Pose Estimation and Object Detection using Deep Convolutional Networks

Jianning Quan

A Thesis
in
The Concordia Institute
for
Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Quality Systems Engineering) at
Concordia University
Montreal, QC, Canada

July 2021

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Jianning Quan

Entitled: Pose Estimation and Object Detection using Deep Convolutional Networks.

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (**Quality Systems Engineering**)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. W. Lucia

_____ Examiner
Dr. M. Ghafouri

_____ Examiner
Dr. W. Lucia

_____ Supervisor
Dr. A. Ben Hamza

Approved by _____
Dr. A. Ben Hamza, Director
Concordia Institute for Information Systems Engineering

_____
Dr. M. Debbabi, Dean
Faculty of Engineering and Computer Science

Date _____

Abstract

Pose Estimation and Object Detection using Deep Convolutional Networks
Jianning Quan, MASc
Concordia University, 2021

Human pose estimation and object detection are fundamental problems in computer vision and autonomous systems with applications ranging from healthcare and sports to surveillance, autonomous driving and traffic monitoring. The task of 3D human pose estimation is to predict the positions of a person's joints, while the goal of object detection is to identify the object category and locate the position using a bounding box for every known object within an image or video. The contributions in this thesis are two-fold. One is to tackle the 3D human pose estimation problem in the graph-theoretic setting. More specifically, we introduce a higher-order graph convolutional framework with initial residual connections for 3D-to-2D pose estimation. The proposed approach is derived from implicit fairing on graphs using a scale-dependent graph Laplacian filtering scheme. Using multi-hop neighborhoods for node feature aggregation, our model is able to capture the long-range dependencies between body joints. Moreover, our approach alleviates the oversmoothing problem caused by repeated graph convolutions, preventing the learned feature representations from converging to similar values thanks in part to residual connections with the first layer of the network. These residual connections are integrated by design in our network architecture, and help ensure that the learned feature representations retain important information from the initial features of the input layer as the network depth increases. Experiments and ablations studies conducted on a standard benchmark demonstrate the effectiveness of our model, achieving superior performance over strong baseline methods for 3D human pose estimation.

The other contribution consists of designing a single-stage object detection model for aerial imagery using a class-balanced loss function in conjunction with a feature pyramid network in an effort to mitigate the data imbalance problem without the need to rely on data augmentation. The key benefit of using the class-balanced focal loss is the ability to adjust the contributions of minority classes to the loss function with the aim to tackle the class imbalance problem, allowing our model to detect different classes evenly. The performance of our proposed object detection model

is demonstrated through extensive experiments on a standard aerial image benchmark, achieving comparable or better object detection results in comparison with competing baselines.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

In this chapter, we present the motivation behind this work, followed by the problem statement, objectives of the study, literature review, an overview of convolutional neural networks, graph convolution networks, and thesis contributions.

1.1 Framework and Motivation

The task of 3D human pose estimation is a fundamental problem in computer vision, robotics, and computer graphics. It refers to the process of predicting the positions of a person's joints (also known as keypoints or landmarks) in images or videos. Application domains of 3D human pose estimation are abundant and range from activity recognition, surveillance and healthcare to games and sports. In the sports industry, for example, 3D pose estimation can help athletes improve their techniques, avoid injury, and train their endurance. Tremendous progress has been made in estimating 3D human pose from images or videos thanks in large part to the rapid development of deep neural network solutions, which have been shown to achieve improved performance over classical approaches that use hand-crafted features. Most existing 3D pose estimation methods use an end-to-end pipeline [1] or a two-stage pipeline [2,3]. The former employs a deep neural network to regress 3D keypoints from images in an end-to-end fashion, whereas the latter is comprised of two main stages, which are usually decoupled from each other. In the first stage of a two-stage pipeline, an off-the-shelf 2D pose detector is typically employed to extract 2D keypoints from an image. The most popular 2D pose detectors are the stacked hourglass network [4] and the cascaded pyramid network [5], which have proven to be more robust for 3D human pose estimation. In

the second stage, the detected 2D keypoints are fed into a regression model (also known as a lifting network) to predict 3D poses. Two-stage approaches for 3D human pose estimation have shown great promise [6–14], outperforming end-to-end models. This better performance is largely attributed to the fact that two-stage methods benefit from intermediate supervision provided in part by robust 2D pose detectors [14]. Martinez *el al.* [7] design a simple fully connected network with residual connections for directly estimating 3D poses from 2D joint detections, outperforming systems trained end-to-end from raw pixels.

Unmanned aerial vehicles or drones are becoming almost ubiquitous, with a growing number of applications in aerial imagery, especially in object detection. Traditional object detection methods usually rely on handcrafted features extracted via shallow network architectures, resulting in poor performance. Recent advances in deep learning have led to the development of various deep neural network architectures for object detection, achieving significant improvements over conventional machine learning approaches. However, one of the most pressing issues facing these deep learning approaches is the class imbalance problem. This issue has underscored the need to develop robust solutions to address the shortcomings of these object detection models. Unmanned aerial vehicles have become increasingly evident in numerous areas, including search and rescue missions, retail delivery and logistics, land surveying, agriculture, oil and gas pipeline inspection, emergency response services, traffic management and monitoring, surveillance of critical assets, pandemic management, and monitoring of infrastructural changes over time [15–18].

1.2 Problem Statement

3D human pose estimation and object detection are fundamental problems in computer vision and autonomous systems. 3D human pose estimation refers to the process of inferring poses, while in object detection we are interested in where the objects are located in images or videos.

1.2.1 3D Human Pose Estimation

Given a 2D human pose as input, we aim to predict the locations of its corresponding 3D joints in a certain coordinate space. More specifically, let $\mathcal{D}_l = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ be a training set of 2D joint positions $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\intercal \in \mathbb{R}^{N\times 2}$ and their associated 3D joint positions $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)^\intercal \in \mathbb{R}^{N\times 3}$. The goal of 3D human pose estimation is to learn the parameters $\boldsymbol{w}$ of a regression model $f : \mathbf{X} \to \mathbf{Y}$ by minimizing the following loss function

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_i), \mathbf{y}_i). \tag{1.1}$$

Since the 3D human pose estimation task is a regression problem, we train the model to minimize the mean squared error as a loss function.

1.2.2 Object Detection

The goal of object detection is to identify the object category and locate the position using a bounding box for every known object within an image or video. In supervised learning tasks, the dataset used for object detection tasks is usually partitioned into three sets: a training set for optimizing the model, a validation set for model selection, and test set for testing the model. The training, validation and test sets are usually selected randomly from the available data. The performance of an object detector is then assessed on test data using evaluation metrics that quantify the discrepancy the ground-truth and predicted labels.

1.3 Objectives

In this thesis, we propose deep learning approaches for 3D human pose estimation and object detection.

- For 3D human pose estimation, we propose a higher-order implicit fairing network for 3D human pose estimation by concatenating feature representations from multi-hop neighborhoods, with the aim to capture long-range dependencies. We follow the two-stage paradigm by employing a state-of-art 2D pose detector, followed by a lifting network for predicting the 3D pose locations from the 2D predictions.
- For object detection, we introduce a class-balanced RetinaNet model for object detection on aerial imagery. We use a class-balanced focal loss function, which has the ability to adjust the contributions of minority classes to the loss function in an effort to mitigate the class imbalance problem and allow our model to detect different classes evenly.

1.4 Literature Review

Both 3D human pose estimation and object detection have received a flurry of research activity over the past few years. Here, we only review the techniques most closely related to ours. Like much previous work discussed next, we approach the problem of 3D human pose estimation using a two-stage pipeline. We also provide an overview of object detection and convolutional neural networks. The most basic task of convolutional neural networks is image classification, in which

we assign a class label to an input image. In object detection, we are not only interested in what objects are in the input image, but we are also interested in where these objects are located.

1.4.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a deep architecture inspired by the way humans process visual information [19]. It makes use of feedforward artificial neural networks in which individual neurons are tiled in such a way that they respond to overlapping regions in the visual field. CNNs are comprised of multiple layers that can be categorized into three types: convolutional, pooling and fully-connected, as shown in Figure 1.1. A convolutional layer consists of a rectangular grid of neurons, and applies a set of filters that process small local parts of the input where these filters are replicated along the whole input space. Each neuron takes inputs from a rectangular section of the previous layer; the weights for this rectangular section are the same for each neuron in the convolutional layer. Thus, the convolutional layer is just an image convolution of the previous layer, where the weights specify the convolution filter. A pooling layer takes small rectangular blocks from the convolutional layer and subsamples it with with average or max pooling to produce a single output from that block. This adds translation invariance and tolerance to minor differences of positions of objects parts. Higher layers use more broad filters that work on lower resolution inputs to process more complex parts of the input. Similar to a feedforward neural network, a fully connected layer takes all neurons in the previous layer and connects them to each of its neurons. CNNs can be trained using standard backpropagation. For classification tasks, an output layer is usually added after the fully connected layer.

1.4.2 Graph Convolution Networks

GCNs have recently become the de facto model for learning representations on graphs, achieving state-of-the-art performance in various application domains, including 3D pose estimation [20–22]. However, GCNs are prone to oversmooting as the network depth increases, and also fail to capture important dependencies between distant nodes. To circumvent these limitations, a plethora of GCN variants have been proposed, including jumping knowledge networks (JK-Nets) [23], approximate personalized propagation of neural predictions (APPNP) [24], graph convolutional networks with initial residual connection and identity mapping (GCNII) [25], and higher-order graph convolutional architectures via sparsified neighborhood mixing (MixHop) [26]. The latter learns neighborhood mixing relationships by repeatedly mixing feature representations of neighbors at various distances through powers of the graph adjacency matrix, while requiring no additional memory or computational complexity.

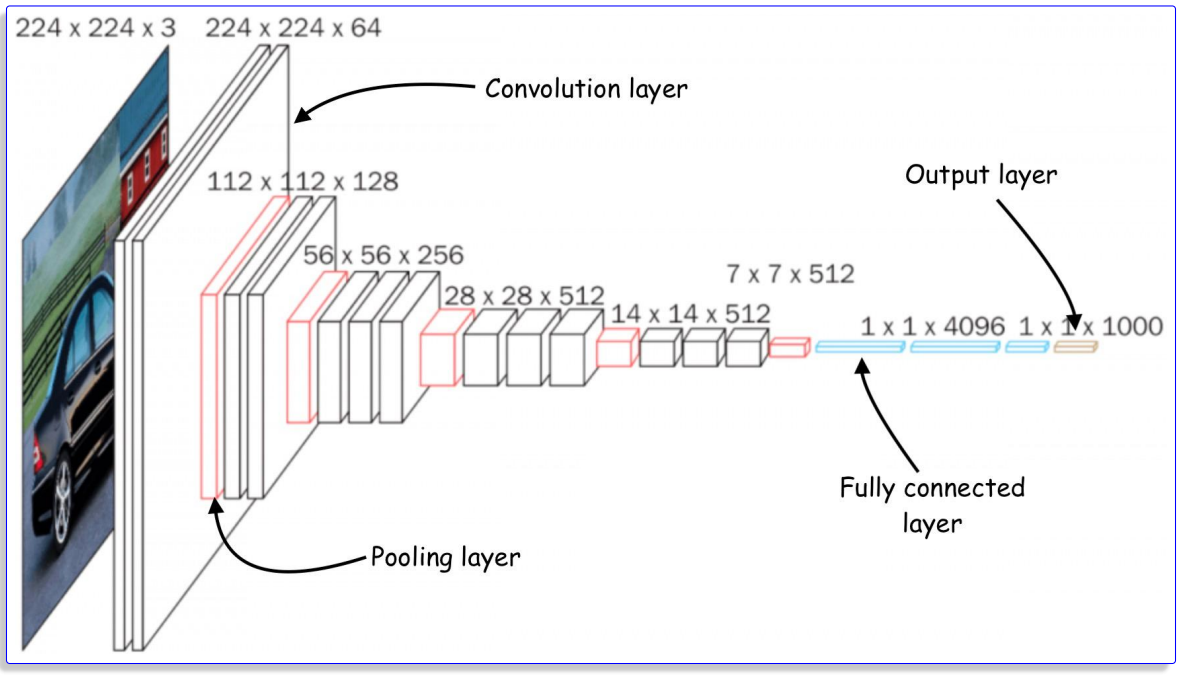

Figure 1.1: Architecture of a convolutional neural network.

1.4.3 3D Human Pose Estimation

The aim of 3D human pose estimation is to predict 3D joints locations from images or videos. Most approaches for 3D human pose estimation can generally be classified into two main categories, namely single-stage and two-stage models, with the former using an end-to-end pipeline to predict 3D poses from images and the latter using a two-stage pipeline, in which 2D joint locations are first extracted using a 2D pose detector and then a lifting network is employed to regress 3D poses from 2D detections. Our proposed approach falls under the category of two-stage models [6–14, 20–22]. In [10], a sequence-to-sequence network with shortcut connections is introduced by exploiting the temporal information across a sequence of 2D joint locations in order to estimate a sequence of 3D poses. Pavlakos *et al.* [11] use ordinal depth relations oh human joints to train convolutional neural networks for 3D human pose estimation in an effort to mitigate the requirement for accurate 3D ground truth. Sharma *et al.* [12] propose a deep conditional variational autoencoder based model for 3D-to-2D pose estimation by synthesizing diverse 3D pose samples conditioned on the estimated 2D pose. More recently, much attention has also been focused on graph convolutions networks for 3D human pose estimation. Zhao *et al.* [20] introduce a semantic graph convolutional network to capture semantic information encoded in a given graph. Cai *et al.* [21] propose a GCN-based model to exploit the spatial configurations and temporal consistencies for 3D pose estimation by treating a sequence of skeletons as a spatial-temporal graph.

Compared to [21], we do not use temporal information in our proposed framework. Zou *et al.* [22] design a high-order GCN for 3D pose estimation based on the MixHop model in a bid to capture long-range dependencies between distant body joints. Moreover, the network architecture of the high-order GCN model uses a residual block repeated several times similar to the network design of Martinez *et al.* [7]. In addition, it inherits the oversmoothing issue of GCNs, where repeated graph convolutions eventually make learned node embeddings indistinguishable; thereby, resulting in performance drop. Our approach, however, has integrated residual connections by design, and hence is able to alleviate the oversmoothing problem.

1.4.4 Object Detection

Object detection is an important research area in computer vision, thanks in large part to its capability of generating valuable information for semantic understanding of images or videos. The main purpose of object detection is not only to discover the location of all objects in a image, but also to classify each object's category. Object detection can be applied to a variety of tasks, including face detection, pedestrian detection, and human joints detection. The pipeline of traditional object detection methods is typically comprised of three stages [27]: informative region selection, feature extraction, and classification. The goal of informative selection is to find all the possible positions of the different objects that appear in an image. An original approach is to scan the whole image by building a multi-scale sliding window. However, this method is computationally expensive due to the large amount of candidate windows. The feature extraction stage aims at extracting useful visual features of objects. Extracting quality features is the premise of success in the next classification stage. While some traditional feature extraction algorithms such as the histogram of oriented gradients [28] and Haar [29, 30] have shown promising results in detecting some of the objects, they cannot, however, be used as general feature extractors. The main purpose of the classification stage is to categorize the objects according to the features extracted in the second stage. Algorithms like support vector machines [31] and deformable part-based models [32] can be used, but they are not effective at learning complex features compared to deep neural networks [33]. Deep learning models for object detection bear much resemblance to image classification models and often use convolutional layers to detect visual features. In fact, most object detection models employ an image classification CNN and reuse it for object detection. CNN-based models for object detection can be grouped into two main categories: two-stage and one-stage detectors. The former consist of two main stages. In the first stage, the model proposes a set of regions of interest by the select search algorithm or a regional proposal network, while in the second stage a classifier only processes the region candidates. Models such as Region-based Convolutional Neu-

ral Network (R-CNN) [34], Fast R-CNN [35], and Faster R-CNN [36] fall under the category of two-stage object detectors whose network architectures[1] are shown in Figure 1.2. R-CNN uses a selective search algorithm to identify a certain number of bounding-box object region candidates (also known as regions of interest), and then extracts features from each region separately using a CNN model. The region selector generates about 2,000 regions of interest for each image, a process that can be computationally expensive; thereby making R-CNN unsuitable for real-time object detection, particularly for UAVs. To improve the speed performance of R-CNN, Girshick *et al.* [35] introduce Fast R-CNN, a neural network architecture that integrates feature extraction and region selection into a single machine learning model. It takes an image and a set of regions of interest (RoIs) as input, and returns a list of bounding boxes and classes of the objects detected in the image. A key novelty in Fast R-CNN is the RoI pooling layer, which takes the feature maps and RoIs for an image and yields the corresponding features for each region. Unlike R-CNN which processes each region separately, Fast R-CNN extracts features for all the RoIs in the image in a single pass, resulting in a significant boost in speed. However, Fast R-CNN is still unsuitable for real-time object detection, as it requires the regions of the image to be extracted and provided as input to the model. To make region proposal more efficient, Ren *et al.* [36] propose a Faster R-CNN network to speed-up the process of region proposal by eliminating the selective search algorithm and integrating the region extraction mechanism into the object detection network. Faster R-CNN takes an image as input and returns a list of object classes and their corresponding bounding boxes. It uses a region proposal network, which takes the feature maps produced by a convolutional neural network and proposes a set of bounding boxes where objects might be located. In other words, the idea is to construct a single, unified model composed of region proposal network and Fast R-CNN with shared convolutional feature layers. While two-stage detectors have shown improved accuracy in object detection, they are, however, computationally expensive in both training and testing time, and can hardly meet the requirements for real-time detection in drones.

On the other hand, one-stage detectors such as YOLO [37], SSD [38], and RetinaNet [39] typically skip the region proposal step and only make predictions over a limited number of bounding boxes, thereby they are faster and simpler without a significant drop in performance. The YOLO model [37] divides an image into several fixed-size grids and predicts the bounding boxes confidence as well as probabilities of multiple classes. Also, some improved methods like YOLO9000 [40] and YOLOv3 [41] adopt dimension clusters when predicting bounding boxes and multi-scale training, as well as batch normalization on all convolutional layers in order to improve the performance of the model. However, one potential disadvantage of YOLO is that it can hardly

[1] https://lilianweng.github.io/

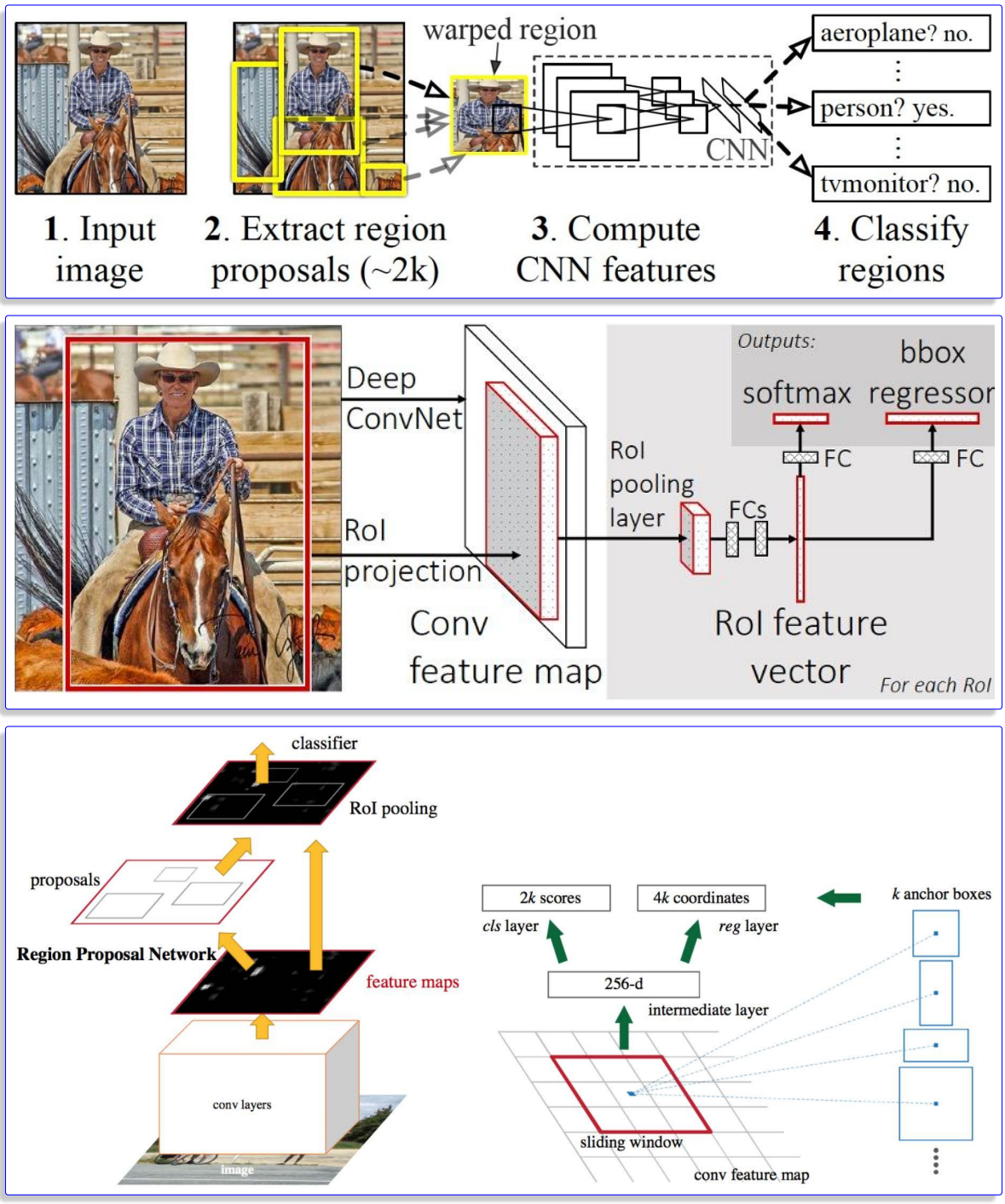

Figure 1.2: Network architectures of two-stage object detectors: R-CNN (top); Fast R-CNN (middle); and Faster R-CNN (bottom).

deal with small objects that appear in groups, and also objects in unusual aspect ratios or configurations, largely because of the limitations of spatial constraints in the YOLO model. To address these problems, Liu *et al.* [38] propose a Single Shot MultiBox Detector (SSD), which adopts de-

fault boxes (similar to the anchor boxes in Faster R-CNN) on several feature maps with different resolutions to let the model discretize the output bounding box shapes. A key disadvantage of SSD is that it is not very effective at detecting small objects because it does not have the feature resampling step as in Faster R-CNN, but this can be relieved by applying data augmentation or changing the network structure [38]. However, according to the characteristics of aerial images and hardware components of drones, the above models are not the most suitable for aerial traffic detection. In order to meet the requirements of object detection on aerial images with drones, we adopt a simple, yet efficient network as a feature extractor.

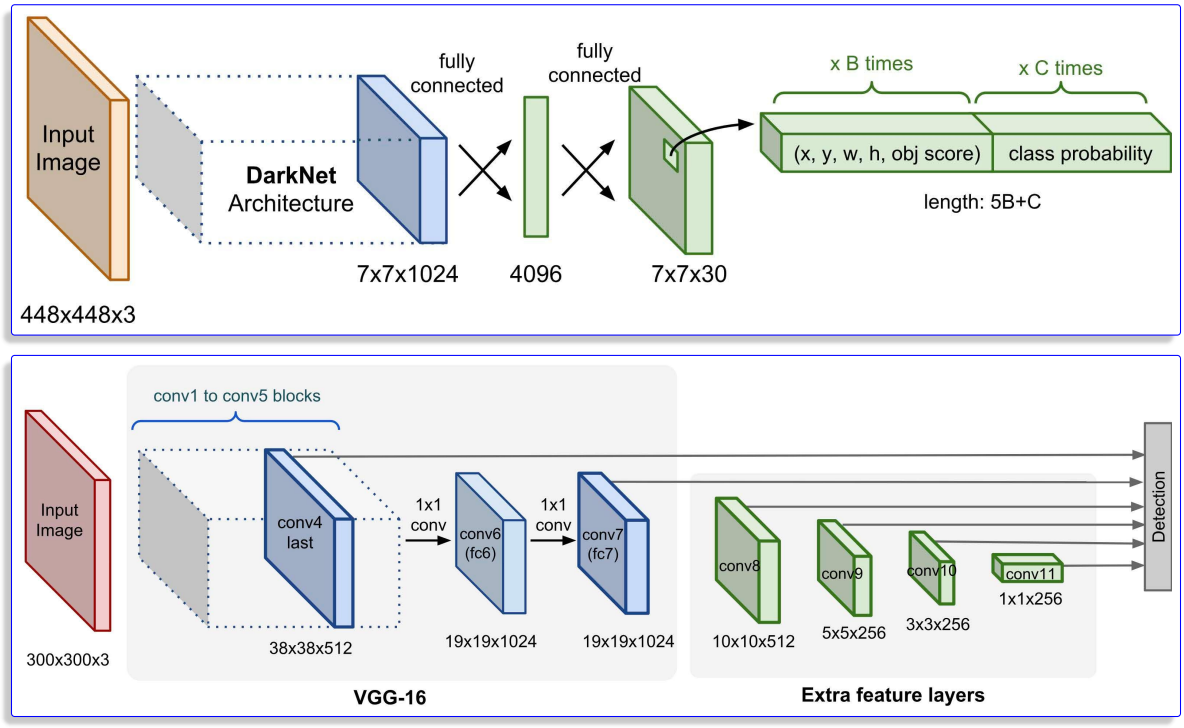

Figure 1.3: Network architectures of one-stage objects detectors: YOLO (top); and SSD (bottom).

1.5 Overview and Contributions

The organization of this thesis is as follows:

- Chapter 1 begins with the motivations and goals for this research, followed by the problem statement, the objective of this study, a literature review with a brief discussion of some algorithms relevant to deep learning in 3D human pose estimation and object detection.
- In Chapter 2, we derive an implicit fairing network (IF-Net) with initial residual connection by iteratively solving the implicit fairing equation on graphs via Jacobi method and propose

a higher-order implicit fairing network (HOIF-Net) for 3D human pose estimation by concatenating feature representations from multi-hop neighborhoods, with the aim to capture long-range dependencies. We demonstrate through extensive experiments and ablation studies that our proposed model achieves state-of-the-art performance in comparison with strong baselines.

- In Chapter 3, we adopt an efficient one-stage object detectors (RetinaNet) to extract features on traffic detection using UAVs and apply class-balanced focal loss to build our Class Balanced RetinaNet(CB-RetinaNet) to solve the class imbalance problems which existing in object detection tasks instead of doing data augmentations on original dataset. We analyze the complexity of the proposed model and train it on a aerial image dataset: VisDrone2019-DET to verify the performance of our model and demonstrate through extensive experiments and ablation study that our model can get a higher precision which lead to state-of-the-art performance across comparing the results with several baseline methods..

- Chapter 4 presents a summary of the contributions of this thesis, limitations, and outlines several directions for future research in this area of study.

CHAPTER 2

Higher-Order Implicit Fairing Networks for 3D Human Pose Estimation

In this chapter, we introduce a higher-order graph convolutional framework with initial residual connections for 3D-to-2D pose estimation. The proposed approach is derived from implicit fairing on graphs using a scale-dependent graph Laplacian filtering scheme. Using multi-hop neighborhoods for node feature aggregation, our model is able to capture the long-range dependencies between body joints. Moreover, our approach alleviates the oversmoothing problem caused by repeated graph convolutions, preventing the learned feature representations from converging to similar values thanks in part to residual connections with the first layer of the network. These residual connections are integrated by design in our network architecture, and help ensure that the learned feature representations retain important information from the initial features of the input layer as the network depth increases. Experiments and ablations studies conducted on a standard benchmark demonstrate the effectiveness of our model, achieving superior performance over strong baseline methods for 3D human pose estimation.

2.1 Introduction

In recent years, there has been a surge of interest in the adoption of graph convolution networks (GCNs) for 3D pose estimation [20–22], achieving state-of-the-art performance. Much of this interest stems from the fact that a 2D human skeleton can naturally be represented as a graph whose nodes are body joints and edges are connections between neighboring joints. A GCN is a

popular semi-supervised graph-based deep learning framework [42], which uses an efficient layer-wise propagation rule based on a first-order approximation of spectral graph convolutions. The feature vector of each graph node in GCN is updated by essentially applying a weighted sum of the features of its immediate neighboring (i.e. one-hop) nodes. In [43], a simplified GCN architecture is presented by removing the nonlinear activation functions between each network layer and only retaining the final softmax, while still having the same increased receptive field of a multi-layer GCN. Zhao *et al.* [20] propose SemGCN, a semantic graph convolutional network, which learns to capture semantic information encoded in a given graph (i.e. local and global relations between nodes), yielding improved performance in 3D human pose estimation while using a much smaller number of parameters.

While GCN is powerful for learning on graph-structured data, it suffers, however, from the oversmoothing problem [44], where the learned node representations become indistinguishable due to repeated graph convolutions as the network depth increases. Several attempts have been made toward remedying this issue of oversmoothing [23–25, 45]. Xu *et al.* [23] propose jumping knowledge networks, which employ dense skip connections to connect each layer of the network with the last layer to preserve the locality of node representations in order circumvent oversmoothing. Klicpera *et al.* [24] present an approximate personalized propagation of neural predictions (APPNP) model derived from personalized PageRank. The propagation procedure of the APPNP approach is based on a power iterative scheme, which uses a teleport probability as a weighting balance hyperparameter to help avoid oversmoothing and leverage the information from a large neighborhood in order to provide the model with more information. In [45], a normalization layer is proposed to help avoid oversmoothing by preventing learned representations of distant nodes from becoming indistinguishable. This normalization layer is performed on intermediate layers during training, and the aim is to apply smoothing over nodes within the same cluster while avoiding smoothing over nodes from different clusters. Also, Chen *et al.* [25] introduce GCNII, a graph convolutional network with initial residual connection and identity mapping, to tackle the oversmoothing problem. At each network layer of GCNII, the initial residual builds a skip connection from the initial feature representation, while the identity mapping uses a weighted sum of an identity matrix and the learnable weight matrix.

Another issue with GCN is that its aggregation scheme uses one-hop neighbors, and hence lacks the ability to capture long-range dependencies. This issue can be mitigated by skipping connections during feature aggregations using, for example, the jumping knowledge networks [23] or by concatenating feature representations of multi-hop neighbors using the MixHop model, which leverages a graph convolutional layer that mixes powers of the adjacency matrix [26]. Motivated

by MixHop, Zou *et al.* [22] propose a high-order GCN for 3D pose estimation, with the goal of capturing long-range dependencies between body joints, and hence reducing the uncertainty caused by occlusion or depth ambiguity.

To address the above issues, we introduce a higher-order graph convolutional framework for 3D pose estimation via implicit fairing on graphs. Building on previous research, we follow the two-stage paradigm by employing a state-of-art 2D pose detector, followed by a lifting network for predicting the 3D pose locations from the 2D predictions. The aggregation scheme of the proposed approach leverages residual connections to help alleviate the oversmoothing problem, and uses multi-hop neighborhoods to capture long-range dependencies between body joints. The main contributions of this work can be summarized as follows:

- We derive an implicit fairing network (IF-Net) with initial residual connection by iteratively solving the implicit fairing equation on graphs via Jacobi method.
- We propose a higher-order implicit fairing network (HOIF-Net) for 3D human pose estimation by concatenating feature representations from multi-hop neighborhoods, with the aim to capture long-range dependencies.
- We demonstrate through extensive experiments and ablation studies that our proposed model achieves state-of-the-art performance in comparison with strong baselines.

The rest of this chapter is structured as follows. In Section 2.2, we summarize the basic notation and concepts, and then provide a problem formulation. In Section 2.3, we describe the main building blocks of the proposed framework, and provide a generalization to higher-order settings. In Section 2.4, we present empirical results comparing our model with state-of-the-art approaches for 3D pose estimation on a large-scale standard benchmark.

2.2 Preliminaries and Problem Statement

We introduce our notation and present a brief background on graph convolutional networks, and Jacobi's method for solving systems of linear equations, followed by our problem formulation of 3D human pose estimation.

2.2.1 Basic Notions.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of N nodes (e.g., body joints) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges (e.g., connections between two body joints). Let $\mathbf{A}$ be an $N \times N$

adjacency matrix whose (i, j)-th entry is equal to the weight of the edge between neighboring nodes i and j, and 0 otherwise. We denote by $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ the adjacency matrix with self-added loops, where $\mathbf{I}$ is the identity matrix. We also denote by $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^\intercal$ an $N \times F$ feature matrix of node attributes, where $\mathbf{x}_i$ is an F-dimensional row vector for node i. This real-valued feature vector is often referred to as a graph signal, which assigns a value to each node in the graph. We define the normalized Laplacian matrix as follows

$$\mathbf{L} = \mathbf{I} - \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{2.1}$$

where $\tilde{\mathbf{D}} = \mathrm{diag}(\tilde{\mathbf{A}}\mathbf{1})$ is the diagonal degree matrix, and $\mathbf{1}$ is an N-dimensional vector of all ones. The Laplacian matrix admits an eigendecomposition given by $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$, where $\mathbf{U}$ is an orthonormal matrix whose columns constitute an orthonormal basis of eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix comprised of the corresponding eigenvalues.

2.2.2 Graph Convolutional Networks (GCNs).

An L-layer GCN learns a new feature representation for each node such that nodes with the same labels have similar features. Given an input feature matrix $\mathbf{H}^{(\ell)} \in \mathbb{R}^{N \times F_\ell}$ of the ℓ-th layer with F_ℓ feature maps, the output feature matrix $\mathbf{H}^{(\ell+1)}$ of GCN is obtained by applying the following layer-wise propagation rule:

$$\mathbf{H}^{(\ell+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}), \quad \ell = 0, \ldots, L-1, \tag{2.2}$$

which is basically a node embedding transformation that projects the input $\mathbf{H}^{(\ell)}$ into a trainable weight matrix $\mathbf{W}^{(\ell)} \in \mathbb{R}^{F_\ell \times F_{\ell+1}}$ with $F_{\ell+1}$ feature maps, followed by a point-wise activation function $\sigma(\cdot)$ such as $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. The input of the first layer is the initial feature matrix $\mathbf{H}^{(0)} = \mathbf{X}$.

2.2.3 Jacobi Method.

The Jacobi method [46] is an iterative approach of solving a matrix equation $\mathbf{M}\mathbf{x} = \mathbf{b}$, where the square matrix $\mathbf{M}$ has no zeros along its main diagonal, by first decomposing $\mathbf{M}$ into a diagonal component and an off-diagonal component, i.e.

$$\mathbf{M} = \mathrm{diag}(\mathbf{M}) + \mathrm{off}(\mathbf{M}). \tag{2.3}$$

Then, the solution of the matrix equation $\mathbf{M}\mathbf{x} = \mathbf{b}$ is obtained iteratively as follows:

$$\mathbf{x}^{(t+1)} = \mathrm{diag}(\mathbf{M})^{-1}(\mathbf{b} - \mathrm{off}(\mathbf{M})\mathbf{x}^{(t)}), \tag{2.4}$$

where $\mathbf{x}^{(t)}$ and $\mathbf{x}^{(t+1)}$ are the t-th and $(t+1)$-th iterations of $\mathbf{x}$, respectively.

2.2.4 Problem Statement.

Since the 3D human pose estimation task is a regression problem, we train the model to minimize the mean squared error as a loss function. Figure 2.1 shows an example of a 2D human pose graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of $|\mathcal{V}| = 17$ nodes (joints) and $|\mathcal{E}| = 16$ edges. The skeletal graph is composed of a set of 17 joints (keypoints) across the body such as the head, neck, upper torso, left and right shoulders, thorax, and pelvis. The latter is usually used as a root joint.

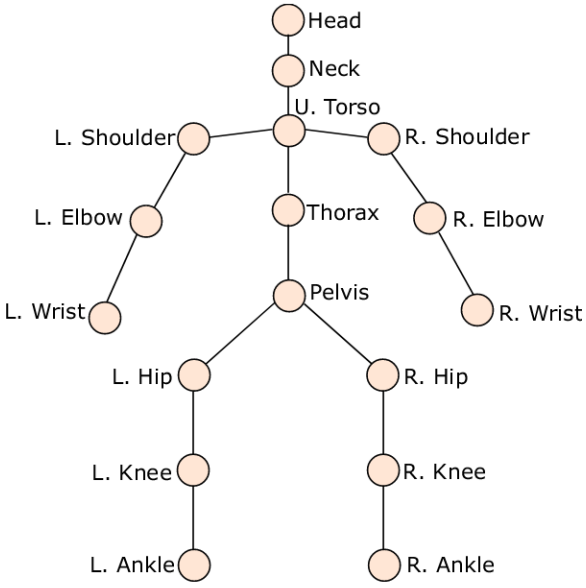

Figure 2.1: Example of a 2D human pose skeletal graph.

2.3 Proposed Method

In this section, we present the main components of the proposed higher-order implicit fairing network for 3D human pose estimation. Typically, in a 2D-to-3D pose estimation problem, a 2D pose detector is usually performed to detect 2D keypoints, which can be represented as a graph, where each node is a keypoint and each edge is a link connecting two neighboring joints. Then, these 2D detections are fed into our graph-based regression model to infer the 3D keypoint predictions.

2.3.1 Implicit Fairing on Graphs

The goal of spectral graph filtering is to use polynomial or rational polynomial filters defined as functions of the graph Laplacian (or equivalently its eigenvalues) in an effort to attenuate high-frequency noise corrupting the graph signal. These functions are usually referred to as frequency

responses or transfer functions. While polynomial filters have finite impulse responses, their rational counterparts have infinite impulse responses. Applying a spectral graph filter with transfer function h on the graph signal $\mathbf{X}$ (e.g., data matrix of 2D joint poses) yields

$$\mathbf{H} = h(\mathbf{L})\mathbf{X} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^{\mathsf{T}}\mathbf{X}, \tag{2.5}$$

where $\mathbf{H}$ is the filtered graph signal. However, this filtering process requires the computation of the eigenvalues and corresponding eigenvectors of the Laplacian matrix. Such computation is prohibitively expensive, particularly for large graphs. To circumvent this issue, spectral graph filters are usually approximated using Chebyshev polynomials [47–49] or rational polynomials [50–52].

Implicit fairing on graphs refers to the process of designing and computing smooth graph signals on a graph in order to filter out undesirable high-frequency noise while retaining the graph geometric features as much as possible. The implicit fairing method, which uses implicit integration of a diffusion process for graph filtering, has shown to allow for both efficiency and stability [53]. The implicit fairing filter is an infinite impulse response filter whose transfer function is given by $h_s(\lambda) = 1/(1 + s\lambda)$, where s is a positive parameter. Substituting h with h_s in Eq. (2.5), we obtain

$$\mathbf{H} = (\mathbf{I} + s\mathbf{L})^{-1}\mathbf{X}, \tag{2.6}$$

where $\mathbf{I} + s\mathbf{L}$ is a symmetric positive definite matrix (all its eigenvalue are positive), and hence admits an inverse. Therefore, performing graph filtering with implicit fairing is equivalent to solving the following sparse linear system:

$$(\mathbf{I} + s\mathbf{L})\mathbf{H} = \mathbf{X}. \tag{2.7}$$

2.3.2 Iterative Solution

The implicit fairing equation (2.7) can be solved iteratively using Jacobi's method, which uses matrix splitting. We can split the matrix $\mathbf{I} + s\mathbf{L}$ into the sum of a diagonal matrix and an off-diagonal matrix as follows:

$$\mathbf{I} + s\mathbf{L} = \mathrm{diag}(\mathbf{I} + s\mathbf{L}) + \mathrm{off}(\mathbf{I} + s\mathbf{L}), \tag{2.8}$$

where

$$\mathrm{diag}(\mathbf{I} + s\mathbf{L}) = (1 + s)\mathbf{I} \text{ and } \mathrm{off}(\mathbf{I} + s\mathbf{L}) = -s\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}.$$

Hence, the iterative solution of the implicit fairing equation is given by

$$\begin{aligned}\mathbf{H}^{(t+1)} &= -(\mathrm{diag}(\mathbf{I} + s\mathbf{L}))^{-1}\,\mathrm{off}(\mathbf{I} + s\mathbf{L})\mathbf{H}^{(t)} \\ &\quad + (\mathrm{diag}(\mathbf{I} + s\mathbf{L}))^{-1}\mathbf{X} \\ &= (s/(1 + s))\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(t)} + (1/(1 + s))\mathbf{X},\end{aligned}$$

which can be rewritten as

$$\mathbf{H}^{(t+1)} = (1-\alpha)\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(t)} + \alpha\mathbf{X}, \tag{2.9}$$

where the hyperparameter $\alpha = 1/(1+s) \in (0,1)$, and $\mathbf{H}^{(t)}$ is the t-th iteration of $\mathbf{H}$.

Relation to APPNP. It is worth pointing out that each power iteration (random walk/propagation) step of the approximate personalized propagation of neural predictions (APPNP) [24] is given by

$$\mathbf{H}^{(t+1)} = (1-\alpha)\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(t)} + \alpha\mathbf{H}^{(0)}, \tag{2.10}$$

where $\mathbf{H}^{(0)} = \text{MLP}(\mathbf{X})$ is obtained by applying a multilayer perception (MLP) to the initial feature matrix, and the hyperparameter $\alpha \in (0,1]$ is referred to as the teleport probability. Hence, instead of using PageRank, the APPNP model can be alternatively derived by iteratively solving the implicit fairing equation using Jacobi's method.

2.3.3 Implicit Fairing Network

Inspired by the Jacobi iterative solution (2.9) of the implicit fairing equation, we propose a multilayer implicit fairing network (IF-Net) with the following layer-wise propagation rule:

$$\mathbf{H}^{(\ell+1)} = \sigma(((1-\alpha)\mathbf{S}\mathbf{H}^{(\ell)} + \alpha\mathbf{X})\tilde{\mathbf{W}}^{(\ell)}), \tag{2.11}$$

where $\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is the normalized adjacency matrix with self-added loops, and $\tilde{\mathbf{W}}^{(\ell)} = \beta_\ell\mathbf{W}^{(\ell)}$ is a scaled, learnable weight matrix with a layer-dependent scale factor defined as $\beta_\ell = \log(\beta/(1+\ell)+1)$, which ensures that the decay of the weight matrix increases in tandem with the network depth [25]. The hyperparameters α and β are selected via grid search with cross-validation. The input feature matrix $\mathbf{H}^{(\ell)}$ of the ℓ-th layer has F_ℓ feature maps, and $\sigma(\cdot)$ is an element-wise activation function.

Note that in addition to performing graph convolution, which essentially averages the features of the immediate (i.e. first-order or 1-hop) neighbors of nodes, the layer-wise propagation rule of IF-Net also applies a residual connection that reuses the initial node features.

Relation to GCNII. The layer-wise propagation rule of IF-Net can be regarded as GCNII [25] without identity mapping. However, a major limitation of GCNII is that it requires the learnable weight matrices to be square in order to add an identity matrix, and consequently GCNII cannot be used as a node embedding transformation. By contrast, our model naturally embeds the input of each layer into a trainable weight matrix; thereby mapping 2D poses into 3D poses.

2.3.4 Higher-Order Implicit Fairing Network

Using the feature diffusion rule of GCN is tantamount to applying a weighted sum of the features of neighboring nodes normalized by their degrees, which essentially performs Laplacian smoothing on the graph, and hence leads to oversmoothing. Also, the aggregation scheme of GCN uses 1-hop neighbors, and hence lacks the ability to capture long-range dependencies. To circumvent these issues, we define a higher-order implicit fairing network (HOIF-Net) with the following layer-wise propagation rule:

$$\mathbf{H}^{(\ell+1)} = \sigma(\mathop{\Big\|}_{k=1}^{K} \tilde{\mathbf{H}}_k^{(\ell)} \tilde{\mathbf{W}}_k^{(\ell)}), \tag{2.12}$$

where

$$\tilde{\mathbf{H}}_k^{(\ell)} = (1-\alpha)\mathbf{S}^k\mathbf{H}^{(\ell)} + \alpha\mathbf{X}, \tag{2.13}$$

and $\mathbf{S}^k$ is the k-th power of the normalized adjacency matrix with self-added loops. Each (i,j)-th entry of $\mathbf{S}^k$ counts the number of walks of length k between nodes i and j. For example, the (i,j)-th entry of $\mathbf{S}^2$ gives the number of common neighbors of nodes i and j. The learnable weight matrix $\tilde{\mathbf{W}}_k$ is associated to the k-hop neighbors, and $\|$ denotes concatenation. For each k-hop neighborhood, the node feature representation $\tilde{\mathbf{H}}_k^{(\ell)}$ given by Eq. (2.13) is a weighted sum of the transformed feature matrix $\mathbf{S}^k\mathbf{H}^{(\ell)}$ for the ℓ-layer and the initial feature matrix $\mathbf{X}$. Intuitively, the transformation $\mathbf{S}^k\mathbf{H}^{(\ell)}$ yields a smooth hidden representation, and hence encourages similar predictions among k-hop neighboring nodes. In other words, feature information from nodes that are k-hops away in the graph is captured. In addition, $\mathbf{S}^k\mathbf{H}^{(\ell)}$ can be efficiently computed by exploit fast sparse-dense matrix multiplication since $= \mathbf{S}$ and k is typically small in practice.The weighting factor α represents the weight assigned to the initial feature information that needs to be carried over, as the number of layers increase. The HOIF-Net model places a greater weight and significance on the transformed feature representations, meaning that these representations have more influence on $\tilde{\mathbf{H}}_k^{(\ell)}$. In general, values of α in the interval $0.1 \leq \alpha \leq 0.2$ work well in practice. A good rule of thumb is to use smaller values of α, as larger values would discount discriminative learned representations at higher layers of the network. Figure 2.2 shows an illustration of the layer-wise propagation rule of HOIF-Net when $K = 3$.

Note that HOIF-Net uses residual connections between the initial feature matrix and each hidden layer. Residual connections not only allow the model to carry over information from the initial node attributes, but also help facilitate training of multi-layer networks. While HOIF-Net mixes feature representations of multi-hop neighbors in much the same way as MixHop [26] and highr-order GCN [22], our model differs in the sense that residual connections are integrated by design in the network architecture, alleviating oversmoothing without adding special layers.

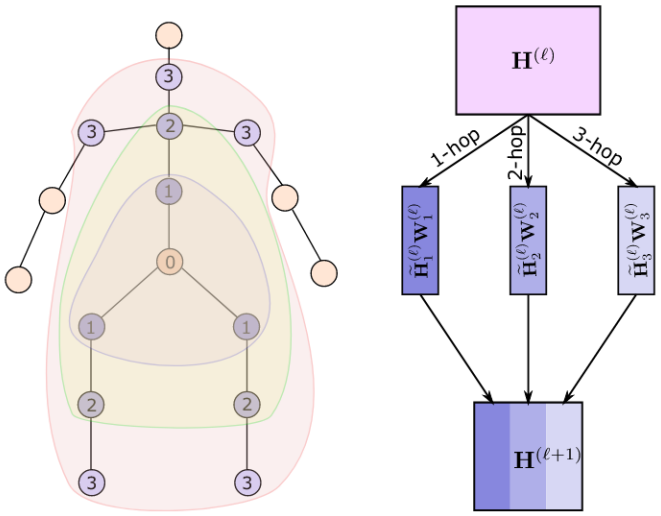

Figure 2.2: Illustration of HOIF-Net feature concatenation for $K = 3$.

2.3.5 Model Architecture

The architecture of our proposed model for 3D human pose estimation is illustrated in Figure 2.3. The input consists of 2D keypoints generated via a 2D pose detector. We use 10 higher-order convolutional layers, each of which is followed by batch normalization and ReLU activation function, except the last convolutional layer. Batch normalization is a layer that has the ability to accelerate training of deep neural networks. It is used to normalize the output of the previous layers by maintaining zero mean and unit variance.

The generated output of the proposed model consists of predicted 3D pose coordinates. We use higher-order graph convolutional layers defined by the layer-wise propagation rule of HOIF-Net to capture long-range structural information between body joints.

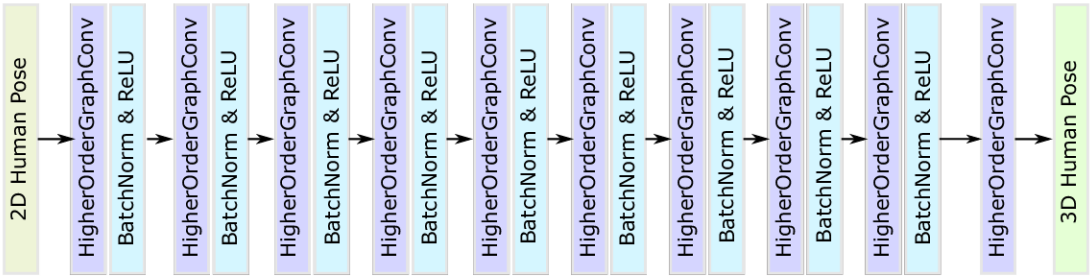

Figure 2.3: Overview of the proposed network architecture for 3D pose estimation. Our model takes 2D pose coordinates (17 joints) as input and generates 3D pose predictions (17 joints) as output. We use ten higher-order graph convolutional layers.

2.3.6 Model Prediction

The output of the last higher-order graph convolutional layer of HOIF-Net contains the final output node embeddings, which are given by

$$\hat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_N)^\intercal \in \mathbb{R}^{N \times 3}, \tag{2.14}$$

where $\hat{\mathbf{y}}_i$ is a three-dimensional raw vector of predicted 3D pose coordinates.

2.3.7 Model Training

The parameters (i.e. weight matrices for different layers) of the proposed HOIF-Net model for 3D human pose estimation are learned by minimizing the following loss function

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2, \tag{2.15}$$

which is the mean squared error between the 3D ground truth poses $\mathbf{y}_i$ and estimated 3D joint poses $\hat{\mathbf{y}}_i$ over a training set consisting of N human poses.

2.4 Experiments

In this section, we perform experiments to assess the performance of our proposed model on 3D human pose estimation in comparison with state-of-the-art methods.

2.4.1 Experimental Setup

Dataset. We perform quantitative and qualitative evaluations on the standard Human 3.6M benchmark [54], which is a large-scale dataset for 3D human pose estimation. The dataset contains 3.6 million 3D human poses for 11 professional actors (6 males and 5 females) and corresponding images captured by a high-speed motion capture (MoCap) system with four different cameras. Each actors performs 15 actions (scenarios), including directions, discussion, eating, greeting, talking on the phone and so on, as shown in Figure 2.4. For each corresponding image of an actor, both 2D and 3D ground truth positions are available in the dataset. For data preprocessing, we apply standard normalization to the 2D and 3D poses before feeding the data to the model in line with previous work [7, 22].

Evaluation Protocols and Metrics. For the Human 3.6M benchmark, there are two commonly used evaluation protocols, referred to as Protocol #1 and Protocol #2. Both protocols use 5 subjects

(S1, S5, S6, S7, S8) for training and 2 subjects (S9, S11) for testing. Under Protocol #1, we report the mean per joint position error (MPJPE), which computes the average Euclidean distance between the predicted 3D joint positions and ground truth after the alignment of the root joint (central hip). Under Protocol #2, we report the Procrustes-aligned mean per joint position error (PA-MPJPE), where MPJPE is computed after rigid alignment of the prediction with respect to the ground truth. Both error metrics are measured in millimeters, and lower values indicate better performance.

Implementation Details. We implement our HOIF-Net model in Pytorch, and all experiments are conducted on desktop computer equipped with an NVIDIA RTX 2070 SUPER GPU. We train our model for 50 epochs using the Adam optimizer with a learning rate of 0.001. We set the decay factor to 0.96 per 100,000 steps, and the batch size to 64. We also set the hyperparameters α and β to 0.2 and 0.5, respectively, via grid search with cross-validation on the training set. To extract 2D keypoints from input images and following common practices in previous work [14, 22], we employ the cascaded pyramid network (CPN) [4], which uses bounding boxes obtained by Mask R-CNN [55]. The CPN and Mask R-CNN detectors are pretrained on the COCO dataset [56], and fine-tuned on 2D projections of the Human3.6M dataset since the keypoints in both datasets are different. For K-hop feature concatenation, we set the value of K to 3, as illustrated in Figure 2.2.

2.4.2 Results and Analysis

Quantitative Results. In Table 2.1 and Table 2.2, we summarize the performance comparison results of our HOIF-Net model and various state-of-the-art methods for 3D pose estimation. As can be seen, our model performs the best in most of the actions and also on average under both Protocol #1 and Protocol #2, indicating that HOIF-Net is very competitive. Under Protocol #1, Table 2.1 shows that HOIF-Net performs better than high-order GCN [22] on 14 out of 15 actions, yielding an error reduction of approximately 1.44% on average over high-order GCN. Moreover, our model outperforms semGCN [20] by a relative improvement of 4.86% on average. Under Protocol #2, Table 2.2 shows that our model performs better than high-order GCN with 1.83% error reduction on average, and also achieves better performance on 13 out of 15 actions.

Qualitative Results. Figures 2.5 and 2.6 show the qualitative results obtained by our model for various actions. As can bee seen, the predictions made by HOIF-Net match perfectly the ground truth, indicating the effectiveness of our proposed approach in tackling the 2D-to-3D pose estimation problem.

Figure 2.4: Various types of actions performed by actors in the Human 3.6M dataset.

Table 2.1: Performance comparison of our model and baseline methoos using MPJPE (in millimeters) between the ground truth and estimated pose on Human3.6M under Protocol #1. The last column report the average errors, and bold face numbers indicate the best 3D pose estimation performance.

	Action															
Method	Dire.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Martinez *et al.* [7]	51.8	56.2	58.1	59.0	69.5	78.4	55.2	58.1	74.0	94.6	62.3	59.1	65.1	49.5	52.4	62.9
Sun *et al.* [3]	52.8	54.8	54.2	54.3	61.8	67.2	53.1	53.6	71.7	86.7	61.5	53.4	61.6	47.1	53.4	59.1
Yang *et al.* [8]	51.5	58.9	**50.4**	57.0	62.1	65.4	49.8	52.7	69.2	85.2	57.4	58.4	**43.6**	60.1	47.7	58.6
Fang *et al.* [9]	50.1	54.3	57.0	57.1	66.6	73.3	53.4	55.7	72.8	88.6	60.3	57.7	62.7	47.5	50.6	60.4
Hossain & Little [10]	48.4	**50.7**	57.2	55.2	63.1	72.6	53.0	51.7	66.1	80.9	59.0	57.3	62.4	46.6	49.6	58.3
Pavlakos *et al.* [11]	48.5	54.4	54.4	**52.0**	59.4	65.3	49.9	52.9	65.8	71.1	56.6	52.9	60.9	44.7	47.8	56.2
Sharma *et al.* [12]	48.6	54.5	54.2	55.7	62.2	72.0	50.5	54.3	70.0	78.3	58.1	55.4	61.4	45.2	49.7	58.0
Zhao *et al.* [20]	47.3	60.7	51.4	60.5	61.1	**49.9**	**47.3**	68.1	86.2	**55.0**	67.8	61.0	42.1	60.6	45.3	57.6
Zou *et al.* [22]	49.0	54.5	52.3	53.6	59.2	71.6	49.6	49.8	66.0	75.5	55.1	53.8	58.5	**40.9**	45.4	55.6
Ours	**47.0**	53.7	50.9	52.4	**57.8**	71.3	50.2	**49.1**	**63.5**	76.3	**54.1**	**51.6**	56.5	41.7	**45.3**	**54.8**

Table 2.2: Performance comparison of our model and baseline methoos using PA-MPJPE (in millimeters) between the ground truth and estimated pose on Human3.6M under Protocol #1. The last column report the average errors, and bold face numbers indicate the best 3D pose estimation performance.

	Action															
Method	Dire.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Pavlakos *et al.* [2]	47.5	50.5	48.3	49.3	50.7	55.2	46.1	48.0	61.1	78.1	51.1	48.3	52.9	41.5	46.4	51.9
Zhou *et al.* [6]	47.9	48.8	52.7	55.0	56.8	49.0	45.5	60.8	81.1	53.7	65.5	51.6	50.4	54.8	55.9	55.3
Martinez *et al.* [7]	39.5	43.2	46.4	47.0	51.0	56.0	41.4	40.6	56.5	69.4	49.2	45.0	49.5	38.0	43.1	47.7
Sun *et al.* [3]	42.1	44.3	45.0	45.4	51.5	53.0	43.2	41.3	59.3	73.3	51.0	44.0	48.0	38.3	44.8	48.3
Fang *et al.* [9]	38.2	41.7	43.7	44.9	48.5	55.3	40.2	38.2	54.5	64.4	47.2	44.3	47.3	36.7	41.7	45.7
Hossain & Little [10]	**35.7**	**39.3**	44.6	43.0	47.2	54.0	38.3	**37.5**	51.6	61.3	46.5	41.4	47.3	34.2	39.4	44.1
Zou *et al.* [22]	38.6	42.8	41.8	43.4	44.6	52.9	**37.5**	38.6	53.3	**60.0**	44.4	40.9	46.9	32.2	37.9	43.7
Ours	36.9	42.1	**40.3**	**42.1**	**43.7**	**52.7**	37.9	37.7	**51.5**	60.3	**43.9**	**39.4**	**45.4**	**31.9**	**37.8**	**42.9**

2.4.3 Ablation study

In our ablation experiments, we use the 2D ground truth as input to our model. We start by investigating the effect of the hyperparameters on model performance. Then, we perform a comparative study of our approach against state-of-the-art GCN based methods for 3D human pose estimation.

Parameter Sensitivity. The weight hyperparameter α and β play an important role in the performance of the proposed HOI-Net framework. We conduct a sensitivity analysis to investigate how the performance of our model changes as we vary these two hyperparameters. In Figure 2.7, we analyze the effect of the hyperparameter α by plotting the error values vs. α for both protocols, where α varies from 0.1 to 0.5, and β is set to 1. We can see that our model achieves the lowest error values of MPJPE and PA-MPJPE when $\alpha = 0.12$ and $\alpha = 0.1$m respectively.

In Figure 2.8, we plot the error values vs. β for both protocols by varying the value of β from 0.1 to 1.5, and setting the value of α to 0.1. Notice that the best performance is generally achieved when $\beta = 0.7$

In Table 2.3, we evaluate the performance of our model by decreasing the number of filters or filter maps (i.e. number of columns in the learnable weight matrix) in the higher-order convolutional layers of HOIF-Net. We also report both training time and inference time on single NVIDIA RTX 2070 SUPER GPU. Note that when we decrease the number of filters from 96 to 64, the number of parameters is reduced by more than half. However, the performance of our model drops slightly by 1.66 mm and 1.52 mm in terms of MPJPE and PA-MPJPE, respectively, indicating the robustness of our approach to the choice of number of filters.

Comparison with GCN-based Baselines. We evaluate our model against SemGCN (Zhao *et*

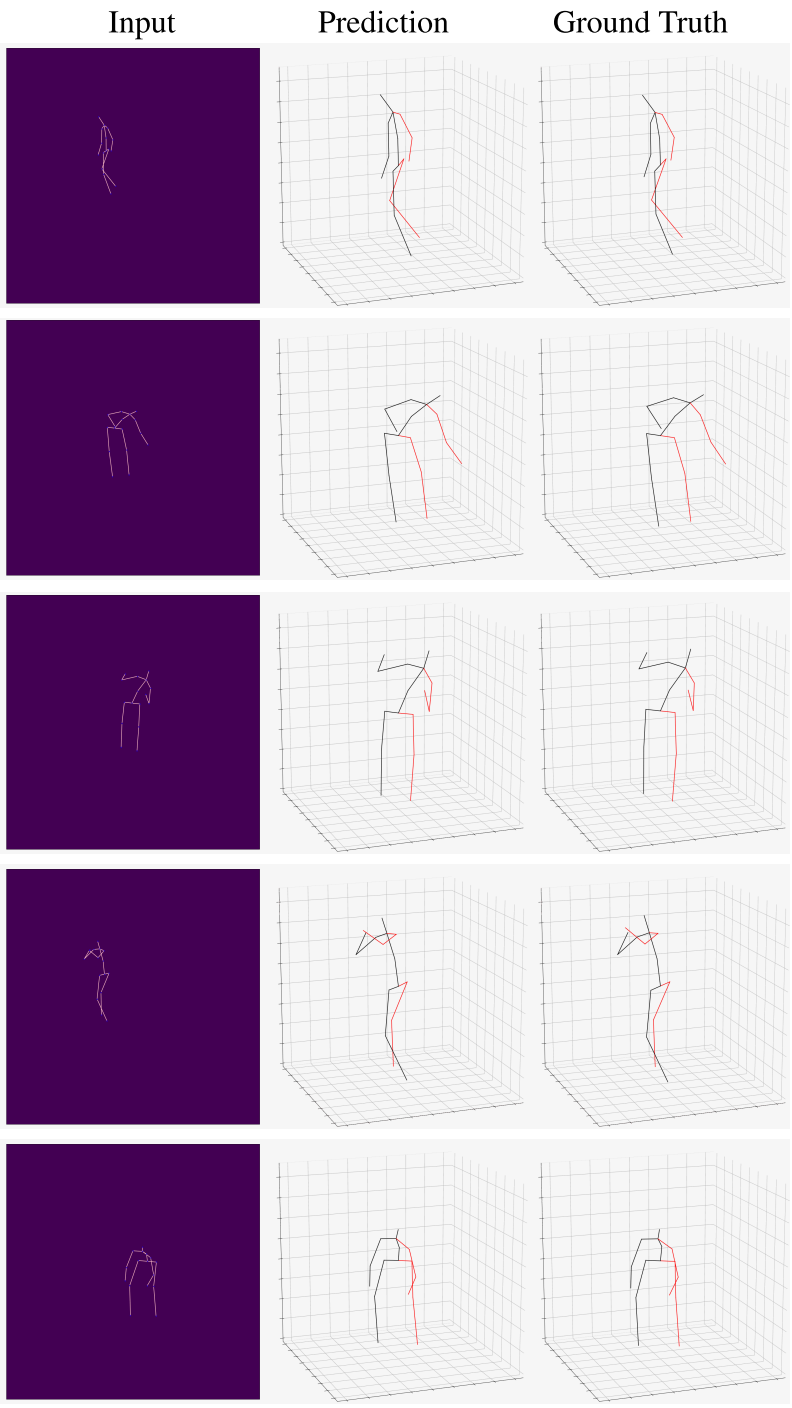

Figure 2.5: Qualitative results obtained by our model on the Human3.6M test set.

al. [20]) and High-order GCN (Zou *et al.* [22]), which are state-of-the-art GCN-based methods for 2D-to-3D pose estimation, and we report the results in Table 2.4. As can be seen, our approach outperforms both semGCN and High-order GCN under Protocol #1, as well as Protocol #2. Under Protocol #1, our HOIF-Net model outperforms semGCN and high-order GCN by 4.02 mm and 1.4

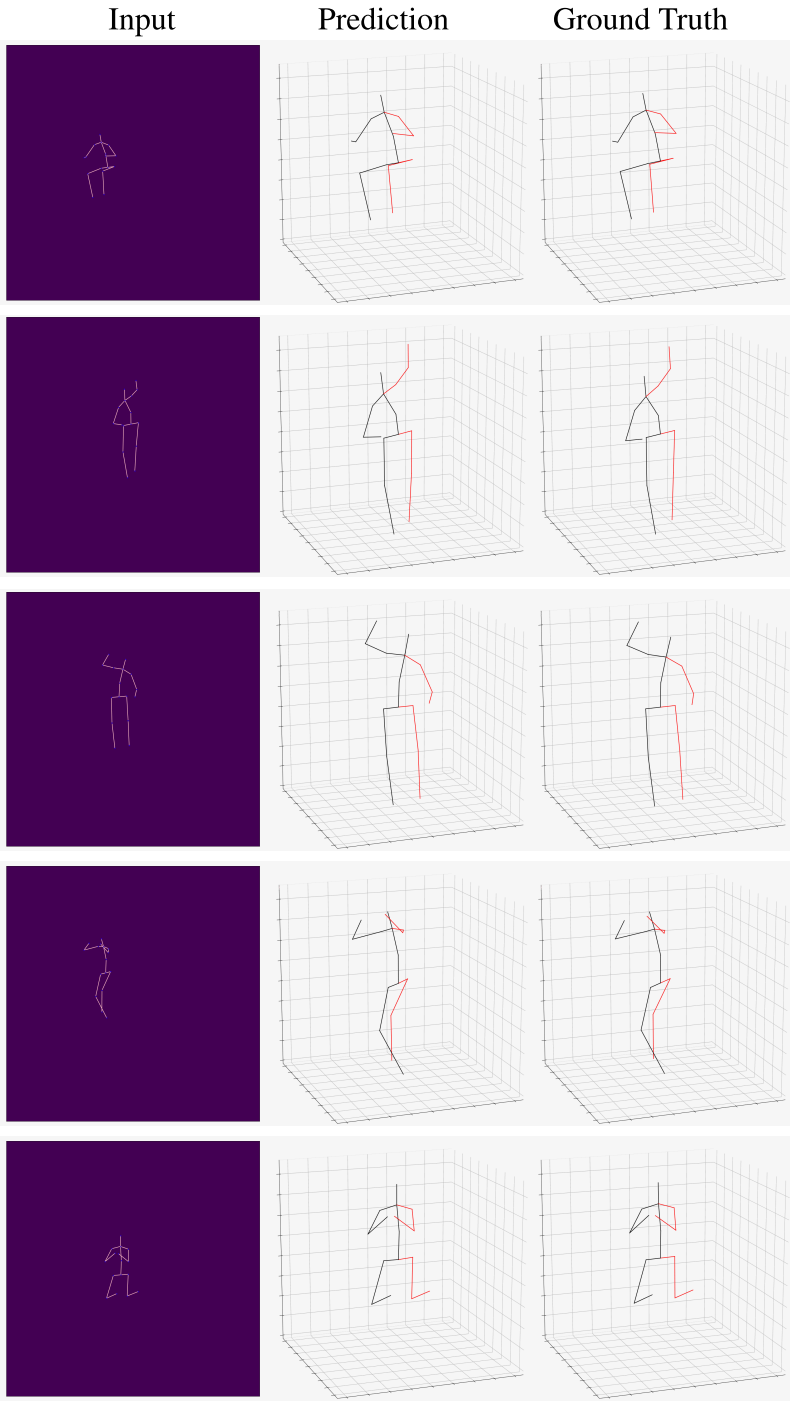

Figure 2.6: Qualitative results obtained by our model on the Human3.6M test set.

mm, corresponding to error reductions of 9.54% and 3.54%, respectively. In the same vein, under Protocol #2, our HOIF-Net model outperforms semGCN and high-order GCN by 3.79 mm and 1.33 mm, corresponding to error reductions of 11.3% and 4.28%, respectively. In addition, our model achieves comparable performance as high-order GCN, while using a much smaller number

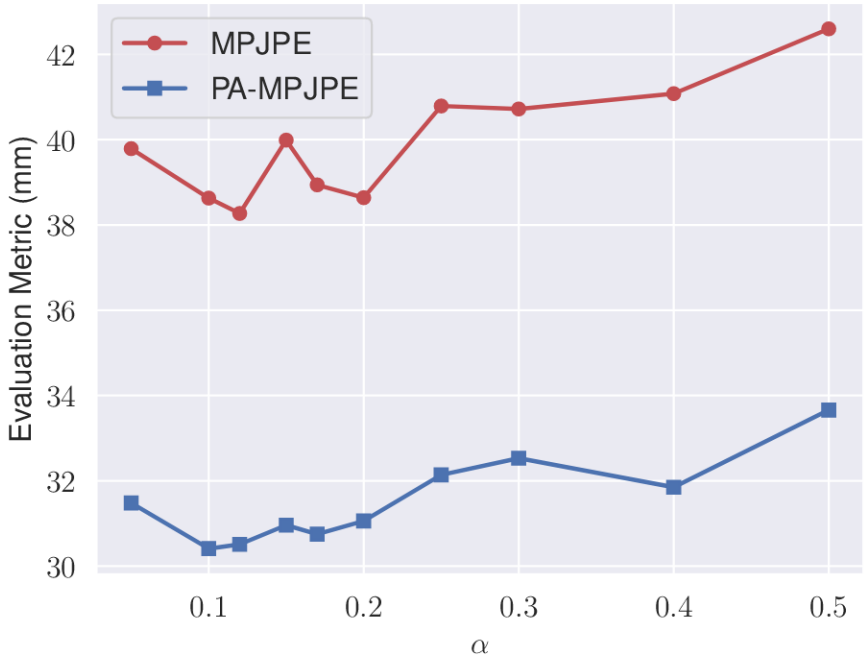

Figure 2.7: Effect of the hyperparameter α on model performance. The value of β is set to 1.

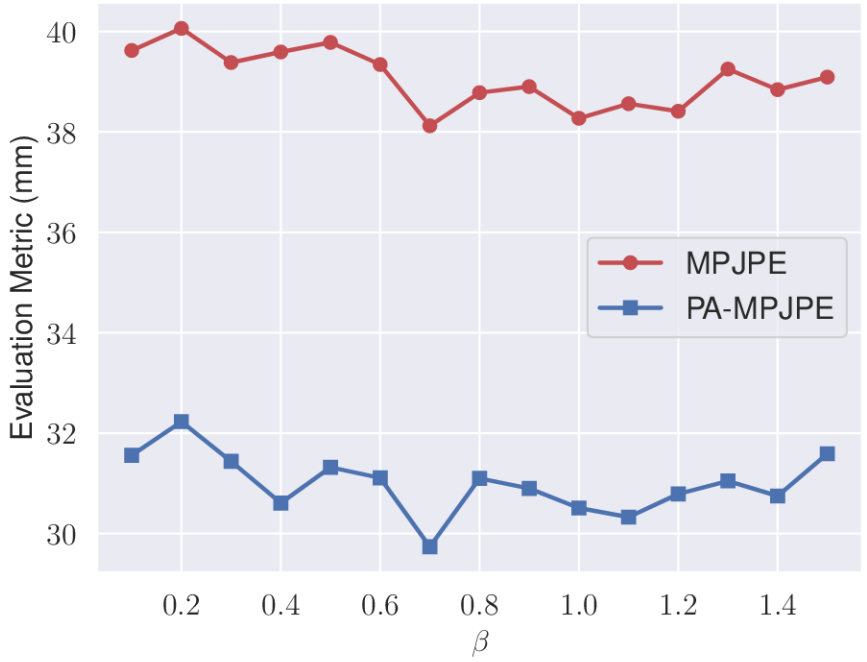

Figure 2.8: Effect of the hyperparameter β on model performance. The value of α is set to 0.1.

of filters (64 compared to 96) and also the number of learned parameters is reduced by more than half.

Table 2.3: Effect of number of filters on model performance. The training time (per-batch) and inference time (per-batch) are also reported. Bold face numbers indicate the best 3D pose estimation performance.

Filters	Parameters	Training Time	Inference Time	MPJPE	PA-MPJPE
96	1.20M	0.070s	0.034s	**38.12**	**29.74**
64	0.54M	0.062s	0.030s	39.78	31.26
32	0.14M	0.059s	0.027s	39.75	32.16
16	0.04M	0.058s	0.027s	43.53	34.74

Table 2.4: Performance comparison of our model and GCN-based methods. Bold face numbers indicate the best 3D pose estimation performance.

Method	Filters	Parameters	MPJPE	PA-MPJPE
SemGCN [20]	96	0.43M	42.14	33.53
High-order GCN [22]	96	1.20M	39.52	31.07
Ours	96	1.20M	**38.12**	**29.74**
Ours	64	**0.54M**	39.78	31.26

CHAPTER 3

Object Detection for Aerial Imagery

In this chapter, we introduce a single-stage object detection model using a class-balanced loss function in conjunction with a feature pyramid network in an effort to mitigate the data imbalance problem without the need to rely on data augmentation. The performance of our model is demonstrated through extensive experiments on a standard aerial image benchmark, achieving comparable or better object detection results in comparison with strong baseline methods.

3.1 Introduction

Unmanned Aerial Vehicles (UAVs) are pilotless aircrafts, which are flown without a pilot-in-command on-board and are either remotely and fully controlled from another place (ground, another aircraft, space) or programmed and fully autonomous. UAVs are generally categorized into systems that are remotely piloted and those that have at least some degree of autonomous operation. The immediate and long-term benefits of UAVs have become increasingly evident in a multitude of areas, including search and rescue missions, retail delivery and logistics, land surveying, agriculture, oil and gas pipeline inspection, emergency response services, traffic management and monitoring, surveillance of critical assets, pandemic management, and monitoring of infrastructural changes over time [15–18]. Benefits offered by UAVs include reduced costs of collecting data, greater efficiency of operations, ability to handle risky operations in dangerous working environments without human intervention, as well as the ability to gather valuable data during and after natural disasters to aid in security and recovery efforts [57]. Also, the recent fifth generation of wireless (5G) technology is anticipated to be a major catalyst for expanded and richer UAV

use cases, especially in industrial automation environments and future potential applications of 5G-enabled autonomous flights.

Object detection refers to the process of detecting and localizing objects in images and videos. It plays a vital role in many UAV applications based on computer vision and artificial intelligence. The basic goal of object detection is to identify the object category and locate the position using a bounding box for every known object within an image. Recent advances in deep learning and control technology have accelerated the development and deployment of UAV applications on a greater scale, making them accessible to a wide range of operators due largely to the relatively low cost of most UAV models. Also, UAVs have a greater range of movement than manned aircraft, as they are able to fly lower and in more directions, allowing them to easily navigate traditionally hard-to-access areas. Since UAVs use the Global Positioning System (GPS), they can be programmed and maneuvered accurately to precise locations. This is especially helpful in a variety of situations, particularly in traffic management and pedestrian public safety using aerial imaging data and videos taken by drones.

With the rapid development of hardware computing and powerful graphics cards, object detection methods have become more accurate and faster at detecting and localizing objects in images and videos. However, the task of UAV traffic detection is much more complex than the general object detection. In fact, objects may appear very small in aerial images depending on the flight altitude of UAVs. Moreover, the shape and outline of the same object can look different due to the various camera viewing angles. Compared to the general object detection tasks, the diverse backgrounds of aerial images can also increase the difficulty in detecting objects. Figure 3.1 shows an example of an aerial image taken from an UAV, where both vehicles and pedestrians are of different shapes and sizes in the image. Deep learning based methods [58] can help mitigate these problems. Convolutional neural networks, for instance, can extract high-level semantic and robust features from images. Compared to traditional models, convolutional neural networks not only have deeper architectures for better expressiveness and performance, but also learn hierarchical feature representations from input images through multi-level non-linear mappings [27]. Hence, deep learning models can achieve state-of-the-art performance in many generic object detection tasks. Nevertheless, there are several challenges to the deployment of deep learning systems on UAVs, particularly for object detection in high-resolution aerial imagery and videos. In addition to accuracy, the detection speed can be crucial, as it determines, for example, the reaction time of decision-makers in emergency situations. Moreover, the performance of a deep learning model depends heavily on the size and quality of image and videos datasets. While the development of powerful graphical processing units can speed up the computation of convolutional neural net-

works, not all deep learning models can be readily applied to object detection for aerial imagery due largely to the size of the deep neural networks architectures, as some models require the learning of millions of parameters.

Existing deep learning based object detectors can be classified into two main categories, namely two-stage detectors and one-stage detectors. Two-stage detectors such as Fast R-CNN [59] and Faster R-CNN [36] use a region proposal network to generate region of interests and get the proposal feature maps to perform the classification of objects. On the other hand, one-stage detectors such as You Only Look Once (YOLO) [37], Single shot multiBox detector (SSD) [38] and RetinaNet [39] remove the step of generating regions of interest, resulting in faster object detection in comparison with two-stage detectors. However, the accuracy of one-stage detectors tends to be relatively lower compared to two-stage detectors. Thus, how to balance the object detection speed and accuracy remains a major challenge in aerial traffic detection tasks. The second challenge of aerial traffic detection is to achieve certain detection accuracy while training the model on limited datasets. While there are several custom datasets available for generic object detection, well-annotated aerial images datasets are not widely available. In addition, compared with image classification problems, the correlation between the number of total objects and the number of images can be extremely insignificant in object detection tasks (One object per image in classification tasks and multiple objects per image in object detection tasks). To address these challenges, we introduce a class-balanced RetinaNet for object detection on aerial images using a class-balanced focal loss function. The main contributions of this work can be summarized as follows:

- We design a class-balanced RetinaNet model for object detection on aerial imagery.
- Instead of performing data augmentation, we use a class-balanced loss function to mitigate the class imbalance problem.
- We demonstrate through extensive experiments and an ablation study that our model outperforms competing baseline methods.

3.2 Proposed Method

In this section, we start by a brief motivation behind the need for class-balanced loss functions. Then, we introduce a Class-Balanced RetinaNet (CB-RetinaNet) model for object detection on aerial images.

Figure 3.1: An aerial image taken from a drone. The objects in an aerial image can have a variety of shapes and sizes depending on the shooting angle of camera, making object detection in aerial images quite challenging.

3.2.1 Motivation

Most of the real-world datasets such as the Microsoft Common Objects in Context (MS-COCO) [60], PASCAL Visual Object Classes (PASCAL-VOC) [61], and KITTI datasets [62] have a common issue of long-tailed distributions of different classes, as illustrated Figure 3.2, leading to a poor performance of deep learning models on those less represented classes [63–65].

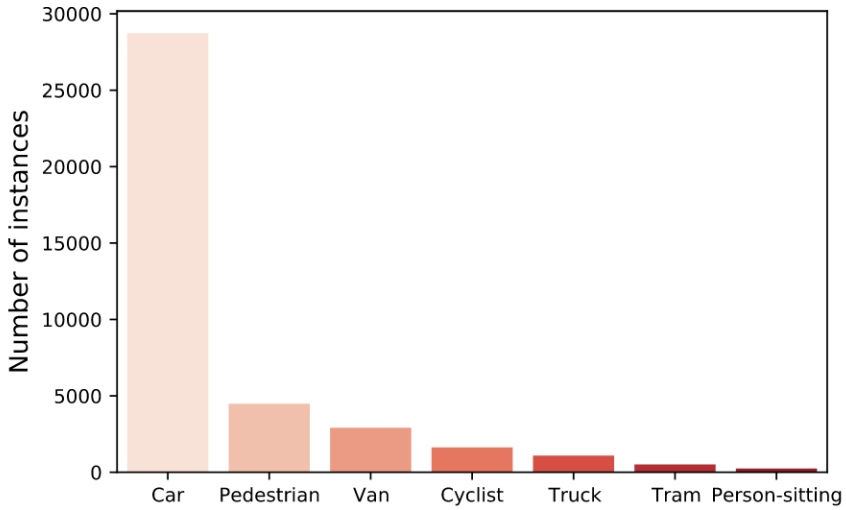

Figure 3.2: Illustration of a long-tailed distribution in the KITTI dataset, which is used for testing computer vision and robotic algorithms targeted to autonomous driving.

There are two main ways to tackle this long-tailed distribution problem: re-weighting the loss or re-sampling the training samples. Re-sampling either up-samples the minority classes or down-samples the majority classes, and both lead to over-fitting of the minority classes when the class imbalance becomes quite large. A remedy is to use data augmentation [66] by performing data augmentation on minority classes in a bid to average the distribution of the dataset. However, data augmentation is not always applicable in object detection, as it operates on a single image and hence may increase the number of instances in the image in an evenly fashion. In addition, data augmentation can be complex and time consuming. Thus, the best remedy is to re-weight the loss to increase the proportion of minority classes in the loss function. In our proposed framework, we adopt a class-balanced focal loss to adjust the contributions of minority classes to the loss function in an effort to mitigate the class imbalance problem and allow our model to detect different classes evenly.

Basic Notions. For a multi-class classification problem consisting of C classes, we denote by $\mathcal{X} = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, N\}$ a set of N samples, where $\mathbf{x}_i$ is the ith training sample and $y_i \in \{1, \ldots, C\}$ is the corresponding true label. The true label of the ith sample can be represented as a one-hot encoding vector $\mathbf{y}_i = (y_{i1}, \ldots, y_{iC})^\intercal$, such that $y_{ic} = 1$ if $i = c$ and 0 otherwise for each class c.

3.2.2 Class-Balanced Loss

Cross-Entropy Loss: For a binary classification problem with two classes 0 (negative class) and 1 (positive class), the cross-entropy loss measures the performance of the classification model whose output is a probability, and is defined as

$$\mathcal{L}(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}), \tag{3.1}$$

where y is the ground-truth label of the positive class, and $\hat{y}$ is the model's predicted label, which is a probability often usually obtained via softmax. Cross-entropy measures how much $\hat{y}$ differs from the true y, and increases as the predicted probability of a sample diverges from the actual value.

Categorical Cross-Entropy Loss: The categorical cross-entropy for the ith sample is defined as

$$\mathcal{L}(\mathbf{y}_i, \hat{\mathbf{y}}_i) = -\sum_{c=1}^{C} y_{ic} \log(\hat{y}_{ic}), \tag{3.2}$$

where C is the total number of classes, y_{ic} is the indicator that the ith sample belongs to the cth class, and $\hat{y}_{ic}$ is the predicted probability that the model associates the ith input with class c.

Note that $\mathbf{y}_i$ is the one-hot encoding vector of the ith sample, while $\hat{\mathbf{y}}_i$ is the vector of predicted probabilities.

The categorical cross-entropy for all samples is given by

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log(\hat{y}_{ic}). \tag{3.3}$$

Balanced Cross-Entropy Loss: The balanced cross-entropy loss is defined as

$$\mathcal{L}(y,\hat{y}) = -\alpha y\log(\hat{y}) - (1-\alpha)(1-y)\log(1-\hat{y}), \tag{3.4}$$

where is α is a weighting factor. If we define $\hat{y}_t$ and α_t as

$$\hat{y}_t = \begin{cases} \hat{y} & \text{if } y = 1 \\ 1-\hat{y} & \text{o.w.} \end{cases} \quad \text{and} \quad \alpha_t = \begin{cases} \alpha & \text{if } y = 1 \\ 1-\alpha & \text{o.w.} \end{cases}$$

then, the balanced cross-entropy loss can be rewritten as

$$\mathcal{L}(\hat{y}_t) = -\alpha_t\log(\hat{y}_t). \tag{3.5}$$

Focal Loss: The large class imbalance encountered during training of dense object detectors overwhelms the cross-entropy loss. To mitigate this issue, the focal loss [39] is used, where a modulation term is applied to the cross-entropy loss function, making it efficient and easy to learn for hard examples. The focal loss is defined as

$$\mathcal{L}(y,\hat{y}) = -\alpha y(1-\hat{y})^{\gamma}\log(\hat{y}) - (1-\alpha)(1-y)\hat{y}^{\gamma}\log(1-\hat{y}), \tag{3.6}$$

where $\gamma > 0$ is a focusing parameter, and α is a weighting factor. The focal loss tries to down-weight the contribution of easy examples so that the network focuses more on hard examples.

The focal loss can be rewritten as

$$\mathcal{L}(\hat{y}_t) = -\alpha_t(1-\hat{y}_t)^{\gamma}\log(\hat{y}_t). \tag{3.7}$$

The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted, while the weighting factor α balances focal loss.

Multi-Class Focal Loss: The multi-class focal loss function is defined as

$$\mathcal{L}(\mathbf{y}_i,\hat{\mathbf{y}}_i) = -\sum_{c=1}^{C}\alpha_c\, y_{ic}(1-\hat{y}_{ic})^{\gamma}\log(\hat{y}_{ic}) \tag{3.8}$$

where C is the total number of classes, γ is a positive focusing parameter, $\alpha_c \in [0, 1]$ is a balancing parameter for the c-th class. The focal loss reduces to the categorical cross-entropy when $\gamma = 0$ and $\alpha_c = 1$.

Note that since the two parameters interact with each other, they should be selected together. Generally speaking, as γ is increased, α_c should be decreased slightly.

The multi-class focal loss function can be rewritten as

$$\mathcal{L}(\mathbf{y}_i, \hat{\mathbf{y}}_i) = -\sum_{c=1}^{C} \alpha_c^t (1 - \hat{y}_{ic}^t)^\gamma \log(\hat{y}_{ic}^t), \tag{3.9}$$

where

$$\hat{y}_{ic}^t = \begin{cases} \hat{y}_{ic} & \text{if } y_{ic} = 1 \\ 1 - \hat{y}_{ic} & \text{o.w.} \end{cases} \text{ and } \alpha_c^t = \begin{cases} \alpha_c & \text{if } y_{ic} = 1 \\ 1 - \alpha_c & \text{o.w.} \end{cases}$$

The multi-class focal loss for all samples is given by

$$\mathcal{L} = -\sum_{i=1}^{N} \sum_{c=1}^{C} \alpha_c^t (1 - \hat{y}_{ic}^t)^\gamma \log(\hat{y}_{ic}^t). \tag{3.10}$$

Class-Balanced Focal Loss: To address the problem of training from imbalanced data, we use the class-balanced focal loss [67], defined as

$$\mathcal{L} = -\sum_{i=1}^{N} \sum_{c=1}^{C} w_{ic}^t (1 - \hat{y}_{ic}^t)^\gamma \log(\hat{y}_{ic}^t), \tag{3.11}$$

with weight w_{ic}^t given by

$$w_{ic}^t = \frac{\alpha_c^t (1 - \beta)}{1 - \beta^{n_i}},$$

where n_i is the number of samples in the ground-truth class $\mathbf{y}_i$, and $\beta \in [0, 1]$ is a hyperparameter chosen via cross-validation. Note that $\beta = 0$ corresponds to the α-balanced focal loss, while $\beta \to 1$ yields $w_{ic}^t = \alpha_c^t / n_i$.

Compared to the focal loss, the classes with a large number of instances contribute less to the class-balanced focal loss, while classes with smaller number of instances contribute more. Therefore, using the class-balanced focal loss we can leverage the distribution of different classes in any dataset to tackle the imbalance problem without applying data augmentation.

3.2.3 Model Architecture

The architecture of our proposed CB-RetinaNet model is similar to RetinaNet in the sense that it is comprised of two task-specific sub-networks, as illustrated in Figure 3.3. The network uses a

Feature Pyramid Network (FPN) [68] backbone combined with a ResNet architecture to extract features. FPN is an image feature extractor that is to able to generate multi-scale feature maps at different image scales in a fully convolutional fashion. Such a pyramid is constructed via both a bottom-up pathway and a top-down pathway. The bottom-up pathway is the feedforward computation of the backbone ConvNet, which computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2, while the top-down pathway hallucinates higher resolution features by upsampling spatially coarser, but semantically stronger, feature maps from higher pyramid levels. The CB-RetinaNet model attaches two sub-networks, one for classifying anchor boxes and another for performing regression from anchor boxes to ground-truth object boxes.

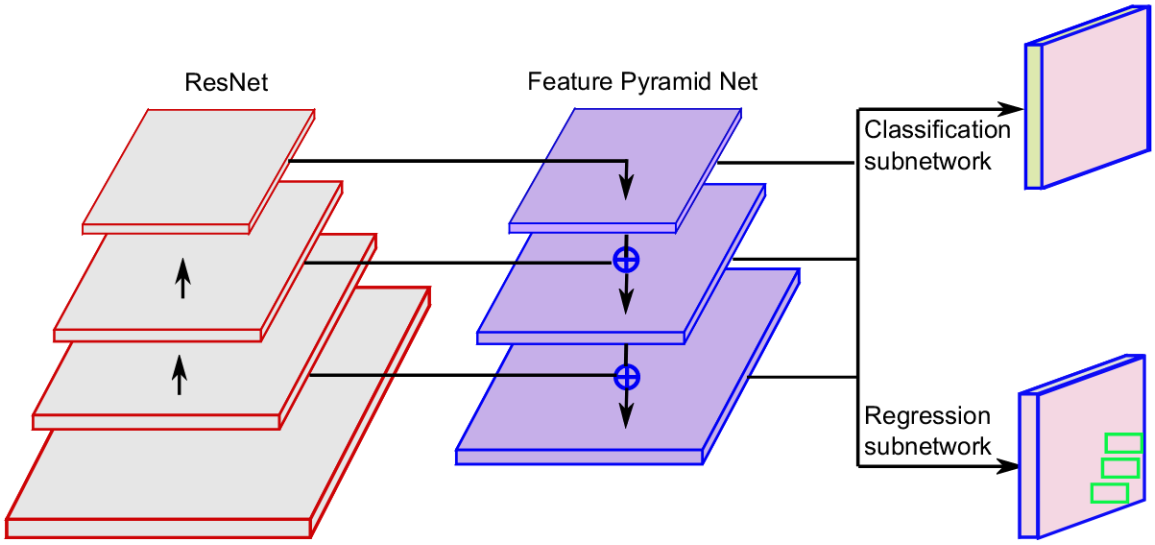

Figure 3.3: CB-RetinaNet architecture for aerial image detection.

3.3 Experiments

In this section, we conduct experiments to assess the performance of our model on an aerial imagery object detection benchmark, and we compare the results of our approach with several state-of-art methods [69, 70].

3.3.1 Experimental Setup

Dataset. We demonstrate and analyze the performance of the proposed CB-RetinaNet model on the VisDrone2019-DET dataset [70], which consists of 10 classes: pedestrian, people, bicycle, car, vans, truck, tricycle, awning-tricycle, bus and motors. The number of instances in each class is shown in Figure 3.4. The dataset consists of 79 sequences with 33,366 frames in total, including

three non-overlapping subsets: training set (56 video clips with 24,198 frames), validation set (7 video clips with 2,846 frames) and testing set (16 video clips with 6,322 frames). These video frames are captured in different cities under different outdoor weather conditions. Also, there are two extra classes, namely ignored regions and others, which indicate the regions in the objects that are either too small or too dense to classify. To simplify training process, we fit all of the classes to train the network.

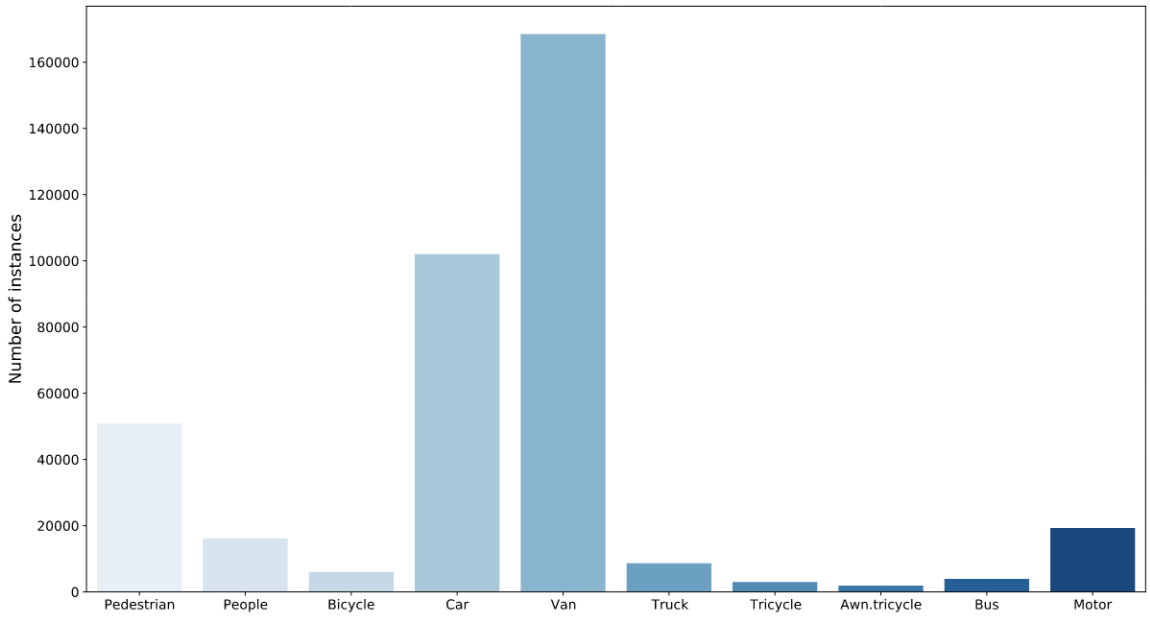

Figure 3.4: Number of class instances in VisDrone2019-DET

Baseline Methods. To assess the performance of our CB-RetinaNet model, we compare the results with both two- and one-stage object detectors. A brief description of the baseline methods can be summarized as follows:

- **Cascade R-CNN** is a multi-stage object detection architecture, which consists of a sequence of detectors trained in end-to-end fashion with increasing IoU thresholds, making the detectors more selective against close false positives [71]. It can be built with any two-stage object detector based on the R-CNN framework.

- **Cascade R-CNN+** is an extension of Cascade R-CNN that leverages the focal loss, and uses the feature pyramid network as a base detector and SEResNeXt-50 [72] as a backbone.

- **Deformable Cascade R-CNN (Cascade R-CNN++)** uses ResNeXt-101 and feature pyramid network to extract features. It also employs the deformable convolution to reinforce the feature extractor.

- **DA-RetinaNet** is an improved RetinaNet model that uses feature alignment and more scales of smaller anchors, as well as ResNet-101 as a backbone to extract features. It also uses an additional channel attention module and a spatial attention module [73] to improve the detection accuracy.

- **EnDet** is an ensemble deep object detector, which combines two YOLOv3-based networks and two Faster R-CNN based networks. The YOLOv3 network is modified with spatial pyramid pooling module, while a residual attention module is incorporated into the feature pyramid network in the Faster R-CNN network based on a graph clique.

- **ConstraintNet** is an object detector that uses constraint keypoint triplets and a constraint corner pooling for feature extraction around the target object in lieu of the entire image. This network detects objects by restricting their width and height, resulting in improved precision and recall.

Implementation Details. We fit the whole dataset into the network and we set the training process of the model for a maximum 50 epochs on a desktop computer equipped with an 8-GB-RAM RTX 2070 SUPER GPU. The hyperparameters γ and β are set to 0.5 and 0.999, respectively. We also use early stopping to avoid overfitting and speed-up the training process. The learning rate is set to 1e-5 and we use ResNet101 as our model's backbone. The range interval for α is set to $[0.25, 0.75]$.

Evaluation Metric. We use the mean average precision (mAP) as a metric to evaluate the performance of our object detection model. The mAP metric compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections. Usually, object detection models are evaluated with different intersection-of-union (IoU) thresholds where each threshold may give different predictions from the other thresholds, where each threshold may give different predictions from the other thresholds. The IoU score measures how close is the predicted box to the ground-truth box, and ranges from 0 to 1, where 1 is the optimal result. When the IoU is greater than the threshold, then the box is classified as "positive" as it surrounds an object. Otherwise, it is classified as "negative". The mAP metric is defined as the mean of average precisions for all classes, i.e.

$$mAP = \frac{1}{C}\sum_{c=1}^{C} AP_c, \tag{3.12}$$

where the average precision (AP) is the area under the Precision-Recall curve, and Precision and Recall are given by

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{3.13}$$

and

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{3.14}$$

with TP (True Positive) denoting the total number of correct detections with IoU > 0.5, and FP (False Positive) denoting the total number of wrong detections with IoU ≤ 0.5.

3.3.2 Results and Analysis

Figure 3.5 shows the training loss of our model and RetinaNet, while Figure 3.6 shows the mAP results during the training process on the validation set. Since the training process can be very time-consuming, we use early stopping to speed it up and also to avoid overfitting. As can be seen in Figure 3.6, the mAP values on the validation set start to fluctuate after just 8 epochs, indicating that the model has been trained well.

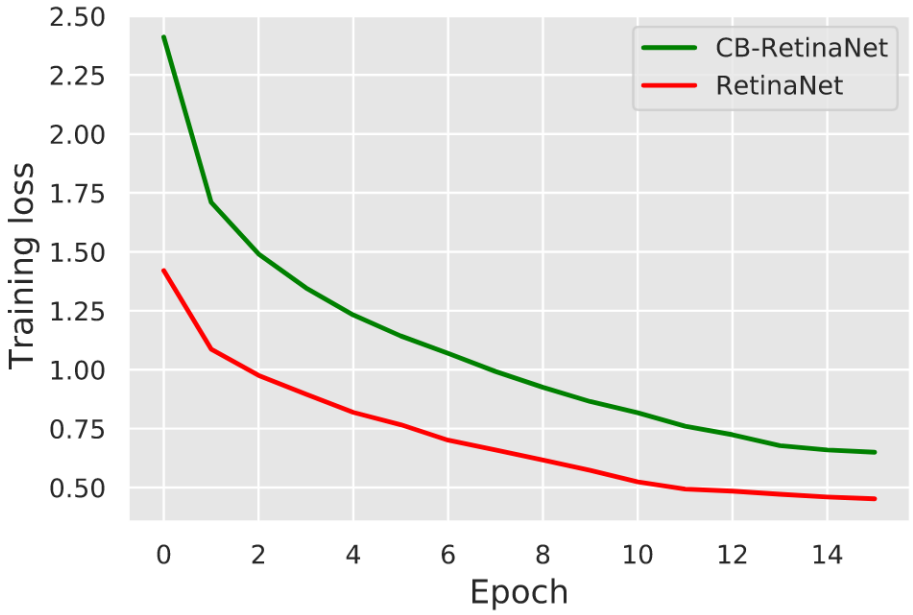

Figure 3.5: Model training history comparison between RetinaNet and CB-RetinaNet.

In Table 3.1, we report the performance comparison results of our model and baseline methods using the AP and mAP evaluation metrics on the VisDrone2019-DET test set. As can be seen, CB-RetinaNet shows significant improvements in terms of AP and mAP compared to the baselines. Our model performs better than RetinaNet on 7 out of 10 classes, achieving around 15%

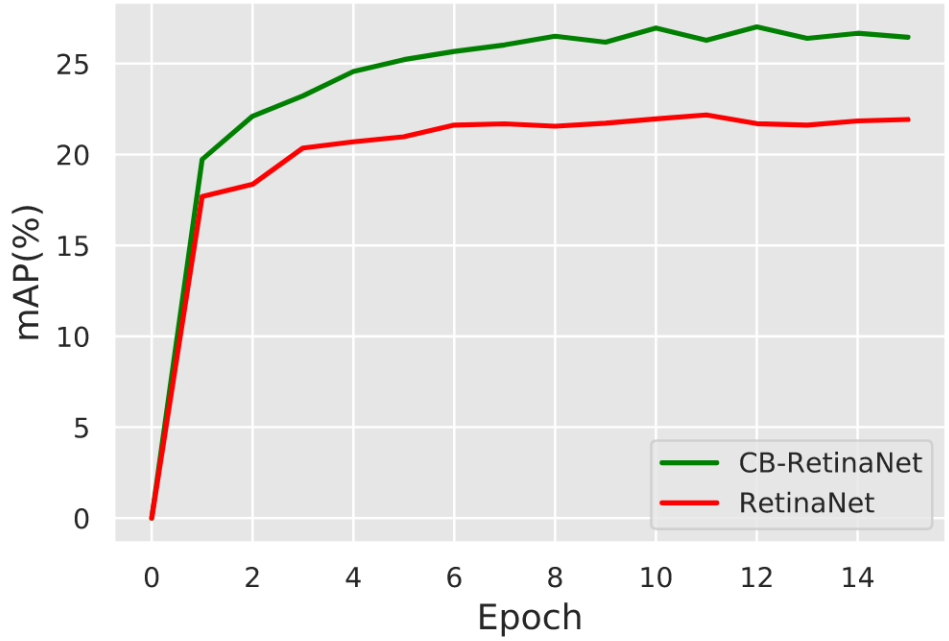

Figure 3.6: mAP results of RetinaNet and CB-RetinaNet on the validation set.

improvement in terms of mAP. Moreover, our model yields an 25% improvement in terms of AP on the car class. In addition, our model is competitive against two-stage detectors, outperforming Cascade R-CNN+ by 6% and Cascade R-CNN++ by 10% in terms of mAP.

Table 3.1: Performance comparison of our model and baseline methods using AP and mAP on the VisDrone2019-DET test set. Boldface numbers indicate the best performance.

	Class										
Method	pedestrian	people	bicycle	car	van	truck	tricycle	awning tricycle	bus	motor	mAP (%)
Cascade R-CNN+	17.75	5.08	3.54	42.01	26.50	22.58	15.96	12.71	33.28	**23.31**	20.27
Cascade R-CNN++	20.96	7.53	3.05	41.92	21.11	15.34	13.78	10.04	22.38	13.19	16.93
DCRCNN	15.21	8.88	7.06	32.51	25.94	20.40	**19.15**	**15.72**	35.65	11.79	19.23
RetinaNet	9.91	2.92	1.32	28.99	17.82	11.35	10.93	8.02	2.21	7.03	11.60
DA-RetinaNet	19.63	7.22	2.76	40.47	22.15	16.95	11.53	9.47	23.74	13.19	16.71
EnDet	17.19	7.45	2.73	40.16	26.31	18.08	15.42	14.19	31.98	11.50	18.50
ConstraintNet	17.49	6.81	2.59	39.17	25.17	14.41	11.02	8.64	18.11	11.84	15.53
CB-RetinaNet	**23.45**	**10.76**	**8.48**	**66.99**	**28.02**	**30.70**	14.62	9.41	**49.19**	20.60	**26.22**

3.3.3 Visualization Results

In Figures 3.7 and 3.8, we show some visualization results of our model. As can bee seen, our model shows high detection performance and reliability at both daytime and nighttime, indicating the robustness of our approach to changes in lighting conditions. In addition, the average inference

time for each image is 0.075 seconds on an RTX 2070 SUPER GPU, and thus it can reach around 14 frames per second (FPS) in real-time processing with the same GPU on a drone. Additional visualization results are shown in Figure 3.9.

Figure 3.7: Detection results at nighttime. The image is chosen from the test set.

Figure 3.8: Detection results at daytime. The image is chosen from the test set.

3.3.4 Ablation study

In our ablation experiments, we start by investigating the effect of the hyperparameters α, β and γ on model performance. Then, we perform a sensitivity analysis of our model to the anchor size.

Figure 3.9: Visualization results on the VisDrone2019-DET test set.

Parameter Sensitivity. The hyperparameters β and γ play an important role in our CB0-RetinaNet model. We conduct a sensitivity analysis to investigate how the performance of our model changes as we vary these two hyperparameters. In Figures 3.10 and 3.11, we analyze the effect of the hyperparameter β by plotting the training loss and mAP on the validation set using various values of β. Notice that our model achieves the highest mAP result when β is set to 0.999.

In Figures 3.12 and 3.13, we plot the the training loss and mAP on the validation for various values of γ. As can be seen, our model achieves the highest mAP when γ is set to 0.5. Hence, for the best combination of hyperparameters, we use $\beta = 0.999$ and $\gamma = 0.5$.

We also tested the performance of our model using different intervals for α. The training loss and mAP results of the validation set are shown in Figures 3.14 and 3.15. As can be seen, our model achieves the best performance when $\alpha \in [0.25, 0.75]$.

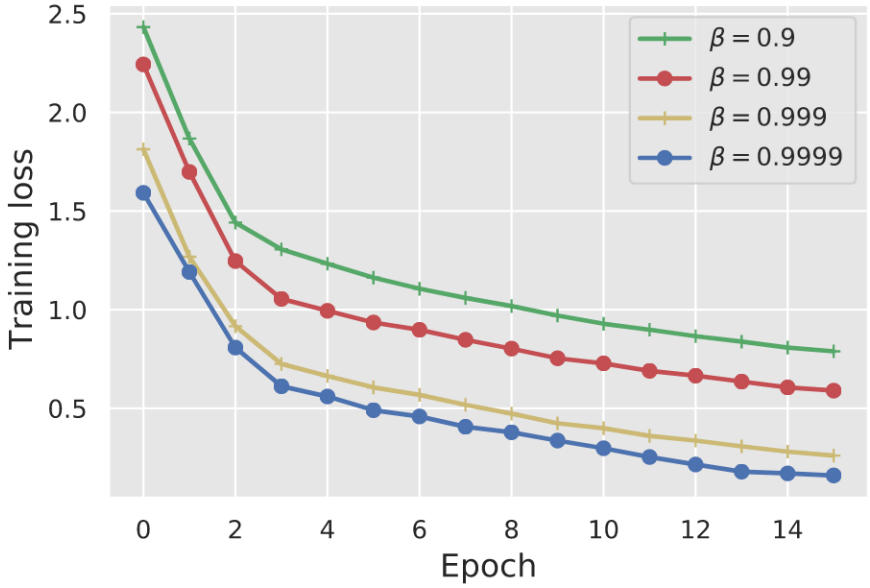

Figure 3.10: Training loss of our model with increasing number of epochs and using different values of β.

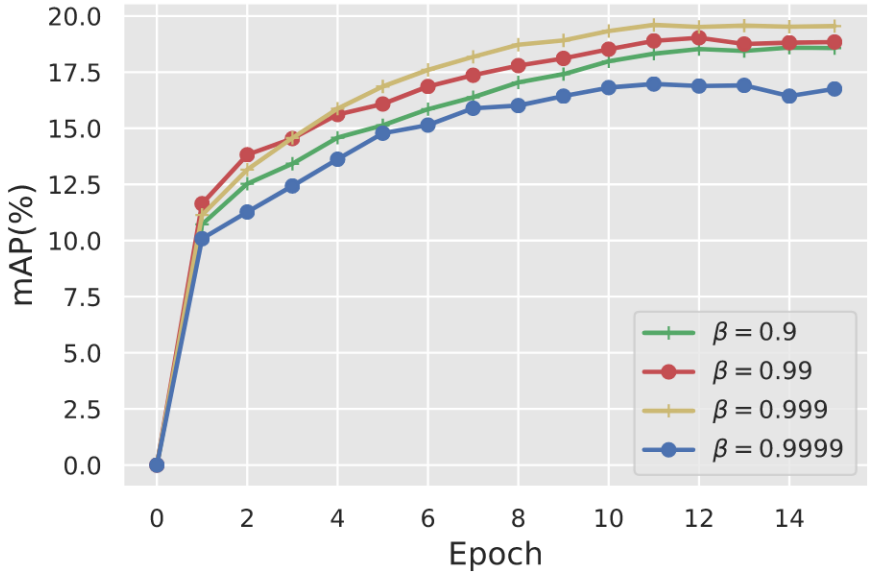

Figure 3.11: Mean average precision results under different values of β.

Impact of anchor size. In Table 3.2, we report the performance of the CB-RetinaNet model under different anchor sizes. According to [39], the original anchors have areas of 32^2 to 512^2 on pyramid levels P3 to P7, respectively. However, as objects in aerial images can be tiny, in our experiments we scale down the anchor areas to 16^2 to 256^2 on pyramid levels P3 to P7. As reported in Table 3.2, after applying a small anchor size, our model shows significant improvements in terms of AP in all of the classes, especially for smaller objects like pedestrians, from 18.74% to 23.45%. Moreover, our model achieves approximately 1.6% improvements in terms of mAP.

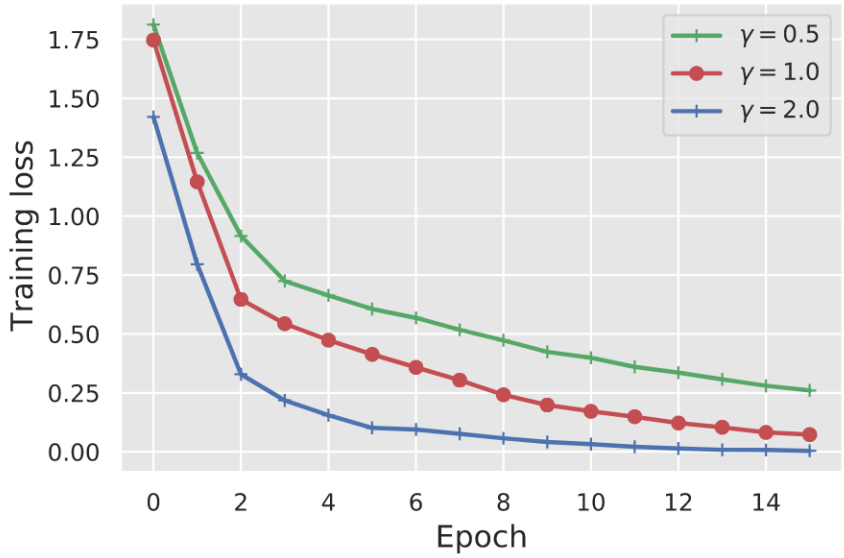

Figure 3.12: Training loss of our model with increasing number of epochs and using different values of γ.

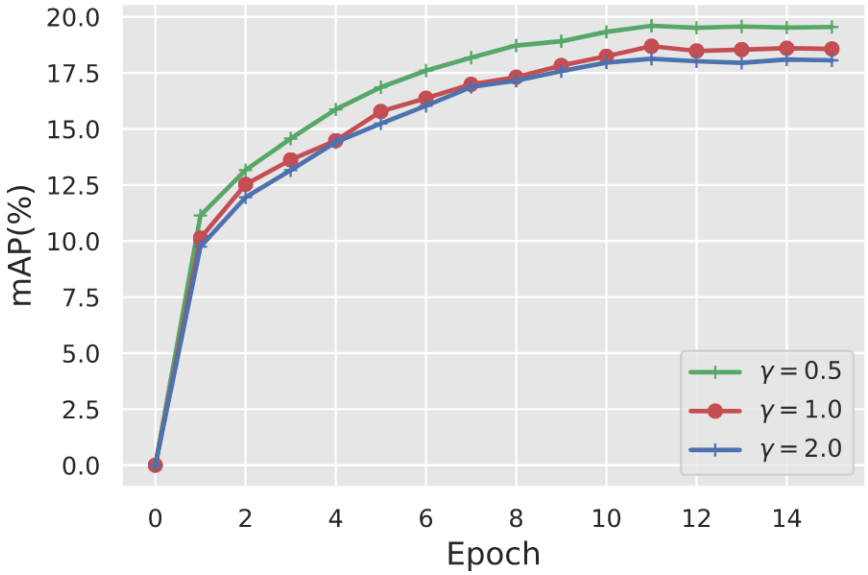

Figure 3.13: Mean average precision results under different values of γ.

Table 3.2: Mean average precision results for our CB-RetianNet model on the VisDrone2019-DET test sets using different anchor sizes. Boldface numbers indicate the best performance.

	Class										
Anchor size	pedestrian	people	bicycle	car	van	truck	tricycle	awning tricycle	bus	motor	mAP(%)
Original	18.74	9.89	5.21	64.84	27.31	29.86	12.84	9.12	48.98	19.95	24.67
Small	**23.45**	**10.76**	**8.48**	**66.99**	**28.02**	**30.70**	**14.62**	**9.41**	49.19	**20.60**	**26.22**

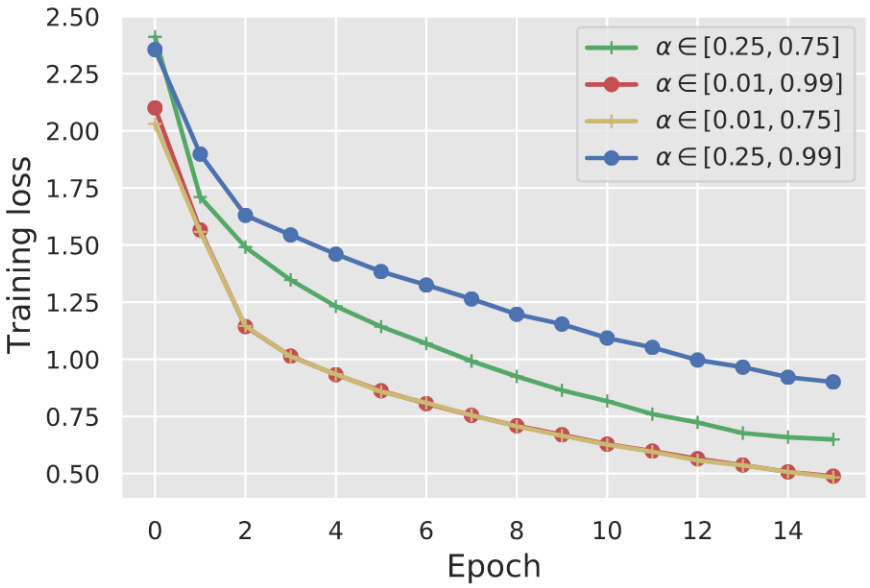

Figure 3.14: Training loss of our model with increasing number of epochs and using different intervals for α.

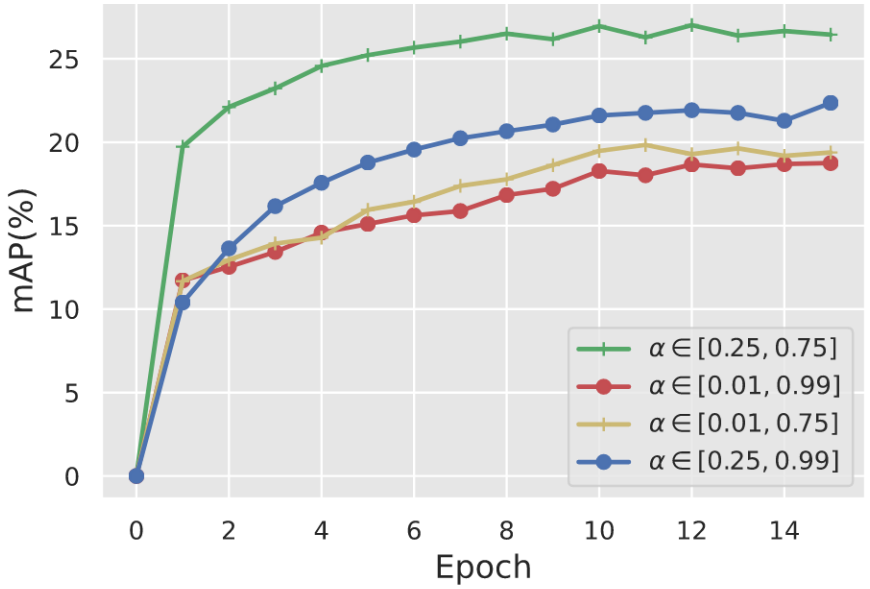

Figure 3.15: Mean average precision results under different intervals for α.

CHAPTER 4

Conclusions and Future Work

This thesis has presented efficient deep neural network architectures for 3D human pose estimation and object detection. The proposed higher-order graph convolutional network for 3D pose estimation is motivated by the concept of implicit fairing on graphs. It takes 2D poses as input and regresses the 3D poses of the human body by following the two-stage paradigm, which consists of using an off-the-shelf, state-of-art 2D pose detector together with a graph convolutional network for predicting the 3D pose locations from the 2D predictions. The aggregation scheme of the proposed approach uses residual connections to tackle the oversmoothing problem, as well as multi-hop neighborhoods to capture long-range dependencies between body joints. For object detection, design a class-balanced RetinaNet model for object detection using a class-balanced focal loss function, achieving competitive accuracy and speed on object detection for UAV imagery. The proposed object detector leverages a feature pyramid network in an effort to mitigate the data imbalance problem without the need to rely on data augmentation. We have demonstrated through extensive experiments and ablation studies the superior performance of the proposed methods in comparison with existing techniques in the literature. For 3D human pose estimation, we performed quantitative and qualitative evaluations on a large-scale dataset, and we evaluated the results using the average Euclidean distance between the predicted 3D joint positions and ground truth after the alignment of the root joint, as well as the the Procrustes-aligned mean per joint position error. In Section 4.1, the contributions made in each of the previous chapters and the concluding results drawn from the associated research work are presented. The limitations of the proposed approaches are discussed in Section 4.2. Suggestions for future research directions related to this thesis are also provided in Section 4.3.

4.1 Contributions of the Thesis

4.1.1 Higher-Order Implicit Fairing Networks for 3D Human Pose Estimation

In Chapter 2, we proposed a higher-order implicit fairing network with initial residual connections for 3D human pose estimation, with the aim to alleviate the oversmoothing problem in graph convolutional networks, and also to capture long-range dependencies between body joints by enabling the model to aggregate multi-hop neighbors through feature concatenation. Empirical experiments and ablation studies showcase the merits of our model, and demonstrate its competitive performance in comparison with state-of-the-art methods for 3D human pose estimation.

4.1.2 Object Detection for Unmanned Aerial Vehicles Imagery

In Chapter 3, we introduced a class-balanced RetinaNet model for object detection on aerial images using a class-balanced loss function in conjunction with a feature pyramid network. We showcased the merits of our model through extensive experiments, demonstrating its competitive performance on the Visdrone-2019Det dataset in comparison with competing baseline methods. Our proposed model not only balances the percentage of minority class contributions in the dataset, but also provides substantial improvements in average accuracy for the minority classes.

4.2 Limitations

While the proposed HOIF-Net framework for 3D human pose estimation has the ability to aggregate multi-hop features of nodes in a skeletal graph while mitigating the oversmoothing problem in deep neural networks, it does not, however, regress 3D keypoints from images in an end-to-end fashion because the HOIF-Net architecture is comprised of two main stages, which are usually decoupled from each other. On the other hand, the performance of the proposed object detection network depends heavily on the size and quality of image and videos datasets. Despite the rapid development of powerful graphical processing units that can considerably speed up the computation of convolutional neural networks, not all deep learning models can be readily adopted to object detection for aerial imagery due largely to the size of the deep neural networks architectures, as some models require the learning of millions of parameters. This issue can be remedied by leveraging very deep pre-trained models.

4.3 Future Work

Several interesting research directions, motivated by this thesis, are discussed below:

4.3.1 HOIF-Net for Other Downstream Tasks

We plan to apply the proposed framework in other downstream tasks such as semi-supervised node and graph classification, anomaly detection, and recommendation. We aim to apply HOIF-Net on complex graph structures to test the improvements of it.

4.3.2 Object Detention in Real-Time

With the fast growing computing power of GPUs, the execution time of complex models has become significantly shorter. We plan to carry out a comprehensive experimental analysis in an effort to improve our object detection model's performance and deploy it in the real-world setting.

4.3.3 Building of larger aerial images dataset

Since the performance of deep learning models for object detection depends on the quality of the training datasets, we intend to collect additional aerial images under different conditions, including altitude, weather, and time in order to expand the existing aerial image dataset. We also plan to add more object classes to the dataset with the goal of training more comprehensive object detectors for UAV imagery.

References

[1] S. Li and A. B. Chan, "3D human pose estimation from monocular images with deep convolutional neural network," in *Proc. Asian Conference on Computer Vision*, pp. 332–347, 2014.

[2] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3D human pose," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7025–7034, 2017.

[3] X. Sun, J. Shang, S. Liang, and Y. Wei, "Compositional human pose regression," in *Proc. IEEE International Conference on Computer Vision*, pp. 2602–2611, 2017.

[4] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Pecognition*, pp. 7103–7112, 2018.

[5] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proc. European Conference on Computer Vision*, pp. 483–499, 2016.

[6] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3D human pose estimation in the wild: a weakly-supervised approach," in *Proc. IEEE International Conference on Computer Vision*, pp. 398–407, 2017.

[7] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3D human pose estimation," in *Proc. IEEE International Conference on Computer Vision*, pp. 2640–2649, 2017.

[8] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang, "3D human pose estimation in the wild by adversarial learning," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5255–5264, 2018.

[9] H.-S. Fang, Y. Xu, W. Wang, X. Liu, and S.-C. Zhu, "Learning pose grammar to encode human body configuration for 3D pose estimation," in *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[10] M. Rayat Imtiaz Hossain and J. J. Little, "Exploiting temporal information for 3D human pose estimation," in *Proc. European Conference on Computer Vision*, pp. 68–84, 2018.

[11] G. Pavlakos, X. Zhou, and K. Daniilidis, "Ordinal depth supervision for 3D human pose estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7307–7316, 2018.

[12] S. Sharma, P. T. Varigonda, P. Bindal, A. Sharma, and A. Jain, "Monocular 3D human pose estimation by generation and ordinal ranking," in *Proc. IEEE International Conference on Computer Vision*, pp. 2325–2334, 2019.

[13] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan, "3D hand shape and pose estimation from a single RGB image," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10833–10842, 2019.

[14] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3D human pose estimation in video with temporal convolutions and semi-supervised training," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7753–7762, 2019.

[15] U. R. Mogili and B. Deepak, "Review on application of drone systems in precision agriculture," *Procedia computer science*, vol. 133, pp. 502–509, 2018.

[16] V. Baiocchi, D. Dominici, and M. Mormile, "UAV application in post-seismic environment," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, p. W2, 2013.

[17] F. Heintz, P. Rudol, and P. Doherty, "From images to traffic behavior - a UAV tracking and monitoring application," in *Proc. International Conference on Information Fusion*, pp. 1–8, 2007.

[18] D. Hausamann, W. Zirnig, G. Schreier, and P. Strobl, "Monitoring of gas pipelines-a civil UAV application," *Aircraft Engineering and Aerospace Technology: An International Journal*, vol. 77, no. 5, pp. 352–360, 2005.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[20] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. N. Metaxas, "Semantic graph convolutional networks for 3D human pose regression," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3425–3435, 2019.

[21] Y. Cai, L. Ge, J. Liu, J. Cai, T.-J. Cham, J. Yuan, and N. M. Thalmann, "Exploiting spatial-temporal relationships for 3D pose estimation via graph convolutional networks," in *Proc. IEEE International Conference on Computer Vision*, pp. 2272–2281, 2019.

[22] Z. Zou, K. Liu, L. Wang, and W. Tang, "High-order graph convolutional networks for 3D human pose estimation," in *Proc. British Machine Vision Conference*, 2020.

[23] K. Xu, C. Li, Y. Tian, T. Sonobe, K. ichi Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. International Conference on Machine Learning*, 2018.

[24] J. Klicpera, A. Bojchevski, and S. G$'$unnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2019.

[25] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*, pp. 1725–1735, 2020.

[26] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. International Conference on Machine Learning*, pp. 21–29, 2019.

[27] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005.

[29] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proc. IEEE International Conference on Image Processing*, vol. 1, pp. I–I, 2002.

[30] S. Zhang, C. Bauckhage, and A. B. Cremers, "Informed Haar-like features improve pedestrian detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 947–954, 2014.

[31] Y. Bazi and F. Melgani, "Convolutional SVM networks for object detection in UAV imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 6, pp. 3107–3118, 2018.

[32] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[33] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review, neural computing," 2017.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[35] R. Girshick, "Fast R-CNN," in *Proc. IEEE international Conference on Computer Vision*, pp. 1440–1448, 2015.

[36] 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*, pp. 91–99, 2015.

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[38] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. European Conference on Computer Vision*, pp. 21–37, 2016.

[39] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017.

[40] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271, 2017.

[41] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[42] T. Kipdf, , and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[43] F. Wu, A. H. Souza, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. International Conference on Machine Learning*, 2019.

[44] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI Conference on Artificial Intelligence*, pp. 3538–3545, 2018.

[45] L. Zhao and L. Akoglu, "PairNorm: Tackling oversmoothing in GNNs," in *International Conference on Learning Representations*, 2020.

[46]

[47] G. Taubin, T. Zhang, and G. Golub, "Optimal surface smoothing as filter design," in *Proc. European Conference on Computer Vision*, 1996.

[48] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[49] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing*, pp. 3844–3852, 2016.

[50] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[51] F. M. Bianchi, D. Grattarola, C. Alippi, and L. Livi, "Graph neural networks with convolutional ARMA filters," *arXiv preprint arXiv:1901.01343*, 2019.

[52] A. Wijesinghe and Q. Wang, "DFNets: Spectral CNNs for graphs with feedback-looped filters," in *Advances in Neural Information Processing Systems*, 2019.

[53] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proc. SIGGRAPH*, pp. 317–324, 1999.

[54] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2013.

[55] K. He, G. Gkioxari, P. Dollár, , and R. Girshick, "Mask R-CNN," in *Proc. IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017.

[56] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. European Conference on Vomputer Vision*, pp. 740–755, 2014.

[57] J. Linchant, J. Lisein, J. Semeki, P. Lejeune, and C. Vermeulen, "Are unmanned aircraft systems (uas s) the future of wildlife monitoring? a review of accomplishments and challenges," *Mammal Review*, vol. 45, no. 4, pp. 239–252, 2015.

[58] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[59] L. Quan, D. Pei, B. Wang, and W. Ruan, "Research on human target recognition algorithm of home service robot based on Fast R-CNN," in *Proc. IEEE International Conference on Intelligent Computation Technology and Automation*, pp. 369–373, 2017.

[60] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[61] 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*, vol. 88, no. 2, pp. 303–338, 2010.

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

[63] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2537–2546, 2019.

[64] S. Sharma, N. Yu, M. Fritz, and B. Schiele, "Long-tailed recognition using class-balanced experts," *arXiv preprint arXiv:2004.03706*, 2020.

[65] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[66] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[67] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277, 2019.

[68] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017.

[69] D. R. Pailla, "Visdrone-det2019: The vision meets drone object detection in image challenge results," 2019.

[70] P. Zhu, L. Wen, D. Du, X. Bian, Q. Hu, and H. Ling, "Vision meets drones: Past, present and future," *arXiv preprint arXiv:2001.06303*, 2020.

[71] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6154–6162, 2018.

[72] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

[73] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "CBAM: Convolutional block attention module," in *Proc. European Conference on Computer Vision*, pp. 3–19, 2018.