconstructions using only images is used. 3D reconstruction from multiple images has been used and found in many applications such as creating movies, computer games, and animations. However, the challenge appears when there is a lack of views, or only a single image is available for reconstruction.

Also, 3D reconstruction is widely applied in the medical field [117]. It has been used to create models of organs as well as brains and teeth. Other applications include body motion modeling, object recognition, surveying such as the modeling of buildings and terrain, robot navigation, surveillance, and teleconferencing. Depending on the application, the quality of the image and the user input are decided. For example, robots and building modeling tend to capture images using stereo cameras, while human body modeling allows for a multiple-camera setup.

The projection from 2D to 3D is difficult because of the mysterious correspondences between the pixels in 2D and the points in 3D space. To this end, we propose a CNN model that solves the task of single-view reconstruction. The model has an encoder-

generator shape where the encoder extracts useful features from the input image and the generator infers the point clouds of the object shown in the 2D image. To generate more accurate point clouds, an initial point cloud is used to improve the reconstruction quality. We find that starting from an initial point cloud, the points distribute equally on the shape surface and preserve the object parts. We summarize our contributions as follows: (1) we design a CNN model that can infer the 3D geometry of an object from a single image. The 3D object is represented as a point cloud. (2) Instead of directly inferring the point cloud, we propose to utilize an initial point cloud of a sphere shape to generate the final object point cloud. The experimental results that using an initial point cloud helps in generating better and more accurate reconstruction. (3) We evaluate the proposed model on synthetic and real data qualitatively and quantitatively. Our model outperforms the state-of-the-art methods and shows significant results for the task of single-view reconstruction. The results represented in this chapter were published in [118].

5.2 Background and Literature Review

In this section, we will discuss two categories of methods that were proposed to solve the problem of single-view reconstruction. The first category imposes priors and cues on the target object that is needed to be reconstructed. The second category depends on deep learning methods. But first, we will revisit different 3D data representations and how they can be used in deep learning.

5.2.1 3D Representation

The recent development in 3D sensing technology and low-cost devices have facilitated the process of 3D data collecting. Based on the used scanning device for data collection, the generated raw data are saved in different forms. They vary in structure and geometric properties. When applying deep learning models to generate 3D shapes or to solve other tasks such as segmentation, recognition, or object classification, the object representation plays an important role in designing the network. This leads

(a) An example of a volumetric representation.

(b) An example of a multi-view representation.

Figure 5.1: Euclidean representation.

to classifying the 3D representation of the objects into Euclidean and Non-Euclidean representations.

Euclidean-structured data have the properties of global parametrization and a common system of coordinates. These properties allow us to easily extend the 2D CNNs for 3D data processing. Following, we will discuss volumetric data and multi-view data representations as they are mostly used in deep learning to represent the 3D data.

Volumetric data can be represented as a regular grid in the 3D space [119]. Voxels are used to visualize 3D data and show the distribution of the 3D object in the 3D space. Each voxel in the 3D space that describes the object can be classified into a visible, occluded, or self-occluded voxel according to the viewpoint. Voxel-based representation is very simple and has the ability to encode the 3D object information and its viewpoint. However, it is sometimes an inefficient representation because each voxel in the 3D grid should be represented (occupied and non-occupied voxels) and this leads to preserve a huge useless space in the memory and increase the computational time when processing the volumetric data. Also, fine-grained shape parts get lost because the voxel is represented as either occupied or unoccupied. Figure 5.1a shows a chair represented in voxels of size $128 \times 128 \times 128$.

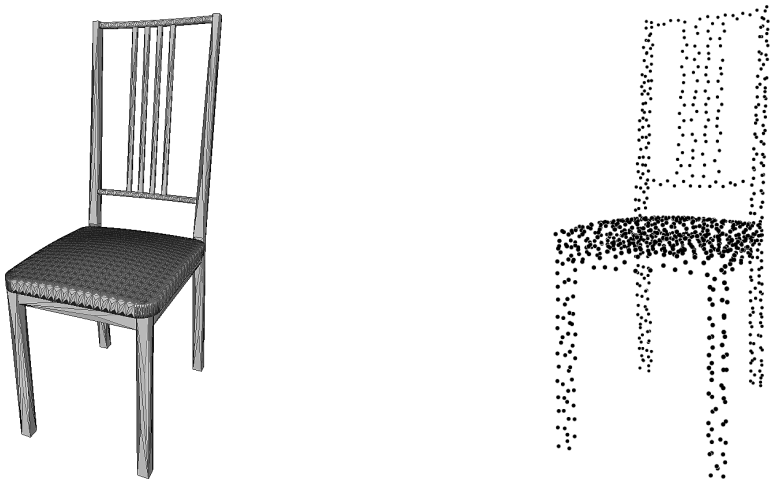
Octree-based representation is an efficient modified version of voxel-based representation [120]. In octree-based representation, the voxel size is not constant nor equal. Each occupied voxel is divided into smaller cubes that are either inside or outside the 3D object. This helps in preserving the fine details of the object compared to voxel-based representation. However, both representations don't preserve the shape smoothness of the surface.

Multi-view data can be understood as capturing multiple 2D images of the same object from different viewpoints [121]. It helps in extracting multiple features for noise reduction and solving the problem of partial occlusion and incomplete shape reconstruction. Representing 3D data as multi-view images aims to learn 3D information of the object from different views separately. Then, the learned information is combined to represent the whole 3D shape and to generalize to other 3D shapes of the same object class. However, there is no optimal number of views that can be used to generate an accurate 3D object. On one hand, a small number of images might not capture the general properties of the 3D shape. On the other hand, a massive number of views can cause unnecessary computational overhead and memory allocation. Figure 5.1b shows an example of a multi-view representation.

Both representations (volumetric and multi-view) are used nowadays in deep learning to solve different tasks, especially with rigid data.

The other type of 3D data representation is for the non-Euclidean data. Non-Euclidean data don't have a global parametrization or a common system of coordinates. This makes it difficult in extending the 3D CNNs to process these data. Point clouds and 3D meshes are mostly used as 3D data representation in deep learning.

3D Meshes are commonly used to represent the 3D shapes. The structure of a 3D mesh comprises a set of polygons which are called faces [122]. These polygons are described using a set of vertices that describe how the mesh coordinates exist in the 3D space. Beside the 3D coordinates of the vertices, there is a connectivity list that specifies how the vertices are connected to each other. Figure 5.2a shows a mesh of a chair.



(a) An example of a mesh.

(b) An example of a point cloud.

Figure 5.2: Non-Euclidean representation.

3D Point Cloud is a set of unordered 3D points that approximate the geometry of 3D objects [123]. This makes it a non-Euclidean geometric data representation. However, a point cloud is a simple uniform structure that can be encoded and learned very easily. It allows simple geometric transformation and deformation without the need for connectivity updates. Figure 5.2b shows a point cloud of a chair represented by 1024 points.

Applying deep learning techniques on 3D data represented as meshes is a challenging task. Extending the 2D CNNs to deal with non-Euclidean data is not a straightforward task because of the irregular representations. Also, these data usually suffer from resolution problems, noise, and missing data. In our work, we applied deep learning directly to generate the point cloud of an object from a single image without the pre-processing or post-processing stages.

5.2.2 Single-view Reconstruction before Deep Learning

There are many approaches that proposed different methods in the field of single view reconstruction. Some of them were applied to real-world images without any assumptions on how the capturing process was performed. Also, the output of these approaches is plausible. The proposed approaches that depend on the cues and priors

can be categorized according to the target objects or scenes. There is a difference between object and scene reconstruction. In scene reconstruction, the proposed method tries to infer depth information and produce 3D features from the whole input image. In object reconstruction, however, the proposed methods focus on a single object in the scene and try to produce smooth objects. The reconstructed target objects can be categorized as the following:

- Curved Objects
- Piecewise Planar Objects
- Learning Specific Objects
- 3D Impression from Scenes

In curved objects approaches, an energy function is defined to minimize the object surface with respect to some constraints such as a fixed area or volume. An early work, done by Zhang *et al.* [124], proposed a method for interactive depth map editing based on the input image. The proposed algorithm takes a set of user-specified constraints as input such as surface positions, surface normals, and silhouettes to reconstruct a plausible 3D surface according to the inputs. The proposed method recalculates and displays the reconstructed result each time the input changes in a real-time manner. Prasad *et al.* [19, 125] proposed a framework for single-view reconstruction. The objects to be reconstructed are of a curved surface. Silhouette is used as a cue to reconstruct the object. An energy function was minimized to generate a smooth surface subject to the silhouette as a constraint. Oswald *et al.* [20] proposed a method that computes the closed minimal curved surface of a symmetric object. The input of the method is a side view of an object and its silhouette that is obtained by segmenting the object from the background. The object is reconstructed by finding a smooth surface that the 2D projection of the output is compatible with the silhouette input. The similarity measurement is calculated using a distance function between the 2D projected image and the silhouette. Toppe *et al.* [22] also proposed a framework similar to [20] that requires the object silhouette as input. The difference between them is how they

represented surface inflation. The first one proposed a heuristic data term for surface inflation, but the second method suggests that the surface volume has to comply with a user-specified volume input.

The second class focuses on piecewise planar objects such as buildings and man-made environments. Criminisi *et al.* [126] used a single perspective view to calculate the 3D affine measurements of a scene containing planes and parallel lines. The authors assumed that a vanishing line and a vanishing point can be calculated from a reference plane. From the vanishing line and point, an affine scene structure can be obtained. Different measurements are calculated from the input images: the distance between parallel planes, length and area ratios on these planes, and the camera positions. Based on the computed vanishing line and point in the image, the 3D reconstruction can be obtained and the camera position can be estimated. Delage *et al.* [127] proposed a method for reconstructing indoor scenes. The indoor scenes consist of parallel and orthogonal planes. They assumed that the camera is calibrated, the scene consists only of planes and edges, and the camera's axis is vertical to the floor of the scene and the height is known. Hong *et al.* [23] studied the relation between the symmetry of objects and the viewer's relative pose of the object. The advantage of the assumption symmetric of objects is that one image of a symmetric object is equivalent to multiple images.

Methods in the third class cannot reconstruct arbitrary objects but are inherently limited to specific object classes by shape information learned from sample databases. Some researchers proposed methods for the reconstruction of objects with some prior knowledge of the object properties. These methods cannot reconstruct arbitrary objects but can reconstruct specific objects with some shape information learned from sample databases. Han and Zhu [24] proposed a reconstruction method with manually defined shape priors. They assumed that identifying prior information of an object automatically is hard to learn because of the lack of real training data. To reconstruct a full 3D scene, the input image is transformed into two graphs; one for the 3D objects and the other one representing the relations between the objects in the scene. An example, in the grass tree-like scene, authors assumed smooth and evenly separated curves for the

objects and the relation graph describes the regularity constraints on touching objects. Nagai *et al.* [128] used a sample database to learn objects. Then, they used Hidden Markov Model to model the correspondence between intensity and depth.

The fourth class methods didn't aim to generate exact or plausible 3D geometry but rather reconstruct a pleasing 3D Impression from Scenes. Hoiem *et al.* [21] proposed a fully automatic method for creating 3D models from single images. They divided the world into the sky, ground, and vertical objects. To reconstruct the 3D model, the bottom of the vertical region is fit with the ground, the sky is removed from the model, and the vertical pixels are assumed to belong to the objects above the ground. In [88], Saxena *et al.* proposed another approach similar to [21] to obtain a 3D structure from a single image. They assumed that the world consists of small planes, whose 3D position and orientation are to be estimated. They divided the input image into superpixels and infer the depth values and the orientation for each of them. This method is modeled using Markov Random Field (MRF) and solved using a linear program. The inferred 3D structure is a polygonal mesh representation of the 3D model.

In general, most of the above literature has advantages and disadvantages. The advantages are that most of them have high reconstruction precision. The results are plausible. They used some inputs to improve the final results. The disadvantages are some of the proposed methods take a long time for reconstruction. They are limited to some objects domain that refer to a set of objects that can be reconstructed successfully.

5.2.3 Single-view Reconstruction using Deep Learning

Theoretically, inferring the 3D structure of an object from a single image is an ill-posed problem, but many attempts have been done such as SFM and SLAM [129, 130]. Moreover, ShapeFromX, where X can be shadow, texture, *etc.*, needs prior knowledge on the nature of the input image [131].

With the emergence of deep learning, enormous models were proposed to solve different computer vision tasks such as object classification, segmentation, *etc.* Nowadays, with the availability of large-scale 3D shape datasets, *e.g.* ShapeNet [68], deep learning 3D-based models have made great progress in solving different tasks

using 3D data directly such as classification, object parts segmentation, and 3D shape completion.

The reconstructed output generated by the deep learning models were represented using different forms. To extend the 2D convolutions to 3D, the volumetric representation has been mostly used. It is simple in implementation and compatible with the 3D-CNN. 3D-GAN [132] proposed a generative adversarial network (GAN) to generate 3D objects from a probabilistic space using volumetric CNN. They mapped a low-dimensional probabilistic space to the 3D object space and by this, they outperform other unsupervised learning methods. Moreover, a 3D recurrent neural network (RNN) has been proposed to estimate the 3D shape of an object. 3D-R2N2 [116] proposed to use long short-term memory (LSTM) to infer the 3D geometry using many images of the target object from different perspectives. A more accurate representation is octree. Octree Generation Networks (OGNs) [120] proposed a decoder that infers the 3D shape in a computed and memory-efficient way by representing the output using octree. The proposed architecture doesn't have the cubic complexity and the output can be represented in higher resolution with more object details. The critical limitation of using the volumetric representation in the above-mentioned methods is the computational and the memory cost and the restriction of the resolution.

To avoid the limitation of the volumetric representation, mesh representation is more attractive for real applications as the shape details can be modeled accurately. Applying deep learning models directly to generate meshes is a challenge as they are not regularly structured. A parameterization-based 3D reconstruction is proposed in [133] that generates geometry images which encode $x; y; z$ surface coordinates. Three separated encoder-decoder networks were used to generate the geometry images. The networks take an RGB image or a depth image as an input and learn the $x; y$; and z geometry images. Other methods proposed to estimate a deformation field from an input image and apply it to a template 3D shape to generate the reconstructed 3D model. In [134], an end-to-end deep learning model was proposed to generate a triangulated 3D mesh from a single image. The proposed network represented the 3D mesh in graph-based CNN (GCNN) and deform an ellipsoid leveraging the perceptual

features extracted from the input image. They adopted a coarse-to-fine strategy that makes the deformation process stable. Kuryenkov *et al.* [135] proposed DeformNet that takes an image and a nearest 3D shape to that image from a dataset as input. Then, the template shape is deformed to match the input image using Free Form Deformation layer (FFD). A limitation of using meshes for reconstruction is that the generated output is limited mostly to the initial mesh or the selected template as an initial shape to be deformed.

Point cloud representation is a flexible form to represent the 3D structure and it becomes a commonly used representation in deep learning as it is simple and highly efficient in terms of memory requirements (compared to volumetric representation). A point cloud is a not-regular structure and applying deep learning models on this representation is a challenging task. A 3D shape can be represented using an unordered set S of N elements where,

$$S = \{(x_i, y_i, z_i)\}_{i=1}^{N} \tag{5.1}$$

To overcome this limitation, points can be represented either as a matrix of size $N \times 3$, a 3-channel grid of size $H \times W \times 3$ where each pixel encodes the (x, y, z) coordinates and $H \times W$ equals to the number of points, or depth maps from different known viewpoints. Point Set Generation Network (PSGN) [123] was the first proposed model to generate a point cloud of an object from a single image and outperforming the volumetric approaches. RealPoint3D [136] has two encoders; one for the input image the target object and the second for the nearest shape retrieved from ShapeNet dataset. The encoded features from both encoders are integrated and forwarded to a decoder to generate fine-grained point clouds. The point cloud from the retrieved shape influences the inferring process and generates finer point clouds. 3D-LMNET [137] trains a 3D point cloud auto-encoder and then learned the mapping from the 2D images to the learned embedded features. Another direction to generate the point cloud is to generate depth images of different perspectives and fuse them to generate the final point cloud. In [138], a generative modeling framework used 2D convolutional operation to predict multiple pre-defined depth images and use them to generate a dense 3D model.

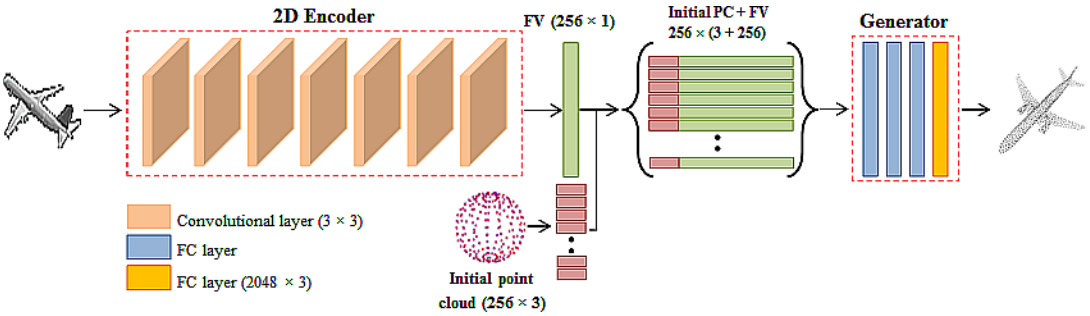


Figure 5.3: The proposed CNN Architecture. The encoder network extracts features from the input image. The extracted feature vector is concatenated with the coordinates of the sphere points. The generator takes the extracted feature vector and the coordinates of the sphere as input and generates the final point cloud of the input object.

5.3 Methodology

Our main goal is to infer a complete 3D shape of an object from a single RGB image. We selected point cloud to represent the 3D shapes (Eq. 5.1). We set the number of the points generated from the CNN to $N = 2048$. From our experiments, we found that this number of points is sufficient to cover the whole surface of the object and preserve the major structures.

5.3.1 3D CNN Model

The proposed network is illustrated in Figure 5.3. It consists of two parts; the encoder part and the generator part. The encoder part is a set of consecutive 2D convolutional layers followed by ReLU as a non-linear activation function. These layers are used to extract the object features from the 2D input images. To predict the 3D point cloud of the object, an initial point cloud with a sphere shape is used. The initial point cloud is concatenated with the extracted features from the encoder as shown in Figure 5.3. Then, it is fed into the generator part to get the final point cloud of the object, where FC layers are used to generate an $N \times 3$ matrix, where each row contains the coordinates of one point. Following, we will explain the network in detail.

Encoder Net The role of the encoder part is to extract the distinction features from the input image that can correctly describe the object with details. It consists of consecutive layers of 2D convolutional layers and ReLU layers. The convolutional layers

are seven layers feature. The first three convolutional layers are with sizes of $32, 64$, and 128, respectively. The remaining layers are with a size of 256. All convolutional layers have a kernel size of 3×3 and a stride of 2. The stride of 2 in the convolutional layers helps in decreasing the spatial size of the features as the pooling layers do. However, the strided convolutional layers are trainable and extract useful features. The size of the input image is 128×128. The extracted feature from the encoder has a size of $1 \times 1 \times 256$, which is then reshaped and concatenated with the initial point cloud.

Generator Net The generator part is a simple network consisting of four FC layers. After extracting the features from the encoder, the feature is reshaped as 1×256 and then concatenated with the initial point cloud. The initial point cloud has a sphere shape and the size of 256×3, where each row represents the coordinates of a point in 3D. The reshaped feature is concatenated with each point of the initial point cloud, and the new feature has a size of $256 \times (3 + 256)$. Figure 5.3 shows the reshape concatenation process. The initial point cloud with the concatenated feature is fed into the generator. After three FC layers followed by ReLU, the generator ends with a FC layer that predicts the final point cloud with a shape of 2048×3.

The proposed network is different from other single-view reconstruction models as the proposed model utilizes an initial point cloud with a sphere shape for better inference of the final point cloud. In the results section, we will discuss and show the importance of using an initial point cloud and how this improves the final results and distributes the points evenly.

5.3.2 Loss Function

Selecting a suitable loss function to train the CNN model is a critical step. The nature of the problem, the dataset representation, and the output values are the points that should be considered when designing the loss function. The loss function measures the error between the inferred output and the corresponding ground-truth and according to the error, the model weights are optimized and updated. In our case, the loss function will measure the distance between the generated point cloud and the ground-truth shape. It should satisfy the following conditions:

1. the selected loss function should be efficient to compute and differentiable, with respect to the point locations, so that it can be used for the backpropagation step, and

2. it should be robust against the outliers.

So, the required loss function L between two 3D shapes, S_{pred}, $S_{gt} \subseteq \mathbb{R}^3$, is defined as:

$$L(\{S_{pred}\}, \{S_{gt}\}) = \sum d(S_{pred}, S_{gt}) \tag{5.2}$$

where S_{pred} and S_{gt} are the predicted 3D shape and the correspondence ground-truth shape, respectively.

Since the point cloud is an orderless representation, the loss function should be invariant to the ordering of the points. To this end, we propose to use and train the proposed model using two different loss functions: Chamfer Distance (CD) [139] and Earth Mover's Distance (EMD) [140]. Both distances are only piecewise differentiable.

Chamfer Distance: The Chamfer Distance between S_{pred}, $S_{gt} \subseteq \mathbb{R}^3$ is defined as:

$$d_{CD}(S_{pred}, S_{gt}) = \sum_{x \in S_{pred}} \min_{y \in S_{gt}} \|x - y\|_2^2 + \sum_{y \in S_{gt}} \min_{x \in S_{pred}} \|x - y\|_2^2 \tag{5.3}$$

In the first term of Eq. 5.3, for each point in the predicted point cloud, CD finds first the nearest neighbor in the ground-truth point cloud and sums the squared distance up. The second term of Eq. 5.3 does the same but from the ground-truth point cloud to the predicted point cloud. CD is piecewise smooth and continuous, and the search process is independent for each point. So, this function is parallelizable and produces high-quality results. The lower the value, the better and more accurate the generated shape. The drawback of CD is that there is no clear mechanism to enforce the uniformity of the generated point cloud because the optimization process leads to a minimum where a subset of points accounts for the whole shape and clusters the remaining points.

Earth Mover's Distance: The EMD between S_{pred}, $S_{gt} \subseteq \mathbb{R}^3$ is defined as:

$$d_{EMD}(S_{pred}, S_{gt}) = \min_{\phi: S_{gt} \to S_{pred}} \sum_{x \in S_{gt}} \| x - \phi(x) \|_2 \tag{5.4}$$

where $\phi : S_{gt} \to S_{pred}$ is a bijection and the size of S_{pred} and S_{gt} is equal, $s = |S_{pred}| = |S_{gt}|$.

In EMD, ϕ maps each point from S_{pred} to a one unique point in S_{gt}. It enforces a point-to-point assignment between the two point clouds. EMD is differentiable almost everywhere and parallelizable but computationally expensive (with respect to the time and the memory for high-resolution point clouds).

5.4 Evaluation

In this section, we will outline the implementation details of the proposed architecture and the datasets used for training. We will also discuss the testing datasets that will be used to evaluate and compare the proposed method against the state-of-the-art.

5.4.1 Implementation Details

We implemented and train the proposed model in TensorFlow [141]. The input image size is 128×128. For each object category, we trained separate models. The encoder outputs a latent feature of dimension 256. The generator network outputs a point cloud of size 2048×3. Adam optimizer [142] was used to optimize the network parameters with a learning rate of $5e^{-5}$ and a minibatch of size 32. We trained the model until the validation accuracy stopped increasing.

5.4.2 Dataset preparation

ShapeNet [68] is a large-scale synthetic 3D dataset that is widely used in 3D research such as 3D model retrieval and reconstruction. **ShapeNetCore** is a subset of the ShapeNet dataset that we used in our experiment. It is manually cleaned and aligned. It has more than $50K$ unique 3D models which cover 55 common object categories. We focus on 13 categories and use the $80\% - 20\%$ train-test split provided by [68]. The

input images provided by [116] are used during training, where each model is rendered from 24 different azimuth angles.

To show the generalization of the proposed method on real images, we tested it using **Pix3D** dataset [143]. Pix3D is a publicly available dataset of aligned real-world image and 3D model pairs. It contains a large diversity in terms of object shapes and backgrounds and is highly challenging. We will test and report the performance of the proposed method on the chair, sofa, and table categories from the Pix3D dataset.

5.4.3 Baselines

We tested the proposed model trained on the ShapeNet dataset. First, we test the proposed model without an initial point cloud and the model is an encoder-generator model that generates the point could directly from the input image. Then, we tested the model with an initial point cloud of a sphere shape. The feature vector generated from the encoder part is concatenated with the 3D coordinates of each point. The generator generates the final point cloud from the initial point cloud and the input image feature vector. Also, we compareed the proposed model against PSGN [123] and 3D-LMNet [137] qualitatively and quantitatively. Chamfer distance (Eq. 5.3) and Earth Mover's distance (Eq. 5.4) are used to report the quantitative evaluation.

5.5 Experimental Results & Comparisons

In this section, we discuss the performance of the proposed model and report it quantitatively and qualitatively. First, we show and discuss the general results and how the proposed model generates the 3D point cloud from a single image on synthetic images. Then, we show the performance of the model with different setups. The performance is also compared against the state-of-the-art. Finally, the proposed method is tested on real images to check its generalization.

5.5.1 General Results

We tested the proposed model on the testing set of ShapeNet. The proposed model was trained on synthetic images of objects rendered from different viewpoints. The testing was performed on 13 different categories. Figure 5.4 shows the qualitative results of 8 different categories. It is clearly demonstrated that the generated point cloud of the objects from a single view is very close to the ground-truth and it captures the object geometry. Also, the proposed model learned to generate the point clouds and keeps the salient features such as free spaces between the splats in the back of the chair and the holes between the back and the seat of the bench. Moreover, the proposed model successfully learned to generate some thin and rare parts such as the stretchers between the chair legs as these parts are not common in the chair category. Many categories have various geometrical shapes such as the top surface of the tables. In Figure 5.4 (last row), two different tables are used to generate the point cloud. They have different top surfaces and different legs' shape. The proposed model generates the circle surface accurately as the input image with the cylindrical pillar and the four small legs. For the second table, the top surface is well reconstructed and the legs are very close to the ground-truth. Furthermore, the proposed model generates complete and plausible shapes. The generated points are evenly distributed and cover the whole parts of the objects.

However, the proposed model fails to generate very accurate shapes in some cases. Figure 5.5 shows some failure cases. Most thinner and narrower parts of the objects are missed such as the chair or bench armrests and the airplane tail. Also, the objects with extra parts that don't usually exist are also missed such as a monitor with two bases or a table with three legs on each side. Normally, the narrow and extra parts are missed because the network didn't learn to predict them. However, if this happens in one example, the network tries to generate and estimate the closest shape to the input image as the table with the six legs in Figure 5.5 (the last row). The proposed model reconstructs and generates a plausible point cloud that is close to the input image but it misses the leg in the middle.

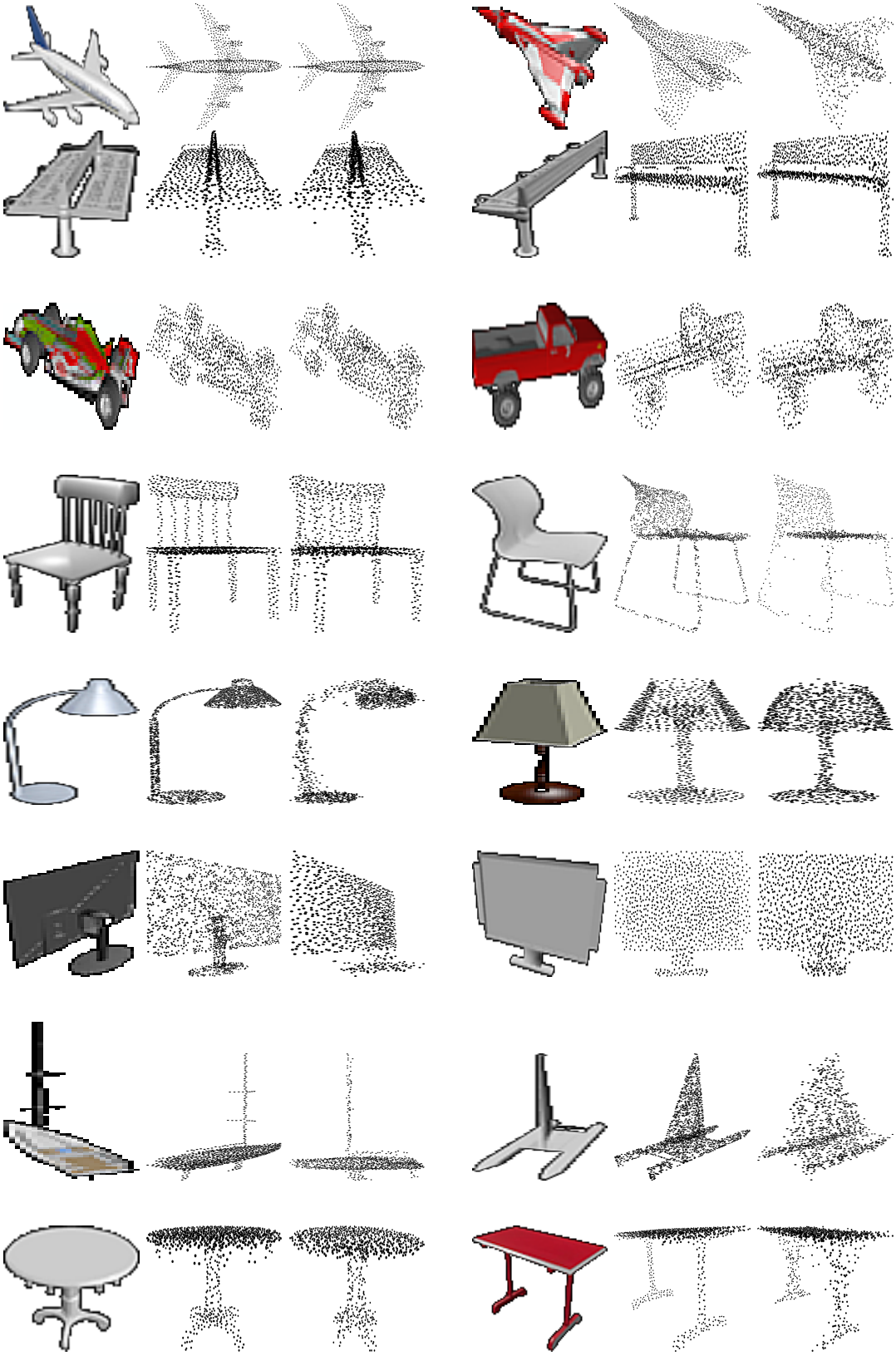


Figure 5.4: Qualitative results of ShapeNet on different categories. From left to right: input image, ground-truth, generated point cloud.

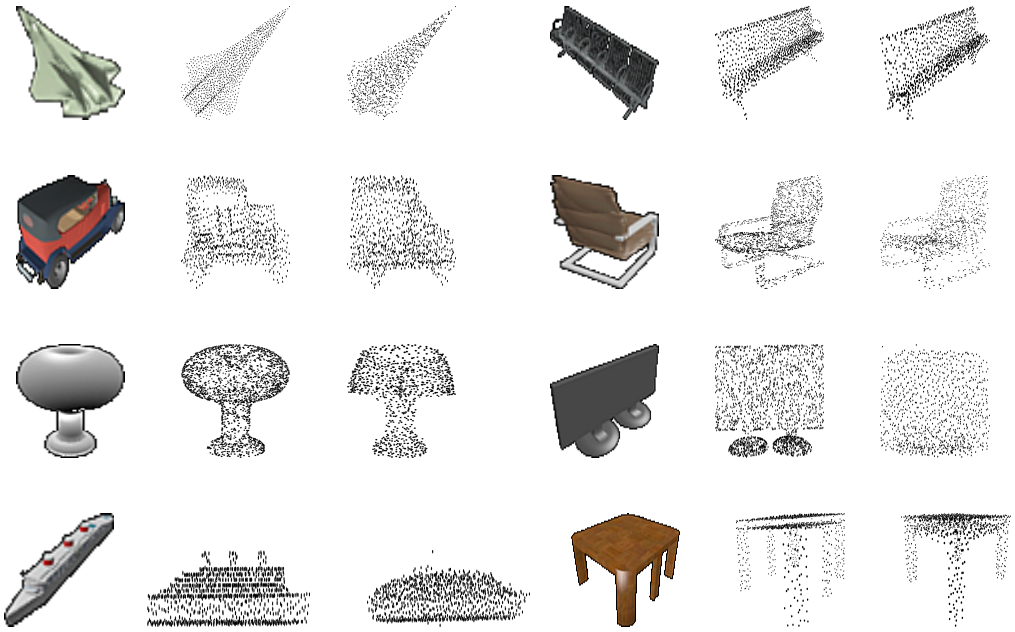
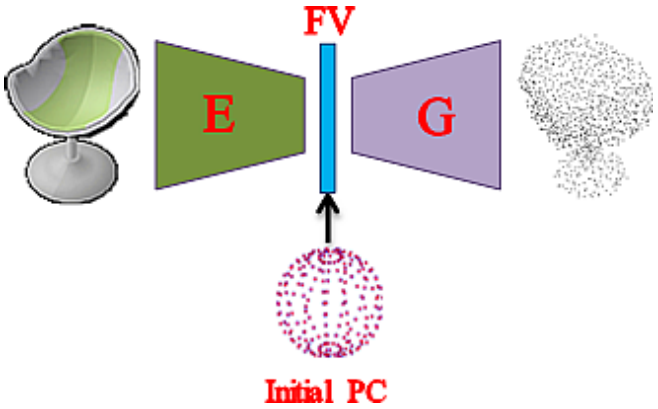


Figure 5.5: Failure cases of ShapeNet on airplane class from two viewpoints. From left to right: input image, ground-truth, generated point cloud.

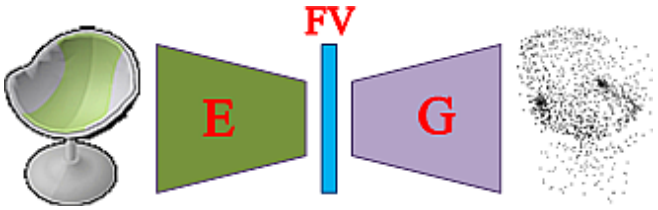
5.5.2 Effect of the Initial Point Cloud

To test the efficacy of using the initial point cloud in reconstructing a finer point cloud, we conducted an experiment to test and evaluate the performance of two different setups of the proposed model as shown in Figure 5.6. The first setup is shown in Figure 5.6a that uses an initial point cloud, where the second setup is the same model but without using the initial point cloud (Figure 5.6b), and the point cloud is reconstructed directly from the input image. Both setups were trained on the training set of ShapeNet and were tested on the testing set of the same dataset.

Figure 5.7 illustrates the results of the different setups of the proposed model. The point clouds generated by the proposed model without using an initial point cloud suffer from the uneven distribution of the points for the whole shape. Many points gather at some parts of the shape. In the chair example, many points are grouped at the back corners of the seats and fewer points are in the legs. However, the points are well distributed and the chair legs are well reconstructed using the model with an initial point cloud. Also, in the table examples, the point cloud generated without an initial point cloud have poor reconstructed legs but they are well reconstructed using



(a) CNN with an initial point cloud.



(b) CNN without an initial point cloud.

Figure 5.6: A general sketch of the proposed CNN model with different setups. (a) the proposed CNN with an initial point cloud. (b) the proposed CNN without the initial point cloud. E: Encoder, G: Generator, PC: Point Cloud, FV: Feature Vector.

an initial point cloud during training. In the plane examples, the engines and the trail are not reconstructed and the points are concentrated on the body of the plane, but they are reconstructed accurately when using the initial point cloud. From Figure 5.7, we conclude that adding the initial point cloud to the proposed model improves the reconstructed point cloud and distributes the points evenly on the whole shape's parts and generates the object details accurately.

Quantitatively, Table 5.1 reports a comparison between the different setups of the model. It is clearly noticed that the model with the initial point cloud outperforms the same model without using the initial point cloud.

5.5.3 Generating Plausible Shapes from Ambiguous 2D Inputs

To validate the performance of the proposed model, we conducted an experiment to test the model whether it can recognize and generate plausible shapes from 2D images of the chair class where the geometry of the objects is almost covered (the back-view

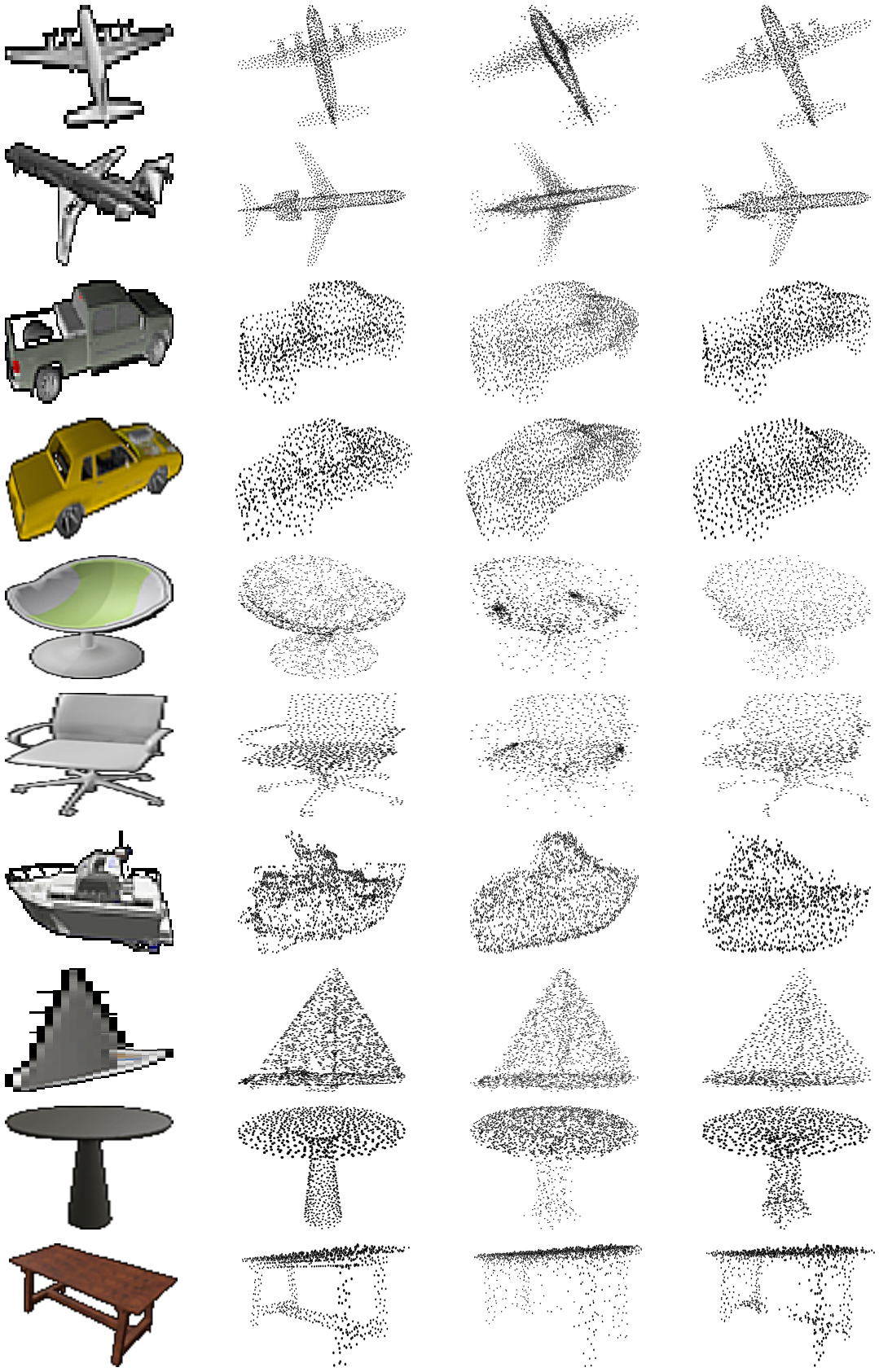


Figure 5.7: Qualitative results of different setups of the proposed model on ShapeNet. From left to right: input image, ground-truth, results generated by the proposed model without initial point cloud, and results generated by the proposed model with initial point cloud.

Table 5.1: Quantitative comparison of different setups of the proposed model on ShapeNet. All metrics are scaled by 100.

Category	Chamfer		EMD	
	w/o. PC	w. PC	w/o. PC	w. PC
airplane	4.03	**3.29**	4.91	**3.82**
bench	**4.34**	4.59	10.20	**4.31**
cabinet	**5.97**	6.07	11.18	**4.94**
car	**4.21**	4.39	4.69	**3.61**
chair	7.00	**6.48**	7.30	**6.45**
lamp	**6.31**	6.58	32.08	**8.45**
monitor	6.62	**6.39**	19.83	**5.94**
rifle	**2.71**	2.89	11.06	**4.25**
sofa	6.49	**5.85**	6.24	**5.03**
speakers	**7.86**	8.39	20.61	**7.37**
table	6.47	**6.26**	7.00	**6.05**
telephone	**4.03**	4.27	6.36	**3.77**
vessel	5.64	**4.55**	6.58	**4.89**
Mean	5.52	**5.38**	13.7	**5.30**

of the chair). Figure 5.8 shows the qualitative results of this experiment. For each image, we show the back and the side views of the reconstructed model along with the ground-truth with the same viewpoint. It is clearly shown that the proposed model succeeded in guessing the 3D geometry of the input image and generating plausible shapes that are consistent with the input images and the ground-truth. Also, the proposed model manages to memorize and reconstruct the chair parts such as the legs and the arms without seeing them in the 2D input images. Figure 5.8 proves that the proposed model can generate plausible shapes that are consistent with the ambiguous 2D images and are close enough to the ground-truth.

5.5.4 Arithmetic Operations on the 2D Input Image Feature Vector

Another interesting experiment is to check if the extracted 2D features from the input images have meaningful information or not. To do so, we extract the 2D features from different 2D images of the same category and apply arithmetic operations on them to generate a new 3D shape. In [144], it was shown that **(King)−vector(Man)+vector(Woman) gives a vector that the nearest neighbor to it was a vector for Queen**. The experiment performs similar to this idea.

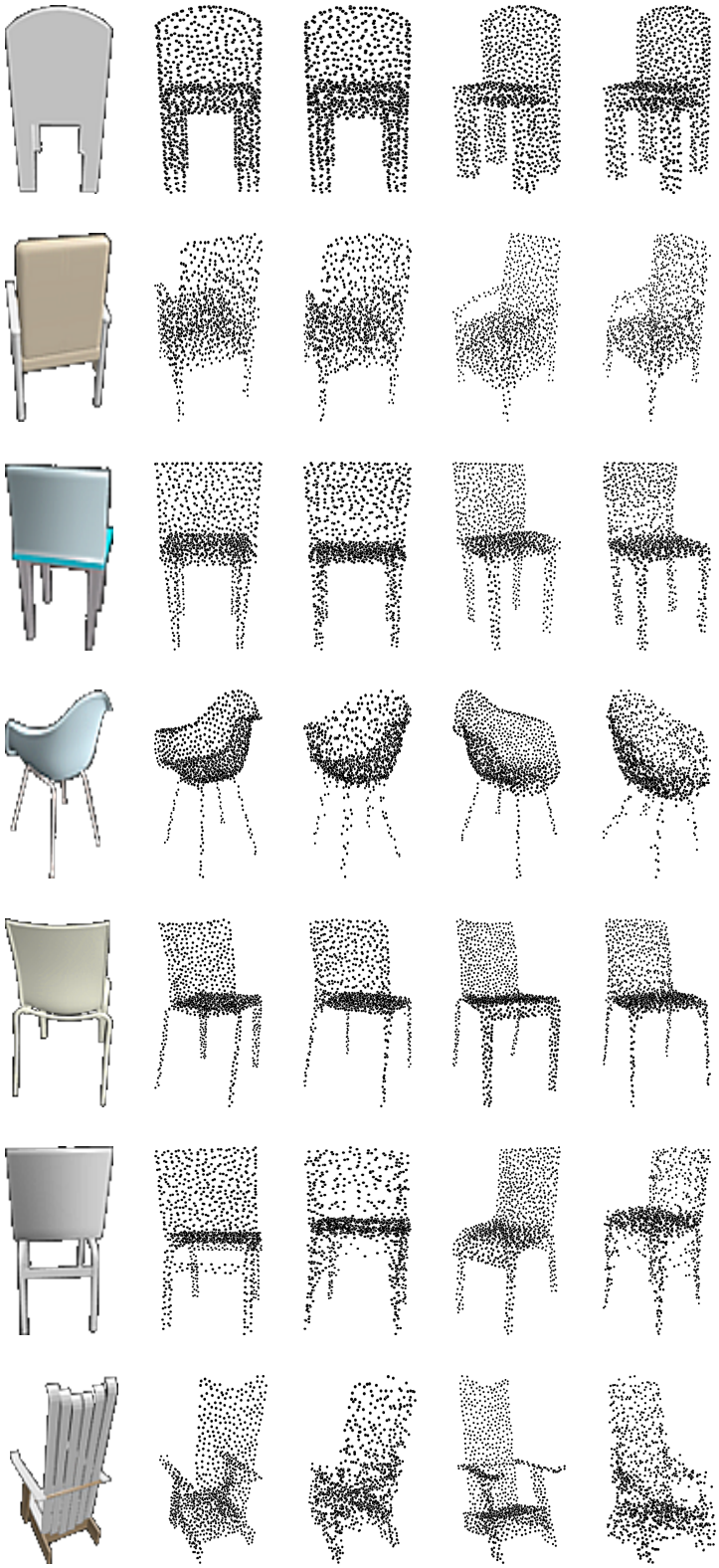
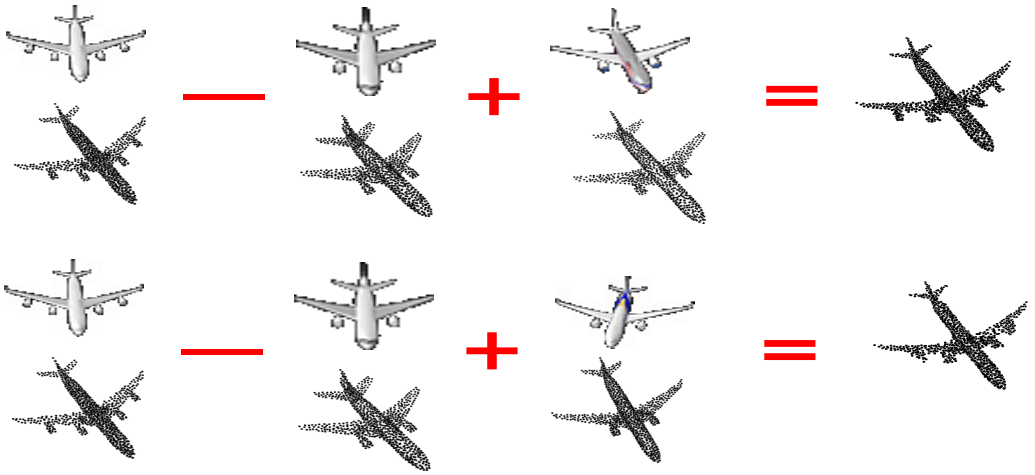
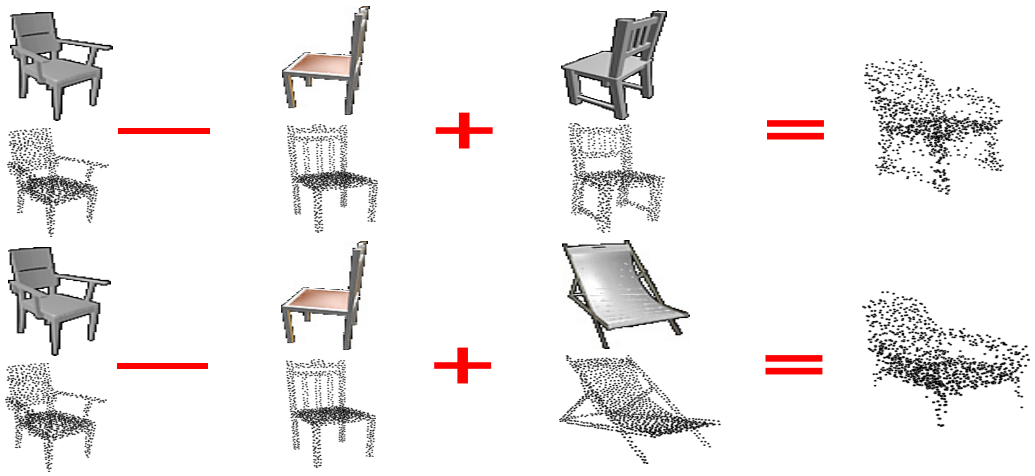


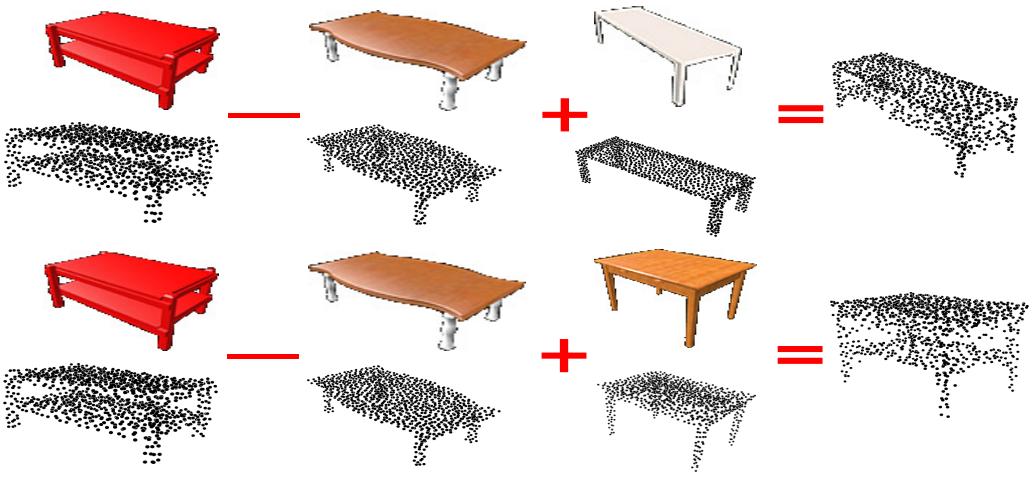
Figure 5.8: Qualitative 3D reconstruction results for ambiguous 2D inputs. From left to right: 2D input image, grount-truth view-1, generated output view-1, ground-truth view-2, generated output view-2.



(a) airplane



(b) chair



(c) table

Figure 5.9: Results of applying arithmatic operations on 2D features extracted by the encoder for different shapes.

We select random triples, extract their 2D features using the encoder network, and apply the arithmetic operations ($fv_1 - fv_2 + fv_3$). The resulting feature is then passed to the generator to generate the 3D point cloud.

Figure 5.9 shows the results of applying the arithmetic operations of some categories. The first experiment was applied to the airplane category. In Figure 5.9a, the first image is an airplane with two engines on each side and the second image is an airplane with one engine on each side. We subtract the extracted features of both images and then add the difference to the third image of an airplane that has just one engine on each side. As shown in Figure 5.9a, the generated new shape is an airplane that has two engines on each side. This means that the difference between the first two images generates a feature of an engine and then adds it to the third image results in a new airplane with two engines.

The second example was applied to the chair category. The main image is for a chair with arms. The other images are chairs without arms. We want to test if we can subtract the arms from the first shape and add them to the new shape. Figure 5.9b shows that when subtracting the feature of a chair that doesn't have arms from a chair that has arms and then adds the new feature to a third one we get the same shape of the third chair but with arms. This means that the difference between the two features generates a feature that has the chair arms information. And when adding this feature to a new image generates a shape that is similar to the input image that contains the transferred arms.

A third example was applied to the table category. The first image is for a table with a bottom shelf and the second image is for a table without the bottom shelf. When we subtract the feature of the second image from the feature of the first image and add the result to the third feature of a new image results in a table with the bottom shelf. The generated table is similar to the third image plus the bottom shelf. As can be seen in Figure 5.9c, the generated tables are similar to the third images where, for example, the table with long legs preserves its geometry after adding the new feature.

As shown in Figure 5.9, the proposed model extracts meaningful features that contain meaningful information. These features can be used to generate real shapes that have extra parts.

5.5.5 Comparison Results against Other Methods

We benchmark our proposed model against PSGN [123] and 3D-LMNet [137]. Both models were trained on the same training set of ShapeNet. PSGN is the first model to solve the problem of single-view reconstruction using CNN that generates point cloud. The reported results in [123] show that the point cloud-based models outperform the state-of-the-art voxel-based models significantly. Table 5.2 reports the comparison results of our proposed model against PSGN and 3D-LMNet on ShapeNet dataset. It is clearly shown in the table that our proposed model outperforms PSGN in 8 out of 13 categories in the Chamfer metric and all 13 categories in the EMD metric. Also, our proposed model outperforms 3D-LMNet in 6 out of 13 categories in the Chamfer metric and all 13 categories in the EMD metric. Overall, the average performance of our proposed model outperforms both models in both metrics despite that our proposed model is simple and efficient comparing with the others. Looking deeper into Table 5.2, EMD values denote better visualization of the generated point cloud of the objects. Also, since EMD is a point-to-point distance, it results in a high penalty when computing the distance between the points, and the two point cloud sets should have the same number of points. In Chamfer distance, the nearest points are used to calculate the distance in a forward manner (from the generated point cloud to the ground-truth) and in a backward manner (from the ground-truth to the generated point cloud). It is not necessary that the generated point cloud and the corresponding ground-truth have the same number of points.

Figure 5.10 highlights the qualitative comparison. It is clearly shown that the generated point clouds by our proposed model are visualized better than the ones generated by PSGN and 3D-LMNet. Our proposed model captures the details of the object and generates the object parts more accurately. In the riffle image (Figure 5.10, $8th$ row), the small parts of the riffle are captured in more detail compared to the

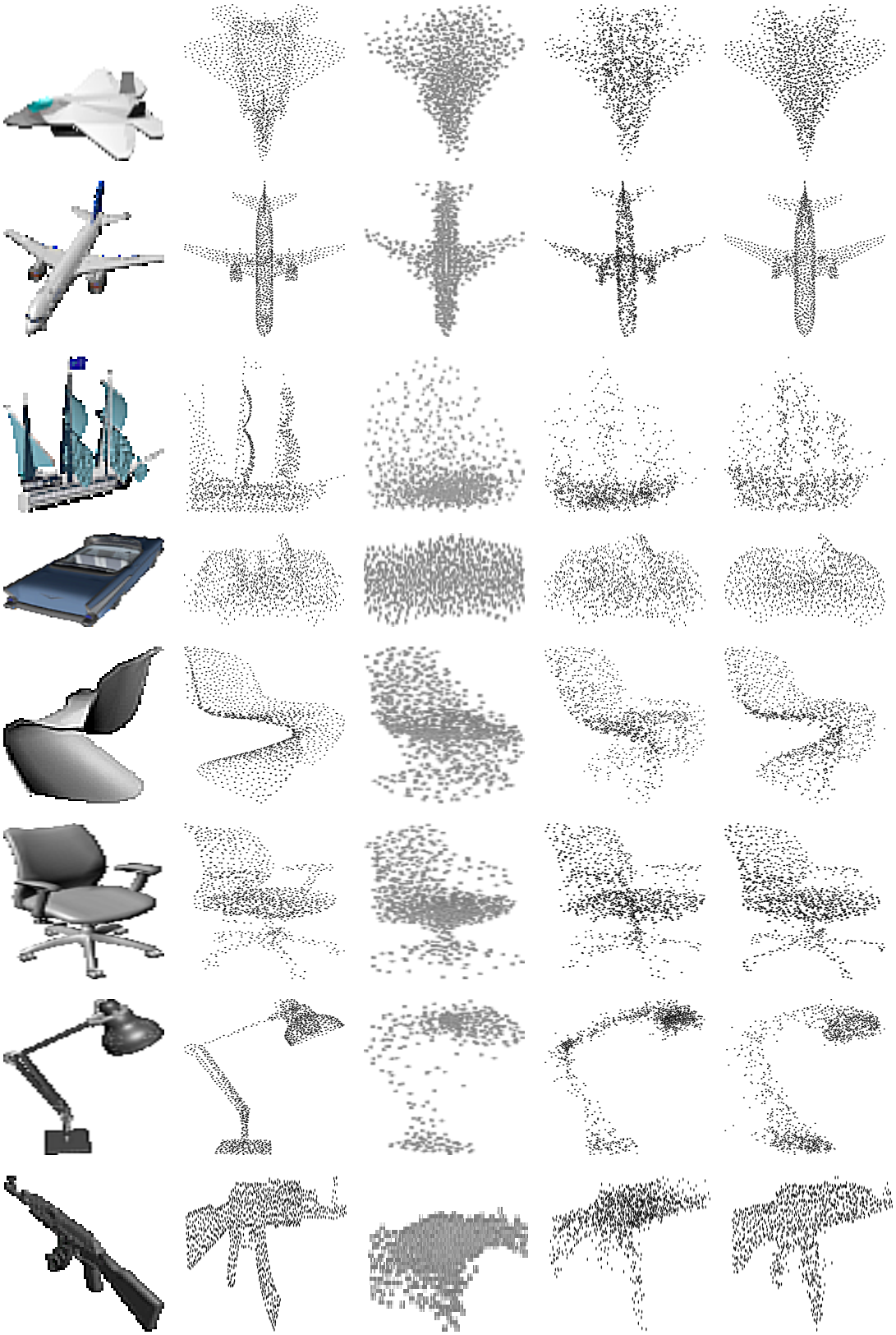


Figure 5.10: Comparison results between different methods on ShapeNet. From left to right: input image, ground-truth, resutls generated from PSGN, results generated from 3D-LMNet, and results generated from the proposed model.

Table 5.2: Quantitative comparison of single-view reconstruction results on ShapeNet dataset. The metrics are computed on 1024 points after performing ICP alignment with the ground-truth point cloud. All metrics are scaled by 100.

Category	Chamfer			EMD		
	PSGN	3D-LMNet	Ours	PSGN	3D-LMNet	Ours
airplane	3.74	3.34	**3.29**	6.38	7.44	**3.82**
bench	4.63	**4.55**	4.59	5.88	4.99	**4.31**
cabinet	6.98	6.09	**6.07**	6.04	6.35	**4.94**
car	5.20	4.55	**4.39**	4.87	4.10	**3.61**
chair	**6.39**	6.41	6.48	9.63	8.02	**6.45**
lamp	**6.33**	7.10	6.58	16.17	15.8	**8.45**
monitor	**6.15**	6.40	6.39	7.59	7.13	**5.94**
rifle	2.91	**2.75**	2.89	8.48	6.08	**4.25**
sofa	6.98	**5.85**	**5.85**	7.42	5.65	**5.03**
speakers	8.75	**8.10**	8.39	8.70	9.15	**7.37**
table	**6.00**	6.05	6.26	8.40	7.82	**6.05**
telephone	4.56	4.63	**4.27**	5.07	5.43	**3.77**
vessel	4.38	**4.37**	4.55	6.18	5.68	**4.89**
Mean	5.62	5.40	**5.38**	7.75	7.00	**5.30**

generated point cloud of 3D-LMNet as it doesn't generate the grip or the magazine and in PSGN where these parts are almost fused with each other and they cannot be separated. Also, our proposed model generates a well-distributed point cloud that the points are fairly distributed on the whole shape and not concentrated in one part or at the center of the shape. Also, this can be noticed in the chair image (Figure 5.10, $6th$ row) where our proposed model successfully generates and separates the chair legs and the armrest. Thanks to the initial point cloud that helps in generating a well-distributed point cloud. However, the other models have considered them either a fully connected part of the chair (*e.g.* the armrest) or one part (*e.g.* the chair legs).

5.5.6 Pix3D Dataset Results

The proposed model was trained on synthetic images that are clean and the objects appear well in the images. To test the performance of the model in real scenarios, the Pix3D dataset is used. This dataset contains a large collection of real images and the corresponding metadata such as masks along with ground-truth 3D CAD models

Table 5.3: Single-view reconstruction results on the real world Pix3D dataset. All metrics are scaled by 100.

Category	Chamfer			EMD		
	PSGN	3D-LMNet	Ours	PSGN	3D-LMNet	Ours
chair	8.05	7.35	**6.82**	12.55	9.14	**7.45**
sofa	8.45	8.18	**3.95**	9.16	7.22	**3.28**
table	10.82	11.20	**5.22**	15.16	12.73	**5.17**
Mean	9.11	8.91	**5.33**	12.29	9.70	**5.30**

of different object categories. The shared categories between ShapeNet and Pix3D datasets are used to test and evaluate the proposed method. The testing images are preprocessed. The images are cropped to center-position the object of interest in the image, and the background is masked. Then the image is resized to match the training image size (128×128). The proposed model isn't fine-tuned on the Pix3D dataset but it is directly tested on the images.

Table 5.3 reports the quantitative results of testing Pix3D images on the proposed model against PSGN and 3D-LMNet. The three models were trained on ShapeNet and tested on Pix3D. The reported numbers are taken from [137]. It is clear that the proposed model outperforms the other models by a large margin in both metrics and on all object categories. This demonstrates the efficiency of the proposed model on real data.

Figure 5.11 visualizes the reconstruction results of some selected Pix3D images generated from the proposed model along with 3D-LMNet. 3D-LMNet performs well on real-world images, but our model performs better and the generated point cloud is more accurate and very similar to the ground-truth. Our model distributes the points evenly on the whole object shape and covers the object parts accurately. This shows that the proposed model generalizes well to the real-world images and generates accurate models that describe the input images, even though the images are from a different distribution than the training set.

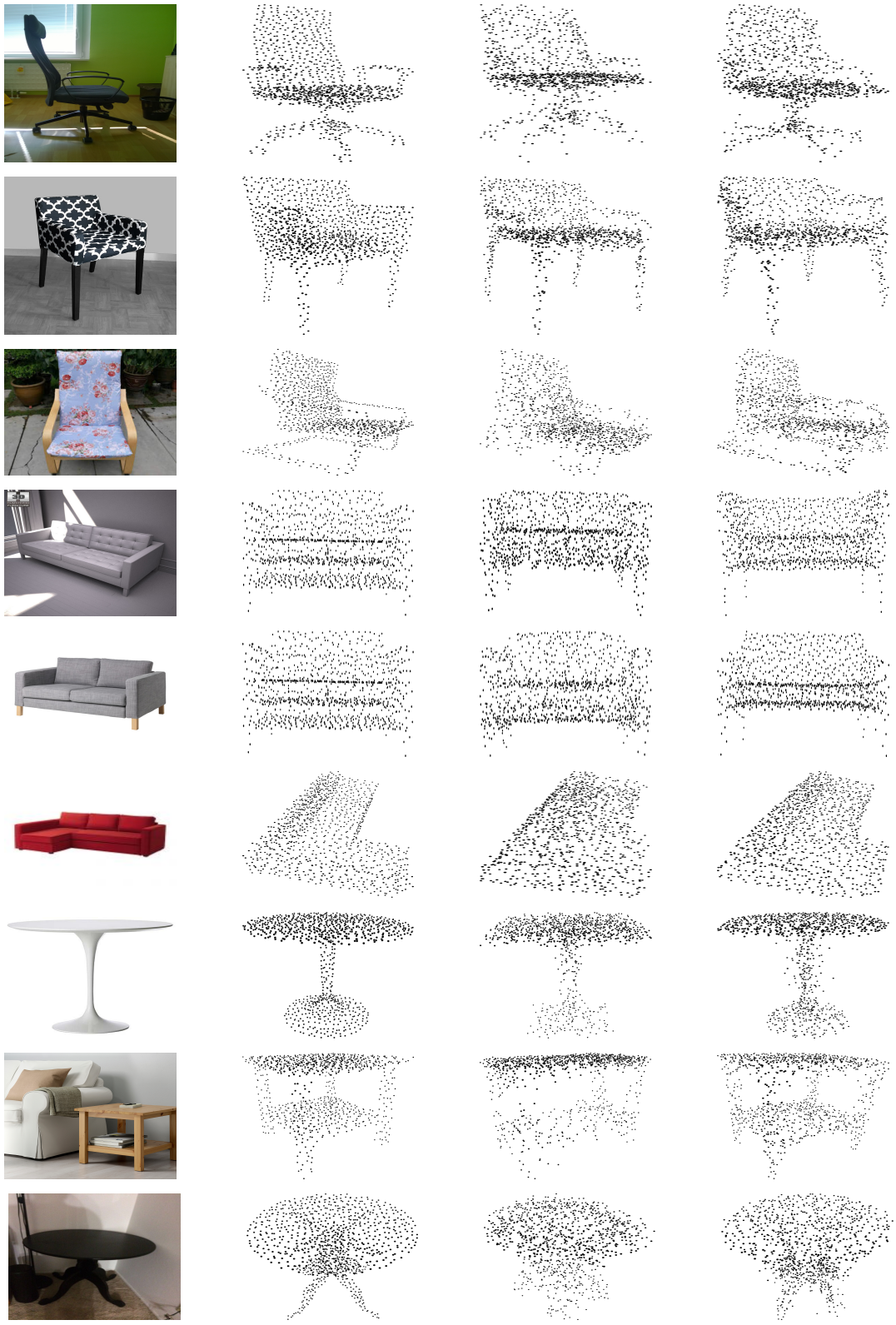


Figure 5.11: Qualitative results on chair, sofa, and table categories from Pix3D dataset. From left to right: input image, ground-truth, results generated from 3D-LMNet, and results generated from the proposed model.

5.6 Discussion & Conclusion

Though single-view 3D object reconstruction is a challenging task, the well-created human eyes and the brain have the ability to infer and predict the geometry of a scene and the objects within it from a single image. With more complicated scenarios such as high occlusion of the objects, the human brain is able to guess a number of plausible shapes that could match what is seen. This is because of the prior information that is stored in the human brain and is retrieved, utilized, and updated when seeing new scenes. Recently, different research fields exploit the ability to reconstruct objects from a single image in many applications such as the field of robotics in object grasping and manipulation. However, it is an ill-posed problem and many plausible reconstructions could be a solution for one single view due to the uncertainty.

In this chapter, we have proposed a simple, yet powerful CNN model to generate the point cloud of an object from a single image. 3D data can be represented in different ways. Point clouds have proven to be a common and simple representation (See Eq. 5.1). The proposed model trained end-to-end on synthetic data with 3D supervision. It takes a single image of an object and generates a point cloud with a fixed number of points ($N = 2048$). An initial point cloud of a sphere shape is used to improve the generated point cloud. Qualitative and quantitative evaluation on synthetic and real data demonstrate that the proposed model is able to generate point clouds that are very close to the ground-truth and more accurate in comparison with other methods. Moreover, we show that the initial point cloud has improved the final results as it distributes the points on the whole object shape evenly. The qualitative results show that the points are grouped in some object parts densely and other parts have fewer points when the proposed model doesn't use the initial point cloud. Furthermore, the performance of the proposed model on the real-world dataset illustrates the outstanding generalization to the new and unseen images and scenes.

6

Conclusion and Outlook

You've achieved success in your field when you don't know whether what you're doing is work or play.

Warren Beatty

This dissertation systematically addresses the problem of understanding objects through a single image from geometry and semantics perspectives using deep learning. Object geometric understanding aims to capture the 3D information of the object within the 3D environment. This includes object viewpoint estimation, object depth prediction, and 3D object reconstruction. Object semantics understanding focuses on retrieving semantic information related to the object such as object classification.

This chapter summarizes the contributions and findings of the work discussed in the dissertation. Section 6.1 discusses the contributions and outcomes of the proposed methods. Section 6.2 lists some limitations and difficulties that emerged during the experiments and the results. In addition, it highlights some potential future directions that help in extending the research in the same direction.

6.1 Summary

Nowadays, deep learning models have proven their significance in solving computer vision tasks. In this dissertation, we proposed different deep learning models to understand the geometry and semantics information of the objects from a single image. These models reduced the need for hand-crafted features engineering and train efficiently thanks to the available datasets. We approached the object understanding task by addressing the following target tasks. Firstly, we studied the task of joint object classification and viewpoint estimation. Then, we addressed the depth estimation task by proposing different CNN models and defined it as pixel-wise depth estimation. Finally, we proposed a simple, yet powerful CNN model to generate 3D point clouds of objects by the guidance of an initial sphere. The findings and the outcomes of this dissertation are summarized as follows:

- Object classification and viewpoint estimation are important tasks in scene understanding. Looking closer at the nature of these tasks, we can conclude that the represented features of both tasks are opposite to each other. On one hand, to classify an object correctly, the network should learn it from different orientations. On the other hand, the network has to learn the orientation of the object accurately so that it can recognize the object's direction. Thus, proposing a network capable of solving these tasks requires a careful design. In Chapter 3, we proposed a multi-task CNN model for joint object classification and viewpoint estimation with a known bounding box around the object in the image. In designing the network, many conditions were considered such as the extracted features and the shared layers. As a result, the shared network performed as a feature extractor while separated networks performed as task-specific feature extractors. We showed that the proposed multi-task network achieved better results than other models for this joint task on the Pascal3D+ dataset.

- Another finding in the direction of object geometry understanding is the dense pixel-wise depth prediction from a single image. Depth estimation from a single

image is an extremely challenging task. In Chapter 4, we proposed two different end-to-end and single-shot trainable CNN models to estimate the depth of objects in a scene from a single image. The first model is a fully CNN that predicts the depth values from a single input image (Section 4.3.1). However, the proposed model generates blurry and low-resolution depth image. To overcome this problem, we proposed another model that solves the previous issues. Mini V-net model (Section 4.3.2) generates a depth image that has the same resolution as the input image with sharper features and displays more object details. Besides the well-designed models, the chosen loss function influences the model accuracy. We proposed to use a non-convex loss function (Tukey's biweight loss function) to train both models (Section 3.1). The quantitative and qualitative results proved the importance of selecting the suitable loss function along with well-designed models and how these factors affect the final results.

- For 3D object reconstruction from a single image, we proposed a simple, yet powerful CNN model that can generate the 3D geometry of an object based on point cloud representation (Chapter 5). The proposed model trained end-to-end on a completely synthetic dataset. It utilizes an initial point cloud of a sphere to generate well-distributed points and shape the geometry structure of the object. The quantitative and qualitative results demonstrated that the proposed single model outperforms other methods on both synthetic and real datasets. Moreover, The encoder network learned to extract useful features from the 2D images that can be used to apply the arithmetic operations on the features and generate plausible objects. Furthermore, the model succeeded to infer plausible 3D point clouds of objects from the single input images of ambiguous views.

All the previous contributions were achieved with the help of the data availability and the carefully chosen loss function. In the object classification and viewpoint estimation task, the proposed multi-task model was trained on a completely synthetic dataset. We rendered many images from 3D synthetic models from different viewpoints. This helped us in overcoming the problem of annotated data. Also, for the 3D object

reconstruction task, we trained the proposed model on synthetic data. For both models, we tested and proved the generalizability of the trained models on real data and compared their performance against other methods. The trained models outperform other methods on both tasks. Thus, we can conclude that synthetic data can be used to train CNN models and overcome the problem of annotated data.

With respect to the loss function, we saw that the carefully selected loss function improved the performance of the model. In the depth estimation task (Chapter 4), we used a non-convex loss function and compared it against the L2 norm as another loss function normally used for regression problems. We found that the model optimized using the non-convex loss function outperforms the same model optimized using the L2 norm. This demonstrates that the loss function is an essential factor that should be carefully considered when training the model. Loss function selection depends on the task that we want to train the model to solve.

To summarize, the proposed solutions for the above-mentioned tasks were a result of modeling the following decisions: **1)** the nature of the problem (*e.g.* classification, regression, *etc.*), **2)** the available dataset and how it is represented, and **3)** the carefully chosen loss function.

6.2 Limitations and Future Work

While the proposed CNN models, presented in the dissertation, proved the capability to solve many tasks, there are still some limitations and potential suggestions for future work.

- CNN models are always data-hungry due to the millions of parameters to be optimized. To achieve a high performance of a CNN, a large-scale and well-annotated dataset should be available. However, collecting and annotating real data are very expensive since the collection process is time-consuming, and the annotation process needs experts for annotating and verification. One solution to overcome the lack of data problem is to use synthetic data for training. Generally, synthetic data are considered as error-free ground-truth because the

whole synthesizing process is fully-controlled. We have shown that the synthetic data can be used for training where we rendered the images from the synthetic models and overlayed them with real backgrounds (Section 3.5). It is also easy to annotate them for the target task. However, there is a gap between the real data and the synthetic data used for training as the synthetic data are not enough to ensure the generalizability of the model. Some works attempt to generate real and natural-looking images using GANs [145]. GANs are promising models which can be used to generate synthetic images that look like the real ones. But still, some generated images could contain incompatible structures. This can be solved by adding some semantic cues while generating the images. Another direction to overcome the lack of data issue is to consider semi-supervised and unsupervised techniques for training. Leveraging large amounts of unlabeled data, besides a small amount of annotated data, enables the understanding of the structure of the data. This is because we are interested in the learned intermediate representation which can carry useful semantic meanings that can be beneficial for other tasks. This is an open direction of how the unlabeled data can be incorporated with the labeled data in training CNN models.

- As we explained in Chapter 5, there are different representations for the 3D data (multiple views, voxel, mesh, and point cloud). Each representation has its own benefits and limitations. However, it is still unclear which one is the most appropriate representation for 3D data. With voxel-based representation, the shape has limited details in a simple representation. When the shape representation has to be dense with more details, point cloud or mesh-based representation is used. However, extending the available CNN models to work directly on 3D data is a fresh area of research. PointNet [146] and PointNet++ [147] are examples where the proposed models work directly on point cloud-based data for classification and segmentation. It is an interesting direction to study how different 3D representations perform in solving different tasks, and how to achieve full geometric and semantic understanding for objects with respect to the invariance property (*e.g.* rotation).

- One last direction is to investigate and explain the internal mechanism of the CNN models and investigate why they succeed in solving different computer vision tasks. During the training process, it is still ambiguous how the layers learn useful features from the most salient regions of the images, how some filters learn to extract the object edges, and other filters learn to extract the shape of the objects. Likewise, other questions arise like, why the earlier layers are extracting general features to the level that the learned parameters can be transferred to other models while the last layers are considered as task-specific layers. One attempt to understand the internal operation and performance of CNNs is by visualizing the features learned by the layers [30]. Though the empirical success of CNNs, the internal understanding and the theoretical proof behind the success of the deep learning models in solving computer vision tasks are still missed.

To summarize, deep learning influences the computer vision community in the current decade. This dissertation opens many research questions and opportunities that will lead to a better understanding of future work.

References

[1] Y. Xiang, R. Mottaghi, and S. Savarese. **Beyond pascal: A benchmark for 3d object detection in the wild**. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82. IEEE, 2014.

[2] B. Pepik, M. Stark, P. Gehler, and B. Schiele. **Teaching 3d geometry to deformable part models**. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3362–3369. IEEE, 2012.

[3] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. **Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views**. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[4] S. Mahendran, H. Ali, and R. Vidal. **3d pose regression using convolutional neural networks**. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2174–2182, 2017.

[5] S. Tulsiani and J. Malik. **Viewpoints and keypoints**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015.

[6] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Kosecka, and A. C. Berg. **Fast single shot detection and pose estimation**. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 676–684. IEEE, 2016.

[7] C. H. Anderson, D. C. Van Essen, and B. A. Olshausen. **Directed visual attention and the dynamic control of information flow**. In *Neurobiology of attention*, pages 11–17. Elsevier, 2005.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. **Imagenet classification with deep convolutional neural networks**. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] K. Simonyan and A. Zisserman. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. In *International Conference on Learning Representations*, 2015.

[10] 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*, pages 770–778, 2016.

[11] A. Bottino, L. Jaulin, and A. Laurentini. **Reconstructing 3D objects from silhouettes with unknown viewpoints: The case of planar orthographic views**. In *Iberoamerican Congress on Pattern Recognition*, pages 153–162. Springer, 2003.

[12] J.-S. Franco and E. Boyer. **Fusion of multi-view silhouette cues using a space occupancy grid**. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, **2**, pages 1747–1753. IEEE, 2005.

[13] K. N. Kutulakos and S. M. Seitz. **A theory of shape by space carving**. *International journal of computer vision*, **38**(3):199–218, 2000.

[14] Y. Furukawa and J. Ponce. **Accurate, dense, and robust multiview stereopsis**. *IEEE transactions on pattern analysis and machine intelligence*, **32**(8):1362–1376, 2009.

[15] M. Kazhdan, M. Bolitho, and H. Hoppe. **Poisson surface reconstruction**. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, **7**, 2006.

[16] M. R. Oswald, E. Töppe, C. Nieuwenhuis, and D. Cremers. **A review of geometry recovery from a single image focusing on curved object reconstruction**. In *Innovations for Shape Analysis*, pages 343–378. Springer, 2013.

[17] B. K. Horn. **Shape from shading: A method for obtaining the shape of a smooth opaque object from one view**. 1970.

[18] F. Ulupinar and R. Nevatia. **Shape from contour: Straight homogeneous generalized cylinders and constant cross section generalized cylinders**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(2):120–135, 1995.

[19] M. Prasad, A. W. Fitzgibbon, and A. Zisserman. **Fast and Controllable 3D Modelling From Silhouettes.** In *Eurographics (Short Presentations)*, pages 9–12, 2005.

[20] M. R. Oswald, E. Töppe, K. Kolev, and D. Cremers. **Non-parametric single view reconstruction of curved objects using convex optimization**. In *Joint Pattern Recognition Symposium*, pages 171–180. Springer, 2009.

[21] D. Hoiem, A. A. Efros, and M. Hebert. **Automatic photo pop-up**. In *ACM SIGGRAPH 2005 Papers*, pages 577–584. 2005.

[22] E. Töppe, M. R. Oswald, D. Cremers, and C. Rother. **Image-based 3d modeling via cheeger sets**. In *Asian Conference on Computer Vision*, pages 53–64. Springer, 2010.

[23] W. Hong, A. Y. Yang, K. Huang, and Y. Ma. **On symmetry and multiple-view geometry: Structure, pose, and calibration from a single image**. *International Journal of Computer Vision*, **60**(3):241–265, 2004.

[24] F. Han and S.-C. Zhu. **Bayesian reconstruction of 3d shapes and scenes from a single image**. In *First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis, 2003. HLK 2003.*, pages 12–20. IEEE, 2003.

[25] P. Koutsourakis, L. Simon, O. Teboul, G. Tziritas, and N. Paragios. **Single view reconstruction using shape grammars for urban environments**. In *2009 IEEE 12th international conference on computer vision*, pages 1795–1802. IEEE, 2009.

[26] D. Rother and G. Sapiro. **Seeing 3D objects in a single 2D image**. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1819–1826. IEEE, 2009.

[27] D. H. Hubel and T. N. Wiesel. **Receptive fields and functional architecture of monkey striate cortex**. *The Journal of physiology*, pages 215–243., 1968.

[28] B. B. L. Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. **Handwritten digit recognition with a back-propagation network**. *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 396–404., 1989.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. **Imagenet: A large-scale hierarchical image database**. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[30] M. D. Zeiler and R. Fergus. **Visualizing and Understanding Convolutional Networks**. *European Conference on Computer Vision*, pages 818–833, 2014.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. **Going deeper with convolutions**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[32] T. A. Soomro, A. J. Afifi, L. Zheng, S. Soomro, J. Gao, O. Hellwich, and M. Paul. **Deep learning models for retinal blood vessels segmentation: A review**. *IEEE Access*, **7**:71696–71717, 2019.

[33] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng. **Tiled convolutional neural networks**. *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, **1**:1279–1287, 2010.

[34] F. Yu and V. Koltun. **Multi-Scale Context Aggregation by Dilated Convolutions**. *The International Conference on Learning Representations (ICLR)*, pages 1–13, 2016.

[35] M. Lin and Q. Chen. **Network In Network**. *the International Conference on Learning Representations (ICLR)*, pages 1–10, 2014.

[36] V. Nair and G. E. Hinton. **Rectified linear units improve restricted boltzmann machines**. In *ICML*, 2010.

[37] A. L. Maas, A. Y. Hannun, and A. Y. Ng. **Rectifier Nonlinearities Improve Neural Network Acoustic Models**. *Proceedings of the International Conference on Machine Learning (ICML)*, **30**:1–6, 2013.

[38] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. **Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)**. *the International Conference on Learning Representations (ICLR)*, pages 1–14, 2016.

[39] M. H.N. and M. C.A. **How to choose an activation function.** *In Advances in Neural Information Processing Systems*, pages 319–326, 1994.

[40] Y.-L. Boureau, J. Ponce, and Y. LeCun. **A theoretical analysis of feature pooling in visual recognition**. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[41] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. **End-to-end text recognition with convolutional neural networks**. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 3304–3308. IEEE, 2012.

[42] K. He, X. Zhang, S. Ren, and J. Sun. **Spatial pyramid pooling in deep convolutional networks for visual recognition**. *IEEE transactions on pattern analysis and machine intelligence*, **37**(9):1904–1916, 2015.

[43] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, L. Wang, G. Wang, J. Cai, and T. Chen. **Recent advances in convolutional neural networks**. *arXiv preprint arXiv:1512.07108*, pages 1–38, 2015.

[44] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. **Improving neural networks by preventing co-adaptation of feature detectors**. *arXiv preprint arXiv:1207.058*, pages 1–18, 2012.

[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and L. F.-F. Alexander C. Berg. **Imagenet large scale visual recognition challenge**. *International Journal of Computer Vision*, **115**(3):211–252., 2015.

[46] S. Ioffe and C. Szegedy. **Batch normalization: Accelerating deep network training by reducing internal covariate shift**. *In International Conference on Machine Learning*, pages 448–456, 2015.

[47] X. Glorot and Y. Bengio. **Understanding the difficulty of training deep feedforward neural networks**. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[48] R. G. Wijnhoven and P. H. de With. **Fast training of object detection using stochastic gradient descent**. *In IEEE 20th International Conference on Pattern Recognition (ICPR)*, pages 424–427, 2010.

[49] N. Qian. **On the momentum term in gradient descent learning algorithms.** *Neural networks*, **12**(1):145–151., 1999.

[50] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. **On the importance of initialization and momentum in deep learning.** *In International conference on machine learning*, pages 1139–1147, 2013.

[51] R. Girshick, J. Donahue, T. Darrell, and J. Malik. **Rich feature hierarchies for accurate object detection and semantic segmentation**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[52] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. **Selective search for object recognition**. *International journal of computer vision*, **104**(2):154–171, 2013.

[53] S. Ren, K. He, R. Girshick, and J. Sun. **Faster r-cnn: Towards real-time object detection with region proposal networks**. In *Advances in neural information processing systems*, pages 91–99, 2015.

[54] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. **You only look once: Unified, real-time object detection**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[55] J. Long, E. Shelhamer, and T. Darrell. **Fully convolutional networks for semantic segmentation**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[56] O. Ronneberger, P. Fischer, and T. Brox. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. *International Conference on Medical Image Computing and Computer-Assisted Intervention Springer, Cham.*, **1**:234–241, 2015.

[57] V. Badrinarayanan, A. Kendall, and R. Cipolla. **Segnet: A deep convolutional encoder-decoder architecture for image segmentation**. *IEEE transactions on pattern analysis and machine intelligence*, **39**(12):2481–2495, 2017.

[58] H. Penedones, R. Collobert, F. Fleuret, and D. Grangier. **Improving object classification using pose information**. Technical report, Idiap, 2012.

[59] Y. Su, M. Allan, and F. Jurie. **Improving object classification using semantic attributes.** In *BMVC*, pages 1–10, 2010.

[60] G. R, D. J, D. T., and M. J. **Region-based convolutional networks for accurate object detection and segmentation**. *IEEE transactions on pattern analysis and machine intelligence*, **38**(1):142–158., 2016.

[61] H. Zhang, T. El-Gaaly, A. Elgammal, and Z. Jiang. **Joint object and pose recognition using homeomorphic manifold analysis**. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[62] N. Dalal and B. Triggs. **Histograms of oriented gradients for human detection**. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, **1**, pages 886–893. IEEE Computer Society, 2005.

[63] D. G. Lowe. **Distinctive image features from scale-invariant keypoints**. *International journal of computer vision*, **60**(2):91–110, 2004.

[64] H. Bay, T. Tuytelaars, and L. Van Gool. **Surf: Speeded up robust features**. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[65] A. J. Afifi and O. Hellwich. **Object depth estimation from a single image using fully convolutional neural network**. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7. IEEE, 2016.

[66] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. **How transferable are features in deep neural networks?** In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[67] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. **3d shapenets: A deep representation for volumetric shapes**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[68] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. **Shapenet: An information-rich 3d model repository**. *arXiv preprint arXiv:1512.03012*, 2015.

[69] A. J. Afifi, O. Hellwich, and T. A. Soomro. **Simultaneous Object Classification and Viewpoint Estimation using Deep Multi-task Convolutional Neural Network.** In *VISIGRAPP (5: VISAPP)*, pages 177–184, 2018.

[70] S. Branson, G. Van Horn, P. Perona, and S. J. Belongie. **Improved Bird Species Recognition Using Pose Normalized Deep Convolutional Nets.** In *BMVC*, **1**, page 7, 2014.

[71] D. Lin, X. Shen, C. Lu, and J. Jia. **Deep lac: Deep localization, alignment and classification for fine-grained recognition**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1666–1674, 2015.

[72] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. **Learning and transferring mid-level image representations using convolutional neural networks**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[73] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. **The pascal visual object classes (voc) challenge**. *International journal of computer vision*, **88**(2):303–338, 2010.

[74] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. **6-dof object pose from semantic keypoints**. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2011–2018. IEEE, 2017.

[75] A. Grabner, P. M. Roth, and V. Lepetit. **3d pose estimation and 3d model retrieval for objects in the wild**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3022–3031, 2018.

[76] R. Girshick. **Fast r-cnn**. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[77] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. **Ssd: Single shot multibox detector**. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[78] I. Kokkinos. **Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138, 2017.

[79] M. Elhoseiny, T. El-Gaaly, A. Bakry, and A. Elgammal. **A comparative analysis and study of multiview CNN models for joint object categorization and pose estimation**. In *International Conference on Machine learning*, pages 888–897. PMLR, 2016.

[80] A. Vedaldi and K. Lenc. **Matconvnet: Convolutional neural networks for matlab**. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

[81] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. **Return of the devil in the details: Delving deep into convolutional nets**. *arXiv preprint arXiv:1405.3531*, 2014.

[82] M. Schwarz, H. Schulz, and S. Behnke. **RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features**. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 1329–1335. IEEE, 2015.

[83] J. Wu, Y. Yu, C. Huang, and K. Yu. **Deep multiple instance learning for image classification and auto-annotation**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3460–3469, 2015.

[84] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. **CNN features off-the-shelf: an astounding baseline for recognition**. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[85] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. **Overfeat: Integrated recognition, localization and detection using convolutional networks**. *arXiv preprint arXiv:1312.6229*, 2013.

[86] L. Ladicky, J. Shi, and M. Pollefeys. **Pulling things out of perspective**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 89–96, 2014.

[87] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, et al. **Efficient human pose estimation from single depth images**. *IEEE transactions on pattern analysis and machine intelligence*, **35**(12):2821–2840, 2012.

[88] A. Saxena, M. Sun, and A. Y. Ng. **Make3d: Learning 3d scene structure from a single still image**. *IEEE transactions on pattern analysis and machine intelligence*, **31**(5):824–840, 2008.

[89] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun. **Learning long-range vision for autonomous off-road driving**. *Journal of Field Robotics*, **26**(2):120–144, 2009.

[90] D. Eigen, C. Puhrsch, and R. Fergus. **Depth map prediction from a single image using a multi-scale deep network**. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

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

[92] X. Ren, L. Bo, and D. Fox. **Rgb-(d) scene labeling: Features and algorithms**. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2759–2766. IEEE, 2012.

[93] R. Roberts, S. N. Sinha, R. Szeliski, and D. Steedly. **Structure from motion for scenes with large duplicate structures**. In *CVPR 2011*, pages 3137–3144. IEEE, 2011.

[94] K. Karsch, C. Liu, and S. B. Kang. **Depth extraction from video using non-parametric sampling**. In *European Conference on Computer Vision*, pages 775–788. Springer, 2012.

[95] F. Perronnin, J. Sánchez, and T. Mensink. **Improving the fisher kernel for large-scale image classification**. In *European conference on computer vision*, pages 143–156. Springer, 2010.

[96] F. Milletari, N. Navab, and S.-A. Ahmadi. **V-net: Fully convolutional neural networks for volumetric medical image segmentation**. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[97] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. **Striving for Simplicity: The All Convolutional Net**. In *ICLR (workshop track)*, 2015.

[98] A. E. Beaton and J. W. Tukey. **The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data**. *Technometrics*, **16**(2):147–185, 1974.

[99] A. J. Afifi, O. Hellwich, and T. A. Soomro. **Mini V-Net: Depth Estimation from Single Indoor-Outdoor Images using Strided-CNN.** In *VISIGRAPP (5: VISAPP)*, pages 205–214, 2020.

[100] R. Zhang, P. Tsai, J. E. Cryer, and M. Shah. **Shape-fromshading: a survey. Pattern Analysis and Machine Intelligence**. *IEEE Transactions on*, **21**(8):690–706, 1999.

[101] S. Suwajanakorn, C. Hernandez, and S. M. Seitz. **Depth from focus with your mobile phone**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3497–3506, 2015.

[102] A. Gupta, M. Hebert, T. Kanade, and D. M. Blei. **Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces**. In *Advances in neural information processing systems*, pages 1288–1296, 2010.

[103] M. Liu, M. Salzmann, and X. He. **Discrete-continuous depth estimation from a single image**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2014.

[104] A. Saxena, M. Sun, and A. Y. Ng. **Make3d: Learning 3d scene structure from a single still image**. *IEEE transactions on pattern analysis and machine intelligence*, **31**(5):824–840, 2009.

[105] B. Liu, S. Gould, and D. Koller. **Single image depth estimation from predicted semantic labels**. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1253–1260. IEEE, 2010.

[106] D. Eigen and R. Fergus. **Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture**. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[107] F. Liu, C. Shen, and G. Lin. **Deep convolutional neural fields for depth estimation from a single image**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170, 2015.

[108] 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*, **38**(10):2024–2039, 2016.

[109] Y. Cao, Z. Wu, and C. Shen. **Estimating depth from monocular images as classification using deep fully convolutional residual networks**. *IEEE Transactions on Circuits and Systems for Video Technology*, **28**(11):3174–3182, 2018.

[110] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab. **Robust optimization for deep regression**. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2830–2838, 2015.

[111] 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*, pages 234–241. Springer, 2015.

[112] B. Xu, N. Wang, T. Chen, and M. Li. **Empirical evaluation of rectified activations in convolutional network**. *arXiv preprint arXiv:1505.00853*, 2015.

[113] F. Liu, G. Lin, and C. Shen. **Discriminative training of deep fully connected continuous CRFs with task-specific loss**. *IEEE Transactions on Image Processing*, **26**(5):2127–2136, 2017.

[114] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun. **A large dataset of object scans**. *arXiv preprint arXiv:1602.02481*, 2016.

[115] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[116] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. **3d-r2n2: A unified approach for single and multi-view 3d object reconstruction**. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[117] U. Khan, A. Yasin, M. Abid, I. Shafi, and S. A. Khan. **A methodological review of 3D reconstruction techniques in tomographic imaging**. *Journal of medical systems*, **42**(10):190, 2018.

[118] A. J. Afifi, J. Magnusson, T. A. Soomro, and O. Hellwich. **Pixel2point: 3D Object Reconstruction From a Single Image Using CNN and Initial Sphere**. *IEEE Access*, **9**:110–121, 2021.

[119] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. **Data-driven 3d voxel patterns for object category recognition**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1903–1911, 2015.

[120] M. Tatarchenko, A. Dosovitskiy, and T. Brox. **Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs**. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[121] J. Zhao, X. Xie, X. Xu, and S. Sun. **Multi-view learning overview: Recent progress and new challenges**. *Information Fusion*, **38**:43–54, 2017.

[122] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. **Geometric deep learning: going beyond euclidean data**. *IEEE Signal Processing Magazine*, **34**(4):18–42, 2017.

[123] 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*, pages 605–613, 2017.

[124] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz. **Single-view modelling of free-form scenes**. *The Journal of Visualization and Computer Animation*, **13**(4):225–235, 2002.

[125] M. Prasad and A. Fitzgibbon. **Single view reconstruction of curved surfaces**. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, **2**, pages 1345–1354. IEEE, 2006.

[126] A. Criminisi, I. Reid, and A. Zisserman. **Single view metrology**. *International Journal of Computer Vision*, **40**(2):123–148, 2000.

[127] E. Delage, H. Lee, and A. Y. Ng. **Automatic single-image 3d reconstructions of indoor manhattan world scenes**. In *Robotics Research*, pages 305–321. Springer, 2007.

[128] T. Nagai, T. Naruse, M. Ikehara, and A. Kurematsu. **Hmm-based surface reconstruction from single images**. In *Proceedings. International Conference on Image Processing*, **2**, pages II–II. IEEE, 2002.

[129] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser. **Simultaneous localization and mapping: A survey of current trends in autonomous driving**. *IEEE Transactions on Intelligent Vehicles*, **2**(3):194–220, 2017.

[130] K. Häming and G. Peters. **The structure-from-motion reconstruction pipeline–a survey with focus on short image sequences**. *Kybernetika*, **46**(5):926–937, 2010.

[131] J. T. Barron and J. Malik. **Shape, illumination, and reflectance from shading**. *IEEE transactions on pattern analysis and machine intelligence*, **37**(8):1670–1687, 2014.

[132] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. **Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling**. In *Advances in neural information processing systems*, pages 82–90, 2016.

[133] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. **Surfnet: Generating 3d shape surfaces using deep residual networks**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6040–6049, 2017.

[134] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. **Pixel2mesh: Generating 3d mesh models from single rgb images**. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.

[135] A. Kurenkov, J. Ji, A. Garg, V. Mehta, J. Gwak, C. Choy, and S. Savarese. **Deformnet: Free-form deformation network for 3d shape reconstruction from a single image**. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 858–866. IEEE, 2018.

[136] Y. Zhang, Z. Liu, T. Liu, B. Peng, and X. Li. **RealPoint3D: An efficient generation network for 3D object reconstruction from a single image**. *IEEE Access*, **7**:57539–57549, 2019.

[137] P. Mandikal, L. NavaneetK., M. Agarwal, and R. V. Babu. **3D-LMNet: Latent Embedding Matching for Accurate and Diverse 3D Point Cloud Reconstruction from a Single Image**. In *BMVC*, 2018.

[138] C.-H. Lin, C. Kong, and S. Lucey. **Learning efficient point cloud generation for dense 3d object reconstruction**. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[139] M.-P. Tran. **3D Contour Closing: A local operator based on Chamfer distance transformation**. 2013.

[140] Y. Rubner, C. Tomasi, and L. J. Guibas. **The earth mover's distance as a metric for image retrieval**. *International journal of computer vision*, **40**(2):99–121, 2000.

[141] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. **Tensorflow: A system for large-scale machine learning**. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

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

[143] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. **Pix3d: Dataset and methods for single-image 3d shape modeling**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2974–2983, 2018.

[144] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. **Distributed representations of words and phrases and their compositionality**. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[145] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. **Generative Adversarial Nets**. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, page 2672–2680, 2014.

[146] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. **Pointnet: Deep learning on point sets for 3d classification and segmentation**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[147] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. **PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space**. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.