UNIVERSITY OF CALIFORNIA
RIVERSIDE

Deep Learning Approaches for Scene Understanding

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Hengyue Liu

June 2024

Dissertation Committee:

Dr. Bir Bhanu, Chairperson
Dr. Konstantinos Karydis
Dr. Nanpeng Yu

The Dissertation of Hengyue Liu is approved:

_____

_____

_____

Committee Chairperson

University of California, Riverside

Acknowledgments

I thank my dissertation chair, Dr. Bir Bhanu, who never gives up on motivating me to finish my degree. I would like to thank Dr. Konstantinos Karydis and Dr. Nanpeng Yu for being my committee members on my dissertation defense. I would also want to thank Dr. Matthew Barth and Dr. Craig Schroeder for conferring my doctoral candidacy. I am grateful for the friends/colleagues who were part of my graduate life: Dr. Adam Witmer, Alex Woonggi Shin, Ankit Jain Rakesh Kumar, Dr. Runze Li, Saisri Padmaja Jonnalagedda, and Xiu Zhang. My research was supported by Bourns Endowment and funds and a gift from SEVAai. Inc. This dissertation was compiled from "Pose-guided R-CNN for Jersey Number Recognition in Sports", "JEDE: Universal Jersey Number Detector for Sports", "Fully Convolutional Scene Graph Generation", "RepSGG: Novel Representations of Entities and Relationships for Scene Graph Generation", and "Dynamically Throttleable Neural Networks, by Hengyue Liu et al. © 2019, 2020 the IEEE, and 2022 Springer-Verlag GmbH Germany, part of Springer Nature.

To my mother Chunhua Bai, my father Tinggeng Liu, I thank you for all the love and support. I also thank my love Heng Fang, Mufasa Fang, and Chototsu Moushin Fang, for their company.

ABSTRACT OF THE DISSERTATION

Deep Learning Approaches for Scene Understanding

by

Hengyue Liu

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, June 2024
Dr. Bir Bhanu, Chairperson

Scene understanding is an important aspect of computer vision, encompassing a variety of tasks such as image classification, object detection, scene graph generation, and action recognition. With the advances of deep learning models, scene understanding has seen significant improvements in accuracy, robustness, and efficiency, enabling more sophisticated and reliable applications in automated sports analytics, visual relationship detection, and dynamic neural networks. Deep learning techniques that perform on standardized benchmarks often struggle when applied in real-world scenarios where the environment variables are often unconstrained and diverse, resulting in challenges that can hinder their accuracy and generalizability. This thesis delves into the evolution of scene understanding through deep learning, and presents novel approaches for specialized tasks like jersey number detection, scene graph generation, and dynamically throttleable neural networks. The developed techniques enhance the deep model's ability to learn robust and transferable representations, enabling better generalization across diverse visual domains. Both theory and experimentation will be presented.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid progress in deep learning-based computer vision has opened unprecedented possibilities in computing various high-level analytics for sports. Artificial intelligence techniques such as predictive analysis, automatic highlight generation, and assistant coaching have been applied to improve performance and decision-making for teams and players. To perform any high-level analysis from a game match, collecting the locations (where) and identities (who) of players is crucial and challenging. Recognizing player jersey number in sports match video streams is a challenging computer vision task. The human pose and view-point variations displayed in frames lead to many difficulties in recognizing the digits on jerseys. These challenges are addressed here using an approach that exploits human body part cues with a Region-based Convolutional Neural Network (R-CNN) variant for digit level localization and classification. We propose a Pose-guided R-CNN [12] framework which adopts the Region Proposal Network (RPN) to perform anchor classification and bounding-box regression over three classes: background, person and digit. The person and digit proposals are geometrically related and fed to a network classifier. Subsequently,

it introduces a human body key-point prediction branch and a pose-guided regressor to get better bounding-box offsets for generating digit proposals. We further propose a universal JErsey number DEtector (JEDE) [13] for player identification that predicts players' bounding boxes and keypoints, along with bounding boxes and classes of jersey digits and numbers in an end-to-end manner. Instead of generating digit proposals from pre-defined anchors, JEDE predicts more robust proposals guided by players' features and pose estimation. Moreover, a dataset is collected from soccer and basketball matches with annotations on players' bounding boxes and body keypoints, and jersey digits' bounding boxes and labels. Extensive experimental results and ablation studies on the collected dataset show that the proposed method outperforms the state-of-the-art methods by a large margin. Both quantitative and qualitative results also demonstrate JEDE's superior practicality and generalizability over different sports. Our frameworks outperform all existing models on jersey number recognition task. This work will be essential to the automation of player identification across multiple sports, and releasing the dataset will ease future research on sports video analysis.

To understand a scene, it is important to infer underlying properties of entities and the relationships between them. Scene Graph Generation (SGG) has achieved significant progress recently. A scene graph is an explicit graph representation for modeling a visual scene, where entities are the nodes, and pairwise relationships are represented as edges. However, most previous works rely heavily on fixed-size entity representations based on bounding box proposals, anchors, or learnable queries. As each representation's cardinality has different trade-offs between performance and computation overhead, extracting highly representative features efficiently and dynamically is both challenging and crucial for SGG. We present a fully convolutional scene graph generation (FCSGG) model [7] that detects objects and relations simultaneously. FCSGG is a conceptually elegant and

efficient bottom-up approach that encodes objects as bounding box center points, and relationships as 2D vector fields which are named as Relation Affinity Fields (RAFs). RAFs encode both semantic and spatial features, and explicitly represent the relationship between a pair of objects by the integral on a sub-region that points from subject to object. FCSGG only utilizes visual features and still generates strong results for scene graph generation. We then propose a novel architecture called RepSGG [14] to address the aforementioned challenges, formulating a subject as queries, an object as keys, and their relationship as the maximum attention weight between pairwise queries and keys. With more fine-grained and flexible representation power for entities and relationships, RepSGG learns to sample semantically discriminative and representative points for relationship inference. Moreover, the long-tailed distribution also poses a significant challenge for generalization of SGG. A run-time performance-guided logit adjustment (PGLA) strategy is proposed such that the relationship logits are modified via affine transformations based on run-time performance during training. This strategy encourages a more balanced performance between dominant and rare classes. Experimental results show that FCSGG and RepSGG achieve the state-of-the-art or comparable performance on the Visual Genome and Open Images V6 datasets with fast inference speed, demonstrating the efficacy and efficiency of the proposed methods.

Conditional computation for deep neural networks reduces overall computational load and improves model accuracy by running a subset of the network. In this work, we present a runtime dynamically throttleable neural network (DTNN) [15] that can self-regulate its own performance target and computing resources by dynamically activating neurons in response to a single control signal, called *utilization*. We describe a generic formulation of throttleable neural networks (TNNs) by grouping and gating partial neural modules with various gating strategies. To directly optimize ar-

bitrary application-level performance metrics and model complexity, a controller network is trained separately to predict a context-aware utilization via deep contextual bandits. Extensive experiments and comparisons on image classification and object detection tasks show that TNNs can be effectively throttled across a wide range of utilization settings, while having peak accuracy and lower cost that are comparable to corresponding vanilla architectures such as VGG, ResNet, ResNeXt and DenseNet. We further demonstrate the effectiveness of the controller network on a throttleable 3D convolutional networks (C3D) for video-based hand gesture recognition, which outperforms the vanilla C3D and all fixed utilization settings.

Besides the topics mentioned above, we also explore using deep learning techniques to solve real-world problems in specialised scene understanding tasks, like early wildfire smoke detection [16], human embryonic stem cell classification [17], and video-language foundation models [18].

Chapter 2

Pose-Guided R-CNN for Jersey Number Recognition in Sports

2.1 Introduction

Broadcast sports are one of the most watched and studied videos in the world. Game analysis is performed in real time by professional commentators and videos are often recorded for coaching purposes. Analysis requires the review of thousands of hours of footage over the course of a season, and requires tasks that are impractical to be performed by human observer. Therefore, the automation of analysis is especially important. Tasks such as player detection, tracking, identification, as well as generation of game synopses, can be automated using computer vision algorithms to gather comprehensive sports match information without ever having to watch a minute of game video. Automated sports video analysis enhances the broadcasting experience for both the narrator and audience by providing auxiliary information of player's location and identity at each time point.

Match statistics from video analysis can be provided directly to coaches and players to improve strategy planning, opponent scouting, and player performance.

Figure 2.1: Illustration of two type of distractions (best viewed in color). Numbers bounded by green box are the jersey numbers of our interest while red-boxed numbers are noise. Our motivation partially comes from how to deal with various kinds of false positives.

Identifying players in sports matches is a key research challenge to make all the merits of automatic sports analysis come true. However, there are numerous problems in recognizing players in unconstrained sports video. The video resolution, viewpoint and motions of cameras, player's pose, lighting conditions, variations of sports fields and jerseys, all these factors can introduce significant challenges for automatic video analysis. Traditional approaches for player recognition in sports can be organized into two categories: identifying players via face recognition or jersey number recognition. Both approaches have their own strength and flaws. Face recognition is robust given high resolution closeup shot, while infeasible for wide shots where faces are indistinguishable or low-resolution images. Jersey number recognition can be achieved under most cases as long as

the numbers can be detected or segmented, but suffers from human pose deformation, shooting angles, motion blur, illumination conditions, *etc*. Moreover, the detection result is influenced by not only these factors but also distractions within or around the playground, such as yard markers, house numbers (illustrated in Figure 2.1), clocks, commercial logos and banners, *etc*.

Figure 2.2: Demonstration of robustness to false positives. Left: original image; middle: Faster R-CNN results; right: proposed pose-guided R-CNN results. Our framework prevents the wrong detection of the character 's' in the background.

This paper introduces a pose-guided R-CNN framework to address the challenges associated with player identification through jersey numbers. Faster R-CNN [19] is a two-stage object detector which can perform classification and bounding-box (b-box) regression, and Mask R-CNN [20] is an extension of it with predictions of segmentation masks. This work adapts and expands these concepts with re-designed region proposal and pose-guided b-box regression. The framework consists of two stages. The first stage addresses the digit-person proposal matching problem using a RPN which outputs candidate object b-boxes across three classes, background, player or digit (as opposed to vanilla RPN, which only proposes two, foreground, background). Person proposals and digit proposals are collected separately from a single RPN without adding many parameters.

The second stage uses a modification of Faster R-CNN that replaces ROIPool with RoIAlign, and includes a human body key-point branch for predicting key-point masks. The classification and b-box regression are performed on pooled digit features concatenated with key-point masks. This framework improves localization performance of digits by associating person and digit Regions of Interest (RoI), as well as adding human pose supervision signal. Consequently, the model only targets digits inside person proposals with the help from keypoint locations. An example of efficacy of our framework is illustrated in Figure 2.2.

The rest of the chapter is organized as follows. Section 2.2 introduces the background of jersey number recognition and related research. Section 2.4 discusses the framework in details. Section 2.5 evaluates several models with the introduced dataset and other wild images from web, demonstrating the applicability across other sports. Several key conclusions are drawn in Section 2.6.

2.2 Related Work

Jersey number recognition problem: The problem of interest can be considered as the combination of person identification and digit recognition problem in the context of sports broadcast videos. Traditional approaches before the dominance of deep learning usually first build an Optical Character Recognition (OCR) system then classify numbers based on segmentation results. Šari *et al*. [21] introduce a complete OCR system to segment images in HSV color space with heavy pre-processing and post-processing. Ye *et al*. [22] combine tracking information of frames and a OCR system to predict jersey number based on voting. Lu *et al*. [23] take the person localizations of deformable part model (DPM) detector then performs OCR and classification with matching

templates. These OCR-based methods have limited flexibility and robustness dealing with larger datasets. Switching to deep learning approaches, Gerke [4] designs a neural network for jersey number recgonition on small number-centered jersey images. A recent work from Li *et al*. [24] embed Spatial Transformer Network (STN) modules [25] into a CNN architecture to localize jersey number more precisely and trains the network with additional manually-labeled transformation quadrangles in a semi-supervised fashion.

Some works take sports field into considerations. Delannay *et al*. [26] formulate ground plane occupancy maps from multi-views detection to perform localization, followed by a OCR system and multi-class Support Vector Machine (SVM). Gerke *et al*. [27] consider the player recognition problem as a classifier fusion of players' positional features and jersey number convolutional neural network (CNN) ones [4]. These works put strong assumptions on the hidden pattern of player's movement and mapping of real-world and image coordinates of players. These assumptions are neither well-constructed nor universal applicable.

The jersey number recognition problem can be formulated as person re-identification (ReID) as well. Some approaches favor performing player identification directly. Lu *et al*. [28] use handcrafted combination of features to create a player representation model, then builds a L1-regularized logistic regression classifier [29] for classification, and a Conditional Random Field (CRF) graphical model to predict unlabeled videos. Lu *et al*. [30] continue the work by introducing homography estimation and a weakly-supervised learning to reduce the labor of manual annotation via auxiliary text log information of game matches. Senocak *et al*. [31] tackle player identification problem by constructing a fused feature of multi-scale features extracted from whole body image and pooled features from body parts. We also consider player identification important since the

jersey number features are highly correlated to human body ones.

Scene Text recognition: Regarding this similar research, Poignant *et al*. [32] propose a video OCR system for text recognition combing audio information to perform person identification. Goodfellow *et al*. [33] tackle number sequences recognition in constrained natural images with deep neural networks. Jaderberg *et al*. [34] proposed a complete text recognition system for natural scene images with heavily-engineered framework. [35, 36] use STNs for natural scene text detection and recognition. Bušta *et al*. [37] modify Region Proposal Network (RPN) [19] with rotation capability for better text localization. The above-mentioned literature addresses the issue of scene text being in irregular shapes which is also common but more complicated in jersey recognition problem. Jersey numbers are often distorted by player pose deformations and fast motion blur. Li *et al*. [24] adopt STN modules [25] in hope of improving localization and rectifying number distortion. However, the success of STN is built upon the fact of there being only one jersey number per image in their dataset. it is not applicable for complex scene with more involved people.

R-CNN based approaches: With the successes of R-CNNs [38, 39, 19, 20], object detection and classification are unified with high practicality. Mask R-CNN [20] and Faster R-CNN [19] are built upon RPNs with pre-defined anchors to generate region proposals, then the features are pooled from these proposals and fed into regression and classification heads. Vanilla RPN has 3 scales and 3 ratios for each anchor, Ma *et al*. [40] extended the anchor design with rotation parameter for better text proposal alignment. Cai *et al*. [41] introduced a multi-stage Cascade R-CNN to address the issue of degraded detection performance when increasing IoU thresholds.

The main concern of recognition problem in nature scenes is: how to get robust region proposals. This work exploits the fact that locations of numbers and players are highly related, and

achieves the state-of-the-art results. Our framework represents a strong advancement in automated analysis of multi-sport videos.

2.3 Contributions of this Chapter

- The RPN has been re-designed to better fit the jersey number recognition problem. The RPN outputs three classes, i.e., ”background”, ”person” and ”digit”. By dividing into person and digit proposals, it is possible to match between them to jointly generate better proposals.
- A pose-guided supervision for digit bounding-box is proposed. It learns the offsets of proposals given the prediction of human body keypoints. This module is considered as the refinement of RPN proposals.
- State-of-the-art performance for the jersey number recognition task in comparison to previously established frameworks. Significantly different from previous works, ours is capable of locating and predicting multiple numbers from input images.
- A noval dataset of 3567 images that offers person and digit bounding-boxes, human body keypoints and digit masks. One or more players and digits are annotated per image. More images are being labeled, and the dataset will be made publicly available.

2.4 Approach

In this section, the jersey number recognition task is defined in details. A vanilla Faster R-CNN is replaced with a 3-class RPN and extended with additional key-point branch and human pose supervision, yielding the ”Pose-guided R-CNN” framework shown in Figure 2.3. For real-time

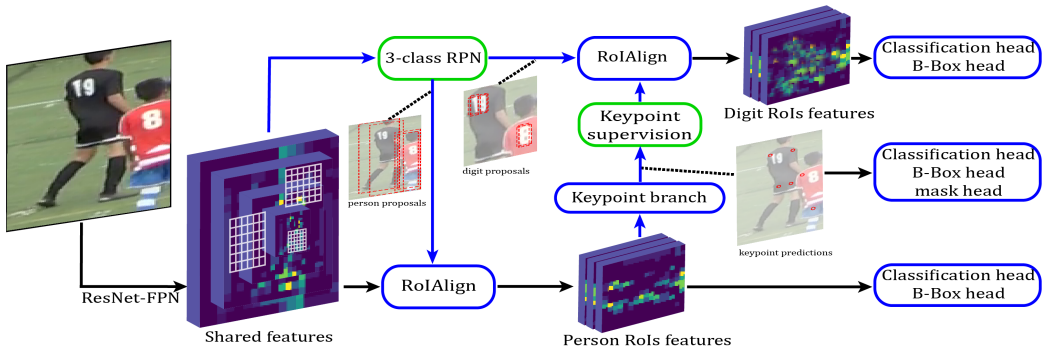


Figure 2.3: The architecture of proposed pose-guided R-CNN (feature maps are just for illustrations and not representing the actual results).

applicability, a corresponding light-weight model without sacrificing much performance that runs at 12 fps on a single NVIDIA GeForce GTX 1080 Ti GPU.

2.4.1 Task Definition

A jersey number is defined as the number worn on a player's uniform in order to identify and distinguish players. In our work, only numbers on the back are considered where player's jersey number is typically printed for most sports. Exact one number is associated to one player, and there can be multiple digits in a jersey number. Consider the input image to the model is a image in which at least one player presents with visible and recognizable jersey number. The task is to predict any human-recognizable digit instance $[0, ..., 9]$ displayed in the image. While this task has been modeled as an exact number classification problem [4] as well as a number length prediction problem [24], this work models the task as a 10-digit classification problem.

2.4.2 RoIAlign Faster R-CNN

From previous task definition, region-based methods are extremely suitable for our problem. One of the successful architectures is Faster R-CNN. It consists of a backbone feature extractor, a Region Proposal Network followed by a feature pooling module, and network heads for b-box regression and classification for each RoI. For an image, shareable convolutional (Conv) features are extracted first with choices of backbone architectures such as VGG-16 [42], ResNet [2] and ResNeXt [43] then the RPN generates a set of reference boxes (anchors) from an image of any size. For each pixel location, there can be arbitrary number of anchors given different scales and aspect ratios. A sliding network will traverse each pixel location and tries to predict if an object exists in the corresponding anchor and regress the b-box from shared features. After the proposals are generated, the pooled features for each RoI will be fed into the fully connected layers to perform detection and classification. Feature extraction from each RoI is done with RoI max pooling (RoIPool) such that a $h \times w$ Conv feature map is divided into numbers of $h/H \times w/W$ sub-windows then max-pooling is performed for each grid with quantization. For each detected b-box, non-maximum suppression (NMS) is used to filter out similar and close b-boxes.

Some modules are improved by Mask R-CNN. First it incorporates the Feature Pyramid Network (FPN) [3] with the backbone to generate multi-scale features. It then replaces RoIPool with RoIAlign which interpolates the sampled feature for better alignment between RoI and input feature maps. Beside, it adds an extra branch to generate object masks in parallel in addition to classification and b-box regression. The output mask is represented as a $m \times m\ px$ binary mask from each RoI without losing the spatial layout of convolutional features. For additional details, we refer interested readers to [19, 3, 20]. Faster R-CNN is referred to this improved implementation

unless specified.

The loss is defined as a multi-task loss both for final prediction and RPN proposals:

$$L = L_{cls} + \lambda L_{reg}, \tag{2.1}$$

where L_{cls} is classification loss, L_{reg} is the b-box regression loss, and λ is the multi-task balance weight. We consider each digit from 0 to 9 as a class, a 'person' class and a 'background' ('BG') class, in total of $K = 12$ independent classes. Ground-truth class is denoted by u where $u_{BG} = 0$ by convention. For each RoI, the output layer will produce a discrete probability distribution $p = (p_0, p_{K-1})$, then the class loss is define as log loss for true class

$$L_{cls}(p, u) = -\log p_u. \tag{2.2}$$

The localization loss is defined as

$$L_{reg}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i), \tag{2.3}$$

' where $u > 0$ ('BG' class does not contribute to the loss), and t_i^u is predicted bounding-box offsets four-tuple (x, y, w, h) for class u. (x, y) is the top-left corner coordinate, (w, h) is the predicted dimension of the b-box. $v = (v_x, v_y, v_w, v_h)$ as the ground-truth b-box. $smooth_{L_1}$ is a robust L_1 loss against outliers defined in [39].

2.4.3 Proposal Association

Up to this point, we have generated proposals of either one digit or a person and same for final detections. To collect the final results in terms of jersey numbers, we reduce our problem into a graph matching problem [44] with some relaxations. Nodes of the graph are the person and digit proposals, and the edges are all possible connections between pairs of person and digit

Figure 2.4: Some examples from the dataset. (g), (k), and (l) are examples of multi-jersey annotations; (a), (f) and (g) are illustrations of jersey numbers under common conditions; (c) and (h) exhibit contrast lighting conditions ; (i) shows a close-view image where number aspect ratio is distorted; (b), (d) and (j) are examples of numbers influenced by pose deformation; (e) is highly distorted but still recognizable.

proposals. The weight of each edge is computed by the Euclidean distance between the two centers of bounding boxes. And for each person node, there must exist k edges matched with digit nodes, where $1 \leq k \leq 2$. So each person node can be matched with up to two other digit nodes which is not necessarily bipartite matching. The problem is then solved by choosing the top-2 digit proposals for each person proposal.

2.4.4 Three-class Region Proposal Network

The original RPN only estimate the probability of each proposal being an object or not. It takes shared features to perform classification and bounding-box regression of anchors. Our motivation is simple: instead of just 2 classes, this work uses 3 classes to represent 'BG', 'person' and 'digit' by adding very few parameters. In this way, anchors are not treated independently. Anchors are divided into person and digit anchors that are then correlated by their spatial relationships.

No modifications are made to the pre-defined anchor settings in [19] that there are lots

of overlaps among anchors. Each anchor is actually associated with many other anchors in terms of location. For example, if an anchor is of scale 512, some anchors of scale less than 512 will be inside it. The proposal scheme is modified to accommodate this anchor association. For training vanilla RPN, each positive anchor is assigned based on two criteria. The following conditions are provided along with three-class RPN:

- Anchors with the highest Intersection-over-Union overlap with certain ground-truth box.
- Person anchors with IoU higher than 0.7.
- Digit anchors with IoU higher than 0.7 and inside any person anchor.

After filtering and assignment of anchors, we associate each digit anchor to its closest person anchor based on Euclidean distance between centers of the two boxes.

2.4.5 Pose-guided Supervision

Mask R-CNN can also perform human body keypoints estimation as stated in [20]. Similar to the binary mask representation of objects, each body keypoint is modeled as an object except that there is only one pixel labeled in the mask. For K types of keypoints, *e.g*. right shoulder, left hip, *etc*., there are K individual one-hot masks. Human body modelling is not required in Mask R-CNN framework to achieve fair results. In the case of jersey number recognition, it is reasonable and achievable to perform jersey number localization better given body keypoints layouts. Though Faster R-CNN is capable of bounding-box regression for jersey numbers, there are limitations under more sophisticated scenarios. For example, complex jersey patterns, different number fonts, and numbers on the court introduce difficulties for RPN to generate satisfactory proposals. To tackle the problem, a pose-guided supervision branch is proposed for refining number localization.

A keypoint branch for predicting key-point mask is added similar to [20, 45]. The key-point detection is only applied on person RoIs. At this point, each person RoI is associated with multiple digit RoIs as a result of three-class RPN. The keypoint mask is fed into a shallow network to obtain b-box offsets with which we can correct the RPN proposals. Features of refined proposals are then pooled via ROIAlign. It involves a transformation from keypoint locations to associated digit b-box regression in a hidden space. Finally, a digit branch is formulated that is responsible for digit recognition on refined RoIs. This cascade design provides digit RoIs with more information outside their regions.

The proposed pose-guided network takes predicted keypoints mask from each person RoI as inputs, and output the b-boxes offsets of corresponding jersey numbers. It is a small but effective network consisting of three fully connected layers.

The loss function 2.1 can be modified accordingly by adding related keypoint classification and regression loss $L_{cls}^{keypoint}$, $L_{reg}^{keypoint}$. Then the regression loss for digit b-box is computed from the RoI refined by keypoint mask. The final loss function is

$$L = L_{cls} + \lambda L_{reg} + \eta\lambda L_{cls}^{keypoint} + \gamma\lambda L_{reg}^{keypoint}, \tag{2.4}$$

where η and γ are hyper-parameters similar to λ.

2.5 Experimental Results

The proposed pose-guided R-CNN, as well as related models are evaluated on the collected dataset, since there is no publicly available dataset on jersey numbers. The evaluation metrics used are standard Average Precision (AP) with IoU thresholds set to 0.5 and 0.75, and AP average (mAP) across IoU from 0.5 to 0.95. Number-level and digit-level accuracies are also reported.

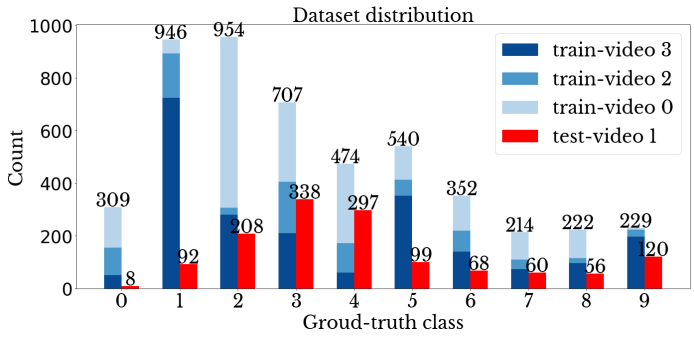


Figure 2.5: Distribution of digits in the data.

2.5.1 Dataset

The dataset is gathered from four full soccer matches. The recording device used is a single Canon XA10 video camera which is installed 15 feet high, and 10 to 20 feet away from the horizontal baseline of the soccer field. For better video qualities in terms of recognizable jersey numbers, the camera operator is allowed to pan and zoom accordingly. Next, we convert the collected videos into frames by two different ways. One is to perform a human detector over frames scaled by 2 to get reliable images containing players. OpenPose [46] is used for person detection. In order to collect more difficult images, Random shifts and paddings are added to detected areas. The detection results are padded by 150px and a random shift of 20px. After data collection was complete, two professional annotators labeled any legible jersey numbers via VGG Image Annotator [47]. As a result, there are arbitrary number of ground-truths (GT) per person per image.

A total of 3567 images are annotated with ground-truth (GT) digit masks resulting in 6293 digit instances, see the distribution in Figure 2.5. All images are also labeled with person bounding-boxes and four human body key-points, namely left shoulder (LS), right shoulder (RS), left hip (LH) and right hip (RH). There are 114 images contain multiple numbers, and each digit is

Table 2.1: Dataset statistics. H, W, h and w are image height, image width, digit b-box height, and digit b-box width respectively. For heights and widths, the unit is pixel; mask area counts the number of pixels on the object; mask center is normalized within range [0, 1].

	H	W	h	w	Digit mask area	Digit mask center
Mean	315.06	214.53	34.70	18.90	424.40	(0.50, 0.29)
Std	92.11	38.47	15.16	7.85	20.69	(0.12, 0.09)

labeled with its associated person box. Figure 2.4 shows a few examples for our dataset. Dataset statistics are illustrated in Table 2.1.

Bounding-box sizes are sorted into small (area $< 32^2$), medium ($32^2 <$ area $< 96^2$) and large (area $< 96^2$) objects like COCO dataset [48]. For person b-boxes, there are 4111 large, 213 medium and 1 small objects; for digit ones, there are 7 large, 1210 medium and 5076 small objects.

2.5.2 Implementation Details

The hyper-parameters in the loss function 2.4 are all set to one. All the experimented models make use of image augmentation technique by applying random affine transformation and hue/saturation manipulation to both original image and corresponding b-box. The backbone feature used in all experiments is ResNet-FPN. We use ResNet features at 4 different stages $[C2, C3, C4, C5]$ to build the feature pyramid. The constructed RPN features are $[P2, P3, P4, P5, P6]$. The light-weight model removes $C5$ and $P6$. For RPN anchors, 5 scales $[32, 64, 128, 256, 512]$ and 3 ratios $[0.3, 0.5, 1]$ are used. For the classification network $head$, $P6$ is not used as input. Partial implementation is adopted from [49].

Person and keypoint branches: The settings for person branch are same as described in

Table 2.2: Comparison of results among approaches. Our method achieves the best accuracy (ACC) for both number-level and digit-level recognition. Input is cropped grayscale image for Gerke's [4], and original RGB image for all other approaches.

Framework	Backbone	Input	ACC_{number}	ACC_{digit}
Gerke[4]	-	40^2	65.04%	-
Li *et al*. [24]	-	200^2	74.41%	77.86%
Li *et al*. [24]	ResNet-50	512^2	77.55%	80.23%
Faster R-CNN	ResNet-FPN-50	256^2	86.13%	89.32%
Faster R-CNN	ResNet-FPN-50	512^2	88.74%	90.09%
Faster R-CNN	ResNet-FPN-101	512^2	89.02%	91.11%
Pose-guided (Ours)	ResNet-FPN-18	512^2	81.66%	83.97%
Pose-guided (Ours)	ResNet-FPN-50	256^2	90.84%	92.13%
Pose-guided (Ours)	ResNet-FPN-50	512^2	91.01%	93.29%
Pose-guided (Ours)	ResNet-FPN-101	512^2	**92.14**%	**94.09**%

[20]. The keypoint branch is based on mask prediction in Mask R-CNN, except that the keypoint mask is up-sampled to 32×32.

Digit branch: The pose-guided supervision module consists of two 512 Fully-Connected (FC) layers, and a $N \times 4$ FC layer with $linear$ activation as digit b-box regression $head$. N is the number of proposals, so it outputs the b-box offsets for each digit RoI. The rest of the branch resembles person branch except for the pooling size to be 16×16 in digit classification $head$. It gives better performance since digits are relative small in images.

Different settings including but not limited to changing the backbone features, input image size, image operations (re-sizing, padding, cropping, *etc*.), number of image channels are used in experimentation. ResNet-FPN-18, ResNet-FPN-50 and ResNet-FPN-101 with/without proposed pose-guided module are investigated. For collecting convincing results, the dataset is divided video-

wisely, with video 0, 2, 3 for training and video 1 for testing.

Pre-train: To accommodate the lack of person-keypoint data in the collected dataset, the network is pre-trained on the COCO dataset [48] with a frozen digit branch. In this dataset, 17 human body keypoints are annotated, but four of them are used for less parameters and better convergence. Person and keypoint branches are then unfrozen, and the digit branch is trained with Street View House Number (SVHN) dataset [33]. This large-scale dataset consists of digit sequences with each digit labeled with bounding box. The model benefits from this dataset for training the backbone feature extractor.

Training: The model is trained for 100 epochs with a starting learning rate (LR) of 0.01. Learning rate is reduced by 10 every 20 epochs. The rest hyper-parameters are same with Mask R-CNN [20].

Testing: The settings are the same as training except that 100 detections are kept.

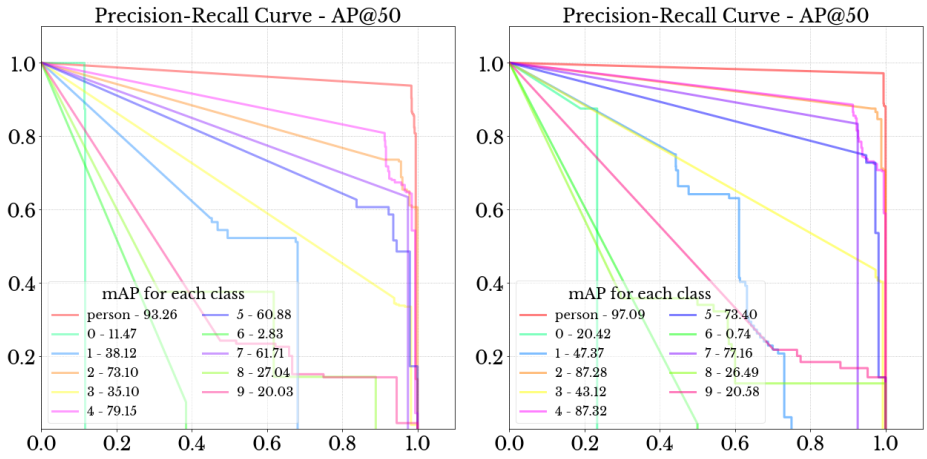


Figure 2.6: Precision-Recall Curves over each class. Left figure shows the Faster R-CNN results; right one shows ours results with improvement.

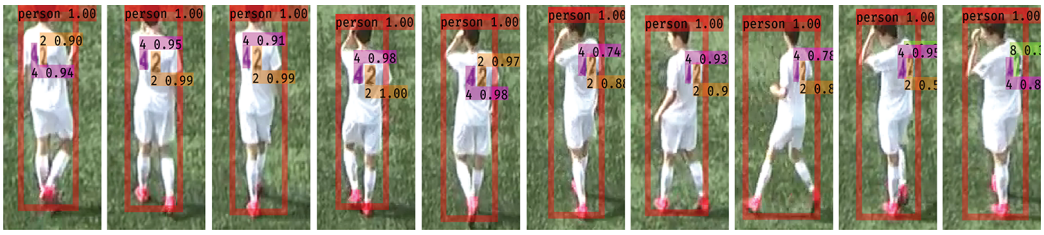


Figure 2.7: Recognition results across different poses. Most-left and most-right poses are extreme cases in our test set for this identity.

2.5.3 Main Results

The proposed model is compared to available methods in the field of jersey number recognition, see Table 2.2. All variants of our model outperform previous state-of-the-art models including Gerke [4] and Li *et al*. [24]. These two approaches can only perform image-level recognition. For fair comparison, multi-number images are removed during training and testing. Each image is grayscale, cropped and re-sized to 40×40 in accordance with [4]. Without access to the dataset of [24], this architecture is implemented without axis supervision. Its variant with ResNet-50 is also implemented. Faster-RCNN is also a strong baseline which already outperforms [4, 24]. The proposed model achieves even better performance that is highly robust to post variations. Figure 2.7 visualizes the recognition results against different poses. We evaluate both digit-level and number-level accuracies for our model and [4, 24]. The results are illustrated in Table 2.2.

Evaluation metrics including number-level and digit-level accuracies, mean average precision (mAP), AP_{50}, and AP_{75} are used to compare variants of the R-CNN approaches. APs for different object scales are not used since most 'person' boxes are large and most 'digit' are small. The results are shown in Table 2.3. The proposed pose-guided R-CNN gives the best overall results.

Table 2.3: Comparison of Faster R-CNN and our pose-guided R-CNN results. The backbone used is ResNet-FPN-50, and input image size is 512×512.

Method	ACC_{number}	ACC_{digit}	mAP	AP_{50}	AP_{75}
Faster R-CNN	87.23%	89.04%	40.60	67.21	45.58
Pose-guided (Ours)	90.44%	93.12%	44.74	73.31	48.77

2.5.4 Ablation Study

In this section, we only consider ResNet-FPN-50 as our backbone given several reasons: it has around $19M$ less parameters; we have a small dataset so ResNet-50 is more suitable; we did not fine-tune the models so better performance can be achieved through regularization. Therefore, we choose ResNet-FPN-50 over ResNet-FPN-101 without sacrificing much performance. Multi-number images are included for experiments in this section.

Input size: To build feature pyramid for ResNet-50, we need to resize the image so that its width and height can be divided by 2 at least 5 times. We need the image size to be large enough since the numbers in the dataset are mostly small objects. For simplicity, we re-size to square image with paddings while keeping the aspect ratio. We did experiments with several sizes: 128, 256, 512, 1024. When the input size is 512, it achieves the best performance of mAP 44.74, which outperforms 10.20, 3.12 and 0.56 points with respect to size 128, 256, and 1024.

Does 3-class RPN solely help: With the baseline of Faster R-CNN, we want to evaluate if replace the vanilla RPN with our 3-class RPN help improve the performance. We use image size of 512×512 as input, and ResNet-FPN backbone for this experiment's settings. Three-class RPN has -0.09, 0.12 and -0.14 gain respectively over vanilla RPN on mAP, AP_{50}, and AP_{75}. Both

give similar experimental results, so it suggest that by just switching to three-class RPN, the performance is not significantly influenced. RPN is a shallow 'neck' network for anchor classification and regression. Splitting 'object' class into 'person' and 'digit' does not introduce hardness for these two tasks, but we can not guarantee multi-class RPN will work for more classes. The key function of our three-class RPN is dividing then matching person and digit anchors. If the following structure remains the same with Faster R-CNN, the results are expected to be similar. However, as we already match the anchors in three-class RPN, the proposal association procedure for number-level prediction can be removed.

Pose-guided R-CNN: Table 2.3 suggests that there is 4.14 gain over Faster R-CNN. We also report AP_{50} for each class for these two models illustrated in Figure 2.6. It shows significant improvement achieved by adding pose supervision which has a keypoint mAP of 58.2. The reason of poor performance on '0' is that We have very few images contain '0' in test dataset, so it drops drastically even if only one of them is classified incorrectly. Figure 2.7 provides recognition results of our pose-guided R-CNN model against different poses. However, there are still some limitations under extreme poses as the last two examples shown in Figure 2.7. For testing our model's generalization, We also collected some images from internet videos for different sports: basketball, American football and hockey. The results are illustrated in Figure 2.8. Fair detection results are still obtained, but classification performance is reduced. Recognition is possibly simpler for soccer and basketball due to plain jerseys, while jerseys in American football and hockey are normally bulky with sharp contours. Better performance can be achieved by gathering more data.

Figure 2.8: Exemplary results from wild images collected from internet. Fails: '7' is classified wrongly as '2' in second last image; '6' in the last image is classified wrongly as '5'.

2.6 Conclusion

In this work, a pose-guided R-CNN multi-task framework is proposed as an all-in-one solution for person detection, body keypoints prediction and jersey number recognition. It produces the best digit accuracy of 94.09% comparing with related literature. Three insights are used to achieve this performance: 1. re-designed three-class RPN for anchor association; 2. implementation of pose-guided localization network that can impose proposal refinement for jersey number location through human pose; 3. the generality of region-based CNN model. By combining the

three components, the proposed approach is end-to-end trainable and can be easily extended to other sports.

Chapter 3

JEDE: Universal Jersey Number Detector for Sports

3.1 Introduction

Recently, there has been a tremendous growing interest in artificial intelligence (AI) technology for sports. Every aspect of sports, from the recruitment of athletes to the analysis of performance, from game planning to injury management, from audience experience to media, is empowered by AI. Not only industry has substantially explored new technologies for sports, but also academia has dramatically strengthened the research capacity on the topic. Among various AI applications, computer vision (CV) for sports has one of the most significant potentials which may have a huge impact on the way people view and consume sports content. For example, current tracking systems [50, 51] are deployed in stadiums and collect comprehensive game data on players, referees, and the ball in real-time. The basic statistics of players' moving direction, speed, and acceleration,

and even more advanced statistics could be obtained via CV. Knowing players' locations, augmented reality (AR) can be applied in live broadcasting to provide entertainment enhancements such as ball movement diagrams, player identifications, scoring probabilities, *etc*. There are many other CV applications for sports, such as event detection [52, 53, 54], activity recognition [55], human pose estimation [56, 57, 58], human motion prediction [59, 60], automatic highlight generation [61], and image generative models [62]. Moreover, the research in CV for sports is not only about generating statistics, but also about scene understanding and human behavior analysis.

We have seen deep learning models [63] that defeat humans in many games like Go, Chess, and Atari. These games serve as perfect simulators for learning. Analogically for real-world applications, sports games are the perfect simulation environment for scene and human behavior understanding. Tuyls *et al*. [64] propose three foundational areas associated with soccer AI research: statistical learning, computer vision, and game theory. Computer vision models provide the complementary high-level and spatially-detailed features for the other two areas, while benefit from low-dimensional game-related statistics and metadata from them. Shih [65] proposes the content pyramid for sports video analytics, which consists of four layers: video, object, action, and conclusion. The object layer as the second lowest level, connects the raw data processing and higher-level analysis. Undoubtedly, object detection or player identification is the most important building block for sports video analysis. Traditional methods for player identification rely on hand-crafted features [66, 28] or face recognition [67, 68, 69, 70, 71], which are infeasible for complex scenes or different fields-of-view. Researchers also try to solve the problem by detecting the jersey number since it is the generic visual cue of identity. Early approaches [22, 21, 72, 23] are based on optical character recognition (OCR) to extract and classify numbers. However, these methods are not robust

to the challenges in broadcast sports videos, such as illumination changes [73], low jersey number resolution, viewpoint and camera movements [28], players' pose deformation, occlusion, motion blur [74], and stadium distractions [12], *etc*. Deep learning has been widely applied in CV, but there are only a few papers on jersey number classification [4, 24] or player detection [75, 76, 77], not to mention end-to-end jersey number detection. Previous work [4, 24] is only applicable for single-person images to perform image classification, but not for frames consisting of multiple players.

In this paper, we propose a novel jersey number detection framework for player identification in sports videos, named as universal JErsey number DEtector (JEDE). It is a multi-stage detector, which predicts players' bounding boxes and pose estimations, with associated jersey digits' bounding boxes and classes all at once. The first stage extracts image features through a "backbone" network (*e.g*., ResNet-50 [2]) and constructs a feature pyramid [3]; then, a Region Proposal Network (RPN) [19] is used for generating player candidate proposals. The second stage extracts features using RoIAlign [20] from each player's candidate box and performs classification, bounding-box regression, and human-body keypoint regression. In parallel with these detections, we add a branch that predicts the bounding boxes of digits of the jersey number within each player's bounding box. Both the player's features and corresponding keypoint predictions are used for generating digit proposals. More specifically, we model individual digit as an object, which is represented by the center and size of its bounding box. Within each player proposal, we regress the center and size heatmaps of the digits given the extracted player features and keypoint heatmaps. By conditioning on human pose information, the localization of digits is significantly improved. We then extract features from digit proposals and perform digit classification and bounding box regression. Finally, the digit detections are paired as number detections. Our framework is performed on a per-frame basis with fast

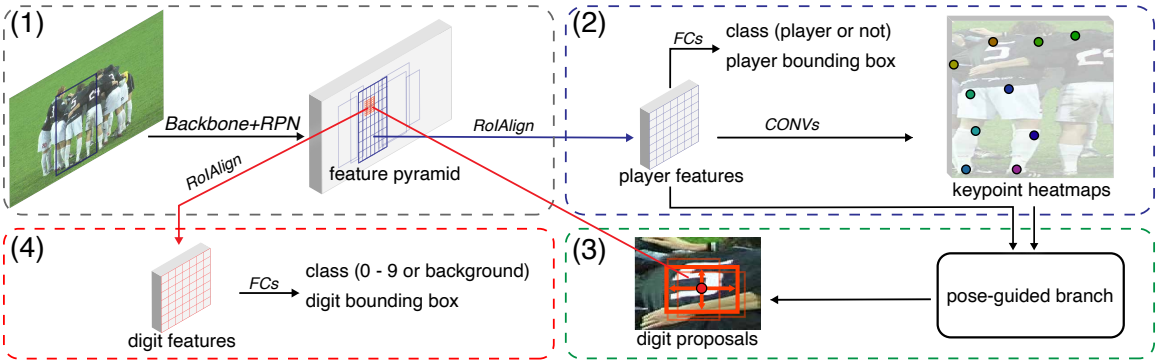


Figure 3.1: The architecture of JEDE - it can be divided into four modules: (1) a backbone network that extracts features and constructs a feature pyramid (*e.g.*, ResNet-50 FPN) followed by a RPN (Section 3.4.2.1); (2) a player branch that extracts features from player proposals generated from the RPN via RoIAlign, performs classification, bounding-box regression, and keypoints regression (Section 3.4.2.2); (3) a pose-guided branch that predicts digit proposals from the pooled player's features and corresponding keypoint heatmaps (Section 3.4.3); (4) a digit branch that extracts features from digit proposals, and then performs digit classification and bounding-box regression (Section 3.4.4.1). Fully-connected layers are denoted by FCs, and convolutional layers by CONVs.

inference speed, and no motion information is used. Besides the novel architecture, two data augmentation techniques called `CopypasteMix` and `SwapDigit` are proposed. `CopypasteMix` creates new training data by copying and pasting among images, while `SwapDigit` by swapping digit instances with data from other datasets such as Street View House Number (SVHN) [33]. This paper significantly extends our previous work [12] by re-designing the architecture, proposing new data augmentation methods, and providing more experimental results and ablation studies.

3.2 Related Work

This work is mainly focused on jersey number detection, which is also highly related to many general vision tasks such as person re-identification (Re-ID) [78], object detection [19, 79], multi-object tracking (MOT) [80], and scene text detection [81, 82, 83]. Reviewing all the related literature is beyond the scope, thus we only discuss the most relevant research on sports analysis.

3.2.1 Player Detection and Tracking

Player detection and tracking are important techniques that are required for sports video analysis, providing the spatial and temporal information about players. Player detection is the preliminary step for player identification that generates bounding boxes of players, while player tracking associates the bounding boxes between frames and assigns a tracking ID for each bounding box. Traditional methods rely on hand-crafted features. For example, Lu *et al*. [23, 30] use the deformable part model (DPM) for player detection and then perform player classification based on handcrafted features and tracking information. Gerke [84] *et al*. augment Histogram of Oriented Gradients (HOG) features [85] with jersey color information to improve player detection performance. Modern deep-learning approaches adopt off-the-shelf object detectors and their variants for player detection [86, 87, 88, 89]. For player tracking, off-the-shelf multi-object tracking algorithms are commonly used [90, 91, 92, 93]. Sentioscope [94] is one example of such a system that maps the image to a modeled soccer field, performs player detection, builds a likelihood model based on appearance and motion, classifies teams based on jersey colors, and assigns identity tags to tracks.

3.2.2 Jersey Number Recognition

Jersey number recognition can be considered as the task of person identification (ID) in the context of sports broadcast videos where each player's ID is uniquely associated with the jersey number. Player identification can be performed directly based on the player's appearance or pose features [30, 31, 95, 96, 93], but re-training is required if the match roster or target sport changes. Jersey number recognition provides a relatively more general and robust solution to player identification. Most approaches can only perform jersey number recognition on images that only contain

a single player. Traditional approaches before the dominance of deep learning usually first build an OCR system, and then classify numbers based on segmentation results. Šari *et al*. [21] introduce an OCR system to segment images in HSV color space with heavy pre-processing and post-processing. Ye *et al*. [22] combine tracking information of frames and a OCR system to predict jersey number based on voting. These OCR-based methods have limited flexibility and robustness on real-world data. Switching to deep learning approaches, Gerke [4] designs a neural network for jersey number recognition in cropped jersey number images. Li *et al*. [24] propose a framework that adopts Spatial Transformer Network (STN) [25] to refine jersey number features automatically, which is trained with additional labeled transformation quadrangles in a semi-supervised fashion. Some work takes the given sports field into consideration: Delannay *et al*. [97] create ground plane occupancy maps from multi-view detections to perform localization, followed by an OCR system with a Support Vector Machine classifier; Gerke *et al*. [27] combine the players' spatial constellation features and jersey number features from CNNs to achieve better per-game player recognition performance. These work make strong assumptions on the hidden pattern of player's movement and accurate inverse homography, which is not practical or generalizable for other sports.

3.2.3 Jersey Number Detection

There is limited work on jersey number detection due to significant challenges like human pose deformation, camera view changes, motion blur, and various illumination conditions. Traditional OCR-based methods [22, 21] can only perform single jersey number detection on single-player images with close-up views. Our previous work [12] explores deep-learning-based multi-player multi-digit jersey number detection, and proposes a pose-guided R-CNN that still has some limitations. It requires associations between player and digit bounding boxes, where wrong associa-

tions may occur in crowded scenes. It does not work well on images with a wider field-of-view due to insufficient training data. In this paper, JEDE addresses these challenges and limitations. The major extension over [12] lies in the pose-guided branch and data augmentation. The re-designed pose-guided branch directly predicts digit proposals from each player proposal instead of using RPN, so no association of bounding boxes is needed. It provides more accurate and robust digit proposals based on the player's features and keypoints. The proposed data augmentation strategies `CopypasteMix` and `SwapDigit` introduce more training data variations that significantly alleviate the problem of limited data as compared to [12]. As a result, the proposed framework JEDE achieves the state-of-the-art results and outperforms pose-guided R-CNN by a large margin. The contributions of the paper are summarized in Section 3.1.

3.3 Contributions of this Chapter

1. We tackle the player identification problem via jersey number detection that is more robust to real-world variations. We propose the first framework that can simultaneously predict players' bounding boxes, pose estimations, jersey digits' and numbers' bounding boxes and classes. The rich predictions provided by our framework are significant for higher-level analysis.

2. Unlike previous jersey number recognition frameworks, jersey number detection addressed in this paper is a challenging multi-player multi-digit detection problem. Our proposed model JEDE generates jersey number detections from instance-level digit localization and classification, which is much more accurate and reliable.

3. We collect a dataset consisting of 4477 images from soccer and basketball matches. There are 6054 labeled players with 5406 labeled human body pose, and 6293 labeled digits. More-

over, we propose data augmentation strategies named `CopypasteMix` and `SwapDigit` that effectively improve detection performance and robustness. We also explore pre-training on COCO [48] and SVHN [33] datasets, which further improve the results.

4. We conduct comprehensive evaluations, ablation studies, and comparisons of the proposed framework with the state-of-the-art methods for jersey number recognition, object detection, and scene text detection on the collected dataset. We also show that the proposed method is easily generalized on wild images across different sports with superb performance.

3.4 Technical Approach

R-CNN and its variants [98, 39, 19, 20, 99] are flexible, general, and extensible for many computer vision tasks, such as object detection, instance segmentation, human pose estimation, and panoptic segmentation. This flexibility provides more capabilities for sports analysis that involves more complex scene dynamics. In this section, we explain our overall framework and individual modules of our proposed method.

3.4.1 Problem Setup

A jersey number is defined as the unique number on the player's uniform to identify players. In our work, only the number printed on the back is considered since it typically exists for most team sports. As a jersey number consists of a sequence of at most two digits in most sports [100], we only consider detecting the number with a maximum length of two digits. The task is then to predict the bounding box and class of any visible and recognizable digit instance on the back of the jersey in an image. We formulate jersey number detection as a multi-step approach:

player detection, digit detection, and jersey number detection. Player detection is based on two-stage Faster R-CNN; digit detection is a top-down approach performed on each Region-of-Interest (RoI) of detected players; we then generate jersey number candidates based on the predicted digits. For jersey number recognition, [4, 24] simply treats it as a number classification task, while our approach performs per-instance digit classification and association within each player's RoI.

The overall architecture of JEDE is presented in Fig. 3.1. Inspired by Mask R-CNN [20], the framework consists of four main components: a feature pyramid network (FPN) [3] as the backbone, followed by a region proposal network (RPN) [19] for generating player proposals; a player branch for player/background classification, bounding boxes regression, and pose estimation; a pose-guided branch for generating digit proposals; a digit branch for digit classification and bounding boxes regression. The final jersey numbers are generated from digit detections as a post-processing step which will be discussed in Section 3.4.4.2.

3.4.2 Backbone, RPN and Player Branch

3.4.2.1 Backbone and RPN

Similar to scene text detection, jersey number detection is challenging because of varying sizes and fonts of jersey numbers in sports broadcasting. The scale of a player changes with the change of the camera and its viewpoint. Therefore, the scale of jersey numbers also changes in a wide range. To capture high-level semantic features at all scales, a feature pyramid [3] is constructed from ResNet [2] features. RPN is used to generate player proposals for the subsequent player and pose-guided branches. We use 5 scales of anchors $\{32, 64, 128, 256, 512\}$, and 3 aspect ratios $\{0.5, 1, 2\}$ following Faster R-CNN [19, 3]. As shown in section 3.5 later, we achieve similar results with

faster inference speed by removing the anchor size of 32.

3.4.2.2 Player Branch

The player branch includes three tasks: binary classification (player s background), bounding box regression, and keypoints regression. Given the player proposals from RPN, RoIAlign [20] is used for extracting features. We keep the same Mask R-CNN [20] heads (small prediction networks) with pre-trained weights for faster convergence, where the pooling size is 7×7 (pixels in feature maps) for classification and bounding box regression, and 14×14 for keypoints regression. For human body keypoint detection, we predict a mask of shape $17 \times 56 \times 56$ for each player RoI, where there are 17 types of person keypoints following the COCO dataset [48], and the output feature side length is 56. Please refer to Mask R-CNN [20] for more details.

3.4.3 Pose-Guided Branch

For jersey number detection task, previous work [12] has demonstrated that better jersey number localization can be achieved given human pose information. Though Faster R-CNN is capable of regressing jersey number or digit bounding boxes directly, there are limitations under more difficult scenarios. For example, varying jersey patterns, fonts, hash marks, and commercial banners introduce difficulties in generating satisfactory proposals for RPN. To tackle these problems, we introduce a pose-guided branch for refining digit localization conditioned on the player detection and pose estimation. Each digit proposal is generated based on the regressions of its center and bounding box size within a player proposal. We also provide a theoretical analysis on why the human pose helps in digit localization in the supplementary material.

3.4.3.1 Design

We consider a single player proposal for illustration. Given the player's bounding box predicted from the player branch, the player features are pooled from the feature pyramid as one input to the pose-guided branch. Another input is the regressed keypoint heatmaps. Since the feature dimensions may be different, we use small fully convolutional networks (FCNs) for adjusting feature dimensions. Specifically, a convolution kernel with stride of 2 is used for downsampling, and bilinear interpolation is used for upsampling, if needed. The player features generated by the FCN (two 3×3 64-channel convolutional layers by default) is denoted by F_{player}. Depending on the configurations, the spatial dimension of F_{player} may remain the same or increase to have higher-resolution features for digit localization. The keypoint heatmaps are downsampled spatially to have larger reception fields via a FCN, capturing more semantic features from the pose estimations. We name the resulting features as F_{kpts}. Both features are then fused as $F = \mathsf{fusion}(F_{\text{player}}, F_{\text{kpts}}) \in \mathbb{R}^{C \times M \times M}$, where fusion is the function that combines the two input features, C is the output number of channels, and M is the output feature side length. The feature fusion can be either concatenation, addition, or multiplication. The ablation study on fusion methods is provided in Section 3.5.6. Optionally, positional information can be considered as additional features. We adopt the extended 2D version of positional embeddings [101], and concatenate them with F_{kpts}. We can also concatenate the embeddings with F which is less effective as shown in the ablation study (Section 3.5.6).

3.4.3.2 Output and Ground-truth Generation

We then regress heatmaps for digit center, center offset, and size respectively. The fused features F will be fed into 3 FCN prediction heads, each of which consists of four 3×3 64-

channel convolutional layers. Since there are at most 2 digits for a jersey number for most of the sports, predicting two-channel center heatmaps $\mathbf{O} \in \mathbb{R}^{2\times M\times M}$ is sufficient. For a single-digit jersey number, the ground-truth (GT) center is only defined on the first channel. As for a two-digit jersey number, the left digit center is defined on the first channel, and the right digit center on the second channel. Specifically, we define the GT digit center class $d \in \{0, 1\}$ and the bounding box (x_0, y_0, x_1, y_1) where (x_0, y_0) and (x_1, y_1) denote the coordinates of the left-top and right-bottom corners. The center is computed as $\mathbf{o} = (o_x, o_y) = ((x_0 + x_1)/2, (y_0 + y_1)/2)$. We then need to map the digit center coordinates into the feature scale. Given the corresponding player's bounding box $(x_0^p, y_0^p, x_1^p, y_1^p)$, we compute the relative coordinates with respect to the player's bounding box as $(o_x - x_0^p, o_y - y_0^p)$. The feature-to-bounding-box width and height ratios are computed as

$$r_w = M/(x_1^p - x_0^p),\ r_h = M/(y_1^p - y_0^p), \tag{3.1}$$

respectively. Finally, the digit center in the $M \times M$ feature grid is

$$(o'_x, o'_y) = (r_w \cdot (o_x - x_0^p), r_h \cdot (o_y - y_0^p)). \tag{3.2}$$

For regression of the GT digit center, we quantize the coordinates, and assign the value of 1 at $(\lfloor o'_x \rfloor, \lfloor o'_y \rfloor)$ and 0 otherwise, where $\lfloor \cdot \rfloor$ is the floor function. To obtain more positive training samples, the object center will be modulated by a bivariate Gaussian distribution along the x-axis and y-axis following Law [102] and Zhou *et al.* [103]. Given the size of the bounding box $(w, h) = (x_1 - x_0, y_1 - y_0)$, the feature-scale size is $(w', h') = (r_w \cdot w, r_h \cdot h)$. Based on the feature-scale bounding box size, and the desired minimum Intersection over Union (IoU) denoted by min_iou, the Gaussian standard deviations σ_x and σ_y are derived as:

$$(\sigma_x, \sigma_y) = \frac{1}{3}(a, b) = \frac{1}{3}\lfloor \frac{1 - \sqrt{min\_iou}}{\sqrt{2}} \cdot (w', h') + 1 \rfloor, \tag{3.3}$$

where σ_x and σ_y control the spread of the distribution, and a and b are the semi-minor axes along the x-axis and y-axis respectively. The value of min_iou is chosen in the range of $(0, 1)$, such that for any point (x, y) in the ellipse region $R = \{(x, y)|\frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1\}$, when using it as a center to create a bounding box of size (w, h), the bounding box has at least min_iou IoU with the GT. Then, for a digit of class d, its GT center heatmaps are computed as

$$\mathbf{O}_{d,x,y} = \begin{cases} \exp\big(- \frac{\|x - \lfloor o'_x \rfloor\|_2^2}{2\sigma_x^2} - \frac{\|y - \lfloor o'_y \rfloor\|_2^2}{2\sigma_y^2}\big), \forall (x, y) \in R \\ 0 \quad \text{otherwise.} \end{cases} \tag{3.4}$$

In addition to center heatmaps, we regress digit center offsets $\mathbf{\Delta} \in \mathbb{R}^{2\times M\times M}$ for recovering from discretized coordinates. The first and second channels of $\mathbf{\Delta}$ represent the offsets along x-axis and y-axis respectively. The offset target at the digit center is $\mathbf{\Delta}_{\lfloor o'_x \rfloor, \lfloor o'_y \rfloor} = (o'_x - \lfloor o'_x \rfloor, o'_y - \lfloor o'_y \rfloor)$, and 0 on all other locations. For regression of size $\mathbf{S} \in \mathbb{R}^{2\times M\times M}$, the size target at $(\lfloor o'_x \rfloor, \lfloor o'_y \rfloor)$ is the feature-scale width and height (w', h') and 0 otherwise.

3.4.3.3 Losses

Regressions are performed for the three output heatmaps of the pose-guided branch. We use Gaussian focal loss [102, 104, 103] for digit center heatmaps with default hyper-parameters $\alpha = 2$ and $\gamma = 4$ for weight balancing. Let $\hat{\mathbf{O}}$ be the predicted center heatmaps, then the pixel-wise loss $\mathcal{L}_{\mathbf{O}_{d,x,y}}$ is defined as:

$$\mathcal{L}_{\mathbf{O}_{d,x,y}} = -\begin{cases} (1 - \hat{\mathbf{O}}_{d,x,y})^{\alpha} \log(\hat{\mathbf{O}}_{d,x,y}) & \text{if } \mathbf{O}_{d,x,y} = 1 \\ (\hat{\mathbf{O}}_{d,x,y})^{\alpha} (1 - \mathbf{O}_{d,x,y})^{\gamma} \log(1 - \hat{\mathbf{O}}_{d,x,y}) & \text{o/w.} \end{cases} \tag{3.5}$$

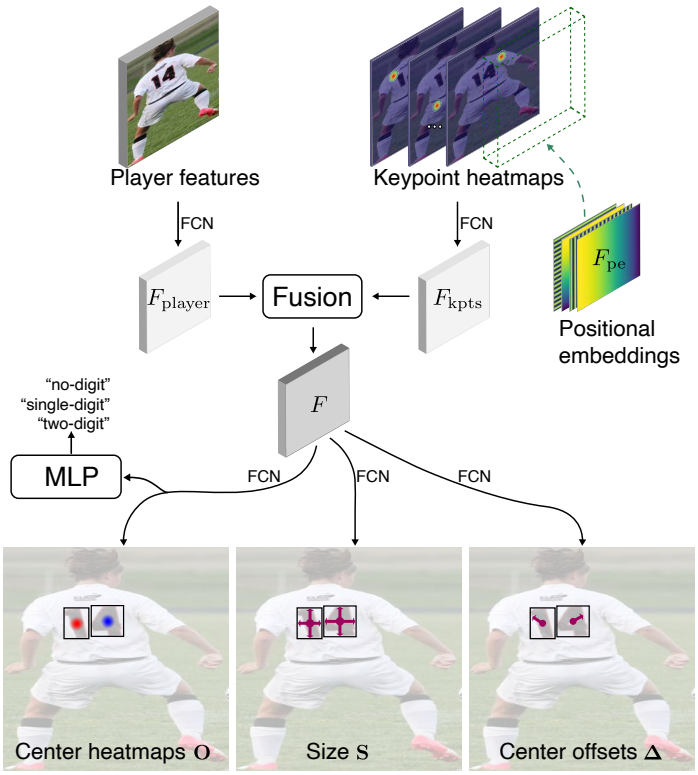


Figure 3.2: The architecture of the pose-guided branch. In this example, a two-digit jersey number is predicted. the center of digit "1" is predicted on the first channel of $\mathbf{O}$ shown as the red dot, and the center of digit "4" is predicted on the second channel of $\mathbf{O}$ shown as the blue dot. The predictions of size and center offset are location-aware and class-agnostic.

The offset and size targets are only defined at the GT digit center locations where $\mathbf{O}_{d,x,y} = 1$, and regressed via L1 loss as $\mathcal{L}_{\mathbf{\Delta}_{x,y}}$ and $\mathcal{L}_{\mathbf{S}_{x,y}}$. The overall digit detection objective is

$$\mathcal{L}_{det} = \frac{1}{N} \sum_{d,x,y} \left(\mathcal{L}_{\mathbf{O}_{d,x,y}} + \lambda_{\mathbf{\Delta}} \mathcal{L}_{\mathbf{\Delta}_{x,y}} + \lambda_{\mathbf{s}} \mathcal{L}_{\mathbf{S}_{x,y}} \right), \tag{3.6}$$

where N is the total number of digits within the player proposal; $\lambda_{\mathbf{\Delta}}$ and $\lambda_{\mathbf{s}}$ are hyper-parameters for weight balancing. We empirically set $\lambda_{\mathbf{\Delta}} = 1$ and $\lambda_{\mathbf{s}} = 1$ for all experiments.

3.4.3.4 Decoding Digit Proposals

With the output digit center heatmaps $\hat{\mathbf{O}}$, center offsets $\hat{\mathbf{\Delta}}$, and digit bounding box sizes $\hat{\mathbf{S}}$, we need to decode the digit proposals for both training and inference. To get the digit centers, we

follow the same step in [103]. Specifically, a `sigmoid` function is applied to the predicted center heatmaps $\hat{\mathbf{O}}$ such that the values are mapped into the range of $[0, 1]$. Then a 3×3 max pooling is applied to center heatmaps for filtering duplicate detections. The value of $\hat{\mathbf{O}}_{c,x,y}$ is considered as the measurement of the detection score. Then the top peaks in center heatmaps can be extracted as the detected digit centers $\{\hat{\mathbf{o}}^i\}_{i=1}^K$, where $\hat{\mathbf{o}}^i = (\hat{o}_x^i, \hat{o}_y^i)$ and K is the hyper-parameter to control how many digit proposals to keep per player proposal. During training, we set $K = 100$ where the top 50 digit proposals are kept plus 50 random proposals since we need negative training examples for digit classifications. For inference, we set $K = 20$ with the top 20 proposals for balancing the accuracy and inference speed.

To get the corresponding center offset and bounding box size, we gather the values at the detected digit center $(\hat{o}_x, \hat{o}_y)$ from $\hat{\mathbf{\Delta}}$ and $\hat{\mathbf{S}}$, namely the center offset $\hat{\mathbf{\Delta}}_{\hat{o}_x,\hat{o}_y} = (\hat{\delta}_x, \hat{\delta}_y)$ and bounding box size $\hat{\mathbf{S}}_{\hat{o}_x,\hat{o}_y} = (\hat{w}', \hat{h}')$. The width and height ratios, $\hat{r}_w$ and $\hat{r}_h$, can be computed via Equation 3.1 given the predicted player proposal $(\hat{x}_0^p, \hat{y}_0^p, \hat{x}_1^p, \hat{y}_1^p)$. Finally, the digit bounding box $(\hat{x}_0, \hat{y}_0, \hat{x}_1, \hat{y}_1)$ can be recovered as

$$
\begin{aligned}
\hat{x}_0 &= \hat{x}_0^p + (\hat{o}_x + \hat{\delta}_x - \hat{w}'/2)/\hat{r}_w, \\
\hat{y}_0 &= \hat{y}_0^p + (\hat{o}_y + \hat{\delta}_y - \hat{h}'/2)/\hat{r}_h, \\
\hat{x}_1 &= \hat{x}_0^p + (\hat{o}_x + \hat{\delta}_x + \hat{w}'/2)/\hat{r}_w, \\
\hat{y}_1 &= \hat{y}_0^p + (\hat{o}_y + \hat{\delta}_y + \hat{h}'/2)/\hat{r}_h.
\end{aligned}
\tag{3.7}
$$

To obtain jersey number detections, we group the digit detections based on the digit center class, and determine the jersey number length. We add a small multilayer perceptron (MLP) parallel to the output layer of the center prediction head, for classifying the number length ("no-digit", "single-digit", and "two-digit"). The MLP consists of a MaxPool layer (downsampling by a factor

of 2) and 3 fully-connected (FC) layers. The number length is denoted by $l \in \{0, 1, 2\}$, which will be used for generating jersey number detection.

3.4.4 Digit Branch and Jersey Number Detection

3.4.4.1 Digit Branch

We sample both positive and negative (ratio of 1:3) digit proposals from the output of the pose-guided branch for training, then extract 7×7 digit features via RoIAlign [20]. The digit branch architecture is the same as the player branch except that there are 11 output classes for digit classification, including 10 digit classes and 1 background class. The predicted digit class is denoted by c with a confidence score u, which will be used for generating jersey number detections.

3.4.4.2 Jersey Number Detection

We predict the jersey number length $\hat{l}$ for each player proposal as discussed in Section 3.4.3.4. Then within each player proposal, there are multiple digit detections denoted by $\mathbf{B}_{\text{digit}} = \{\hat{\mathbf{b}}^i\}_{i=1}^{K}$, where $\hat{\mathbf{b}}^i = (\hat{x}_0^i, \hat{y}_0^i, \hat{x}_1^i, \hat{y}_1^i, \hat{c}^i, \hat{u}^i)$, with the predicted digit center class $\hat{d}^i$ from the pose-guided branch. The jersey number detection is generated based on the predicted number length $\hat{l}$. For $\hat{l} = 0$, we simply discard all the bounding boxes; for single-digit case $\hat{l} = 1$, all the digit detections of $\hat{c}^i = 0$ are considered as number detections. For a two-digit number where $\hat{l} = 2$, we union digit bounding boxes pair-wisely and use the multiplication of corresponding digit class scores as the jersey number score. The jersey number class is the concatenation ($\oplus$) of digit classes for two-digit numbers. The top 100 jersey number detections are selected (`topk`) for evaluation purpose. The process to obtain the jersey number detections $\mathbf{B}_{\text{number}}$ is described in Algorithm 1.

Algorithm 1 Jersey number detection

Input: Digit detections $\mathbf{B}_{\text{digit}}$, jersey number length $\hat{l}$

Output: Jersey number detections $\mathbf{B}_{\text{number}}$

if $\hat{l} = 0$ **then**

  $\mathbf{B}_{\text{number}} \leftarrow \varnothing$

else if $\hat{l} = 1$ **then**

  $\mathbf{B}_{\text{number}} \leftarrow \mathbf{B}_{\text{digit}}$

else if $\hat{l} = 2$ **then**

  $\mathbf{B}_{d=0} \leftarrow \{\hat{\mathbf{b}}^i | \hat{d}^i = 0\}$, $\mathbf{B}_{d=1} \leftarrow \{\hat{\mathbf{b}}^j | \hat{d}^j = 1\}$

  $\mathbf{B}_{\text{number}} \leftarrow \{(\texttt{min}(\hat{x}_0^i, \hat{x}_0^j), \texttt{min}(\hat{y}_0^i, \hat{y}_0^j),$

  $\texttt{max}(\hat{x}_1^i, \hat{x}_1^j), \texttt{max}(\hat{y}_1^i, \hat{y}_1^j),$

  $\hat{c}^i \oplus \hat{c}^j, \hat{u}^i \times \hat{u}^j \ ) \mid$

  $\hat{\mathbf{b}}^i \in \mathbf{B}_{d=0} \wedge \hat{\mathbf{b}}^j \in \mathbf{B}_{d=1}\}$

end if

$\mathbf{B}_{\text{number}} \leftarrow \texttt{topk}(\mathbf{B}_{\text{number}})$

return $\mathbf{B}_{\text{number}}$

Table 3.1: Dataset statistics. The collected statistics from the second left to the rightmost column are: the number (#) of images, the number of annotated digits, the number of annotated players, the number of players with annotated keypoints, the mean and standard deviation of the bounding box size of players and digits (in pixels), and the bounding box area ratio of digit to player.

Video	# Images	# Digits	# Players	# Kpts	Image width	Image height	Player width	Player height	Digit width	Digit height	Area ratio
1	1030	1940	1323	1062	224.00 ± 0.00	224.00 ± 0.00	100.13 ±25.78	172.18 ± 27.71	20.06 ± 5.69	24.08 ± 6.59	0.03 ± 0.01
2	706	1347	768	732	173.42 ± 30.58	281.64 ±48.28	95.14 ±30.25	242.57 ± 54.12	15.58 ± 3.87	28.94 ± 6.01	0.02 ± 0.01
3	418	829	468	420	224.46 ± 37.58	389.36 ±80.58	154.35 ±46.72	343.98 ± 97.66	28.56 ±10.66	53.08 ±16.22	0.03 ± 0.01
4	1413	2180	1769	1472	225.23 ± 42.41	376.15 ±79.79	139.77 ±52.17	324.87 ±102.18	22.01 ± 7.43	44.69 ±12.62	0.02 ± 0.01
5	910	2779	1726	1720	640.00 ± 0.00	360.00 ± 0.00	82.38 ±23.07	160.83 ± 39.89	10.16 ± 2.97	19.71 ± 3.12	0.02 ± 0.01
Total	4477	9075	6054	5406	301.01 ±174.63	324.19 ±84.19	110.21 ±45.33	235.77 ±102.45	17.61 ± 8.36	31.06 ±14.81	0.02 ± 0.01

3.4.5 Data Augmentations

During training, we employ multi-scale training by randomly resizing the images to several predetermined scales. To further improve both the player and jersey number detection performance, we introduce three data augmentation methods specifically designed for the jersey number detection task in this section. The jersey number detection is highly dependent on the player detection performance. The more accurate player bounding boxes are predicted, the better digit localization performance is thanks to the proposed pose-guided branch. However, the digit classification is challenging due to many factors such as motion blur, clothing deformation, *etc*. We collected cropped player images from several soccer matches with annotations of players' bounding boxes, keypoints, and digit bounding boxes. However, the collected data does not cover enough scene variations and range of jersey numbers. A jersey number may correlate to its popularity and a player's position in certain sports, but the digit distribution suffers significant bias given the limited data. Sufficient data for training both player detection and digit classification is needed.

3.4.5.1 Pretraining

To create a general jersey number detection framework that works for most sports, we need more data besides our dataset. We incorporate the Street View House Number (SVHN) dataset [33] and COCO Keypoints dataset [48] for pretraining. SVHN contains images of numbers with annotated digit bounding boxes, and COCO contains images of persons with annotated bounding boxes and keypoints. Specifically, during each training iteration, we randomly select data from SVHN and COCO with equal probabilities. The backbone network is trained on both datasets for obtaining robust player and digit feature representations; RPN and player branch are only trained on COCO for generating robust person detection and keypoint estimation. As for the digit branch, the digit features are pooled from the ground-truth digit bounding boxes and used for training the digit classifier. During pretraining, the pose-guided branch is unused.

3.4.5.2 CopyPasteMix

As discussed in our previous work [12], our collected images are enlarged patches cropped from whole video frames. Each image contains at least one player with annotations. Inspired from recent work on data augmentation [105, 106, 107, 108, 109], we propose a data augmentation method called `CopyPasteMix` that provides more variability to the training data. It copies random number of images and pastes onto a black background image. The source images are resized randomly, and each image is pasted in order of largest to smallest onto a random location. If one image has an IoU over 0.5 with the previously pasted image, we re-generate the target location and retry. For any two images with IoU less or equal to 0.5, we perform a linear blending of the two images with equal weights. After all the sources images are pasted, we adjust the ground-truth an-

Figure 3.3: Example augmented images (left to right) using `CopyPasteMix`, `SwapDigit`, and `CopyPasteMix+SwapDigit`.

notations accordingly. We then train on the resulting synthetic image. By using `CopyPasteMix`, we have more training targets at different scales and locations per image. Empirically, the number of images used to construct the synthetic image is randomly selected between 1 and 5.

3.4.5.3 SwapDigit

We also design another augmentation method called `SwapDigit`. We borrow the data from SVHN, such that the RoI of a digit in our training images is randomly replaced with a cropped digit RoI in SVHN. By using `SwapDigit`, we effectively mitigate the lack of digit annotations problem such that each digit class can be trained with enough data for the digit classifier. It is worth noting that `CopyPasteMix` and `SwapDigit` can be used simultaneously for maximizing data efficiency and performance gain.

Among all the discussed data augmentation methods, only translation or scaling of bounding boxes are involved. The human body keypoints are changed accordingly with respect to the players' bounding boxes. We do not want to undermine the geometric relationship between the

human pose and digit locations, so no other transformation is carried out. Some examples of augmented images are shown in Figure 3.3. In Section 3.5, we show significant performance gains by using the proposed augmentation methods, and investigate the individual and combinatorial effects of the augmentations in Section 3.5.6. A theoretical analysis on the effectiveness of the proposed data augmentation methods is provided in the supplementary material.

3.5 Experiments

To validate the effectiveness of JEDE, we conduct experiments and compare with other state-of-the-art methods on our collected dataset as there is no publicly available dataset. We conduct evaluations on both jersey digit and number detection tasks. Following the standard COCO [48] metrics for object detection, we report the bounding box AP and AR (averaged precision and recall across IoU thresholds from 0.5 to 0.95 with an interval of 0.05), AP_{50}, AP_{75}, AR_{50}, and AP_{75} where the IoU threshold is denoted by the subscript.

3.5.1 Dataset

There is no publicly available jersey number dataset with instance-level annotations. To identify players in sports scenes, predicting the jersey numbers can be more robust than from other visual information like face and gait. It is natural to consider the jersey number detection as a top-down process where player localization provides robust prior information for digit localization. Human pose information can also be useful by providing implicit constraints on digit locations. Previous work [4, 24] only investigates image-level jersey number recognition which is not practical for real-world applications where multiple players are involved. Instead of performing player identity

classification directly, digit detection provides more accurate visual cues of players' identities. To continue the advances towards the ultimate goal of automatic sports analysis, we introduce a new dataset that addresses three core research problems in detection for sports: player detection, player pose estimation, and digit detection.

We choose soccer and basketball, two of the most popular team sports, for creating the dataset. To collect data with sufficient variations, the game match videos are selected based on different jersey colors, jersey number colors, and jersey number fonts. The soccer data is collected from four matches. The recording device used is a single Canon XA10 video camera which is installed on a pole that is 15 feet high, and 10 to 20 feet away from the horizontal baseline of the soccer field. For better video qualities on jersey numbers, the camera operator is allowed to pan and zoom accordingly. Next, the video frames are enlarged by a factor of 2. An off-the-shelf person detector (*e.g.*, OpenPose [46]) is applied to get players' bounding boxes. The image is cropped around each bounding box with a padding of 150 pixels and a random shift within 20 pixels to create data variations. Besides the soccer sport, we also collect frames from one basketball match. To increase the diversity and add more challenging training data, the basketball frames are enlarged by a factor of 2 and divided into 4 large patches of equal size. After the data collection is completed, the images are labeled via VGG Image Annotator [47]. For each player in an image, we annotate its bounding box and legible digits. For players with digit annotations, 4 human body keypoints (left shoulder, right shoulder, left hip and right hip) are also annotated.

In total, there are 4477 labeled images, including annotations of 6054 players with 5406 of them are labeled with keypoints, and 9075 digits. We list the statistics of the collected dataset in Table 3.1. There are large variations in scales within each collected video, and even larger across

videos. The digit to player bounding box area ratio is only around 2% as shown in the table. The relatively small scale of digits makes jersey number detection even a more challenging task. Some example image and digit class distributions for each video are shown in Figure 3.4. The differences in image appearance and digit distribution are significant. For a fair evaluation on the unbalanced dataset, we perform k-fold cross validation where the training and testing data are divided by videos for examining the generalization power of JEDE.

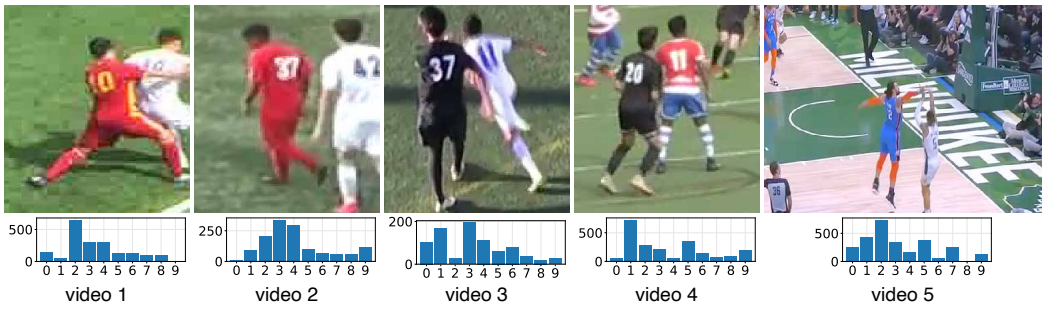


Figure 3.4: Example images from the collected dataset for each video labeled in numerical ascending order. Images are resized for illustration. The second row shows the histogram of digit annotations for each video.

3.5.2 Implementation Details

Our implementation is built upon the codebase Detectron2 [110] and PyTorch [111]. All experiments are conducted on a workstation with two Nvidia 1080 Ti GPUs. The model is trained in parallel and evaluated on a single GPU. We set the hyper-parameters following Mask R-CNN [20]. A fixed random seed is used, and no data balancing strategy is applied in all experiments for fair comparisons. We perform 4-fold cross validation on the collected soccer data, and conduct cross-domain evaluations by training on soccer and testing on basketball data, and vice versa.

We first implement a JEDE baseline with a ResNet-50-FPN backbone [3, 20]. The posi-

Table 3.2: Jersey digit detection results.

Fold	Method	AP	AP_{50}	AP_{75}	AR	AR_{50}	AR_{75}
1	Faster R-CNN	25.26	42.01	27.52	40.00	41.42	41.42
	Cascade R-CNN	25.10	35.41	31.79	39.97	40.32	40.32
	TridentNet	26.06	40.30	31.68	38.89	40.85	40.85
	Pose-guided R-CNN	27.03	44.05	30.55	40.22	42.15	42.15
	JEDE Baseline	30.28	55.96	28.98	43.96	44.67	44.67
	JEDE Augmented	**51.08**	**80.10**	**60.09**	**64.24**	**65.33**	**65.33**
2	Faster R-CNN	49.61	71.79	61.05	67.48	69.09	69.09
	Cascade R-CNN	54.97	76.00	67.00	69.86	70.60	70.60
	TridentNet	56.07	77.68	70.79	69.97	70.60	70.60
	Pose-guided R-CNN	48.33	71.01	60.23	65.43	67.07	67.07
	JEDE Baseline	49.46	73.45	61.31	66.6	67.47	67.47
	JEDE Augmented	**59.70**	**86.28**	**74.73**	**70.89**	**71.23**	**71.23**
3	Faster R-CNN	55.79	76.69	67.44	64.74	67.30	67.30
	Cascade R-CNN	**60.01**	77.93	**72.53**	**72.27**	**74.59**	**74.59**
	TridentNet	57.50	75.35	70.07	67.09	68.54	68.54
	Pose-guided R-CNN	46.74	75.32	53.76	55.89	57.14	57.14
	JEDE Baseline	48.80	77.65	54.38	57.85	59.33	59.33
	JEDE Augmented	56.85	**88.44**	65.76	64.70	66.28	66.28
4	Faster R-CNN	49.17	65.64	60.86	66.33	68.57	68.57
	Cascade R-CNN	62.09	76.18	73.59	72.91	**74.52**	**74.52**
	TridentNet	59.29	75.94	72.65	68.30	69.16	69.16
	Pose-guided R-CNN	51.45	71.27	63.33	65.67	67.01	67.01
	JEDE Baseline	53.32	73.80	65.46	66.93	68.02	68.02
	JEDE Augmented	**67.15**	**91.65**	**82.81**	**73.02**	74.25	74.25

tional embeddings are not included, and only multi-scale training is used during training. We then implement an augmented version of JEDE baseline by adding the positional embeddings and using the proposed augmentation methods. Specifically, we pretrain the model on COCO and SVHN simultaneously with equal sampling probabilities, where `CopyPasteMix` with a maximum of 5 images is applied for SVHN data. Then, we train on our dataset using both `CopyPasteMix` and `SwapDigit` with the same hyper-parameters of the baseline model.

Training: The input images are resized such that their longer edge is no more than 800 pixels. For the short edge, we use multi-scale training such that a random scale is selected during each iteration. We follow some common pixel values for the shorter edge [19, 20] to choose from: 480, 512, 544, 576, 608, and 640. For each image, there are 512 sampled player RoIs with a ratio of 1:3 of positive samples (RoIs with IoU $\geq$ 0.5 over GT player bounding boxes) to negatives. For each detected player, we sample 100 digit proposals from the pose-guided branch. Within the predicted center heatmaps of each player, we sample the top 50 locations with the highest scores, and randomly sample other 50 from the rest as negative samples. The maximum number of sampled digit proposals per image is set as 256. We train the whole framework for 50k iterations, with a mini-batch size of 4 (2 images per GPU on 2 GPUs) and learning rate of 0.0002. The learning rate is decreased by 10 at the 40k-th iteration. We use a weight decay of 0.0001 and momentum of 0.9.

Inference: The test images are resized such that their longer edge is no more than 800 pixels while the shorter edge is at least 480 pixels. Top 1000 player proposals from RPN are kept. We run the player branch on these proposals followed by non-maximum suppression to get the top 100 player bounding boxes ranked by classification scores. The keypoints regression and pose-guided branch are applied to these selected boxes. For each player instance, we keep the top 20 digit proposals. Finally, all the digit proposals are fed into the digit branch to obtain the final classifications and bounding boxes, followed by non-maximum suppression.

3.5.3 Digit Detection Results

In this sub-section, we evaluate the jersey digit detection performance with thorough comparisons of JEDE to pose-guided R-CNN [12], and state-of-the-art object detectors such as Faster

Table 3.3: Jersey number detection results.

Fold	Method	AP	AP_{50}	AP_{75}	AR	AR_{50}	AR_{75}
1	Mask TextSpotter V3	-	2.43	0.25	-	35.84	3.72
	SwinTextSpotter	5.28	6.85	6.44	14.71	16.55	16.58
	Pose-guided R-CNN	13.15	18.35	14.67	27.43	28.02	28.02
	JEDE Baseline	18.07	26.87	20.93	35.69	35.99	35.99
	JEDE Augmented	**37.12**	**50.69**	**45.84**	**50.29**	**50.63**	**50.63**
2	Mask TextSpotter V3	-	0.77	0.11	-	11.49	1.65
	SwinTextSpotter	10.54	12.27	12.00	14.64	18.54	18.54
	Pose-guided R-CNN	32.23	42.27	39.73	47.85	48.84	48.84
	JEDE Baseline	36.02	43.59	42.69	50.26	50.61	50.61
	JEDE Augmented	**36.12**	**45.60**	**44.08**	**54.61**	**54.65**	**54.65**
3	Mask TextSpotter V3	-	1.80	0.48	-	29.24	7.88
	SwinTextSpotter	11.29	13.83	13.49	23.53	26.55	26.63
	Pose-guided R-CNN	43.12	55.12	52.36	58.42	60.12	60.12
	JEDE Baseline	45.17	59.03	55.42	**62.38**	**63.83**	**63.83**
	JEDE Augmented	**48.27**	**61.72**	**57.51**	57.85	59.31	59.31
4	Mask TextSpotter V3	-	0.21	0.12	-	5.65	3.27
	SwinTextSpotter	9.73	12.82	11.35	17.33	19.64	19.87
	Pose-guided R-CNN	33.76	40.92	38.45	44.77	46.58	46.58
	JEDE Baseline	36.35	44.42	42.84	47.95	48.27	48.27
	JEDE Augmented	**42.87**	**52.82**	**51.53**	**53.18**	**54.19**	**54.19**

R-CNN [19], Cascade R-CNN [112], and TridentNet [113]. Serving as competitive methods, Cascade R-CNN includes a sequence of detectors that improve the detection quality, while TridentNet is robust to object scale variations.

The results of JEDE models are listed and compared with other methods in Table 3.2. For each cross validation result, the fold number indicates which test video is used, *e.g.* fold 1 means that we train on videos 2, 3, and 4, then test on video 1. Our models achieve the state-of-the-art results with substantial improvements. JEDE baseline already outperforms Faster R-CNN over most metrics, and the augmented JEDE further improves the results. Fold 1 and 4 involve less training

data and more testing data, and we see more performance gains for JEDE. For example of fold 1, JEDE baseline has 5.02 points improvement in AP over Faster R-CNN, and 3.25 points over pose-guided R-CNN. The results prove the effectiveness of the pose-guided branch. The digit localization can be improved by extracting contextual information from the pose features with limited training data. Moreover, the model trained with augmentations achieves massive gains such that AP is doubled compared with Faster R-CNN. Both precision and recall are dramatically improved over the baseline model, demonstrating that we have more accurate bounding box predictions and better digit classifiers. This highlights that our proposed augmentation methods are capable of diversifying the training data without any extra cost. It can be seen that the results of JEDE for fold 3 are slightly lower than Cascade R-CNN. Since fold 3 only contains 418 testing images, whose size is relatively small compared with other folds, it is a natural disturbance for a such small performance gap. The augmented JEDE still outperforms Cascade R-CNN by 10.51 AP_{50}.

3.5.4 Jersey Number Detection Results

For jersey number detection evaluations, we use the same metrics as in digit detection. Naturally, the precision is less than it is in digit detection, as both digits must be detected corrected for a two-digit jersey number to be counted as a true positive detection. We report the results of JEDE models, and compare with the pose-guided R-CNN and the state-of-the-art scene text detection frameworks Mask TextSpotter V3 [81], and SwinTextSpotter [83]. We use their open-sourced codebases and modify the output number of classes to 11 (“0” -“9” and “background”). The model Mask TextSpotter V3 is initialized with their pre-trained weights, and SwinTextSpotter is trained from scratch. Both models are trained with the same settings as in Section 3.5.2.

Figure 3.5: Qualitative comparisons between JEDE and other methods. Each bounding box is labeled with the predicted class and score, if available. The digit class is labeled under the left bottom corner of the bounding box for all methods (rows 1, 2, 3, 6, 7), and the jersey number is labeled above the box for JEDE models (rows 6, 7). Only jersey numbers are labeled for Mask TextSpotter V3 and SwinTextSpotter. Images are resized for illustration.

Table 3.3 compares our results to Mask TextSpotter V3, SwinTextSpotter, and pose-guided R-CNN. For Mask TextSpotter V3, we only report AP_{50}, AP_{75}, AR_{50}, and AR_{75}. JEDE outperforms other methods in every fold by a large margin. For example in fold 1, the baseline JEDE achieves 26.87 AP_{50}, which shows 24.44 points improvement over Mask TextSpotter V3 and 20.02 points improvement over SwinTextSpotter. The augmented JEDE further pushes the performance with 50.69 AP_{50}. It is worth noting that both Mask TextSpotter V3 and SwinTextSpotter have poor AP but fair AR results. This indicates that these methods are able to detect jersey number bounding boxes but barely classify them correctly. To examine this behavior, we conduct experiments by modifying our detection branches similar to commonly used scene text detection methods. Specifically, we detect jersey number bounding boxes directly where the union of digit bounding

Table 3.4: Performance comparison for S→B task.

Method	AP	AP_{50}	AP_{75}	AR	AR_{50}	AR_{75}
Faster R-CNN	0.00	0.01	0.01	0.12	0.15	0.16
Cascade R-CNN	0.01	0.01	0.01	0.16	0.19	0.19
TridentNet	0.00	0.00	0.00	0.02	0.02	0.02
Pose-guided R-CNN	0.04	0.32	0.00	0.25	0.28	0.28
JEDE Baseline	0.09	0.46	0.00	0.36	0.37	0.37
JEDE Augmented	**10.17**	**30.34**	**3.91**	**19.97**	**24.06**	**24.07**
Mask TextSpotter V3	-	0.00	0.00	-	0.08	0.00
SwinTextSpotter	0.05	0.08	0.08	0.04	0.05	0.05
Pose-guided R-CNN	0.01	0.07	0.00	0.05	0.05	0.05
JEDE Baseline	0.02	0.09	0.00	0.07	0.07	0.07
JEDE Augmented	**9.90**	**21.60**	**7.71**	**20.05**	**22.56**	**22.56**

boxes is considered for a two-digit number. Then we add the sequence modeling (*e.g.*, Bidirectional LSTM [114, 115]) for pooled jersey number features as in [116, 117, 81]. The number classification is performed per column of the features that can be trained using Connectionist Temporal Classification (CTC) [118]. This modified model achieves 13.71 AP_{50} which is much lower than the JEDE baseline. The unsatisfied results can be justified by the sequence modeling of jersey numbers. Scene text detection relies on lexicons and contextual information between characters, while there is no such dependence between digits in a jersey number. Moreover, there are not enough jersey numbers for training a robust recurrent model for sequence classification. Traditional scene text detection frameworks are probably not suitable for directly detecting jersey numbers unless heavy adaptations are applied.

In Figure 3.5, we provide the qualitative comparisons between JEDE and other methods.

Only detections with a confidence score over 0.2 are shown. JEDE models consistently perform better under different conditions. The pose-guided branch provides a strong regularization for the digit locations as seen from columns 7 and 8 in the figure. JEDE models predict accurate digit bounding boxes, while other methods predict wrong locations on arms or hips. It can also be seen that JEDE is more robust to low-resolution images (columns 1 and 2). The last column shows a failure case where an extreme pose is presented: "9" is not recognized correctly due to rare deformations. JEDE Augmented still predicts an accurate digit bounding box, while other methods generate some false positive detections. It further demonstrates that the pose-guided branch provides more reliable digit proposals. Nevertheless, there are some failure cases for the proposed method as well. For example, in the last row of Figure 3.5: in the 4-th image, "46" is not detected due to partial occlusion; in the 9-th image, "39" is not recognized since "single-digit" is predicted from the jersey number length classifier.

3.5.5 Cross-domain Results

It is well known that deep learning vision models are highly dependent on large labeled datasets. Even though a trained model can success on one dataset, its performance often declines significantly on new data or new domain. This problem is referred to as dataset shift [119] where training and testing data distributions are different. It can be observed from Tables 3.2 and 3.3 that the detection performance differs among each testing fold, although both training and testing data are collected from soccer matches. The visual changes between soccer matches are substantial, not to mention the domain shift from soccer to other sports, and vice versa. Thus, we perform two cross-domain experiments between the soccer and basketball data, without using any transfer

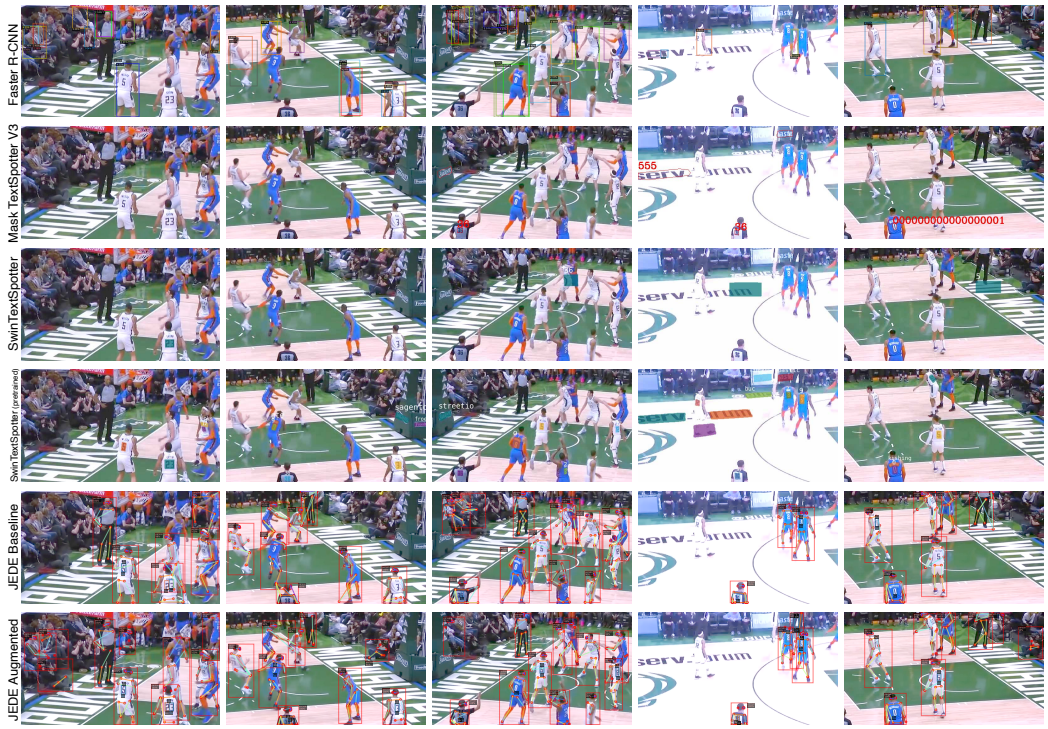


Figure 3.6: Qualitative comparison on the cross-domain task S→B. Faster R-CNN, Mask TextSpotter V3, and SwinTextSpotter achieve poor performances with false positive detections that predict players, body parts, or texts as digits. JEDE Baseline performs well on player detection and pose estimation with fair digit classification performance. JEDE Augmented is much more robust to recognize digits thanks to the proposed data augmentation methods.

learning [120] or domain adaptation [121] technique.

The first experiment is training on soccer and testing on basketball data (S→B). During testing, we resize the input image such that the longer edge is no more than 1600 pixels, and the short edge is at least 960 pixels. The results are shown in Table 3.4. Although models trained without augmentations suffer from the domain shift and limited data, JEDE Baseline still outperforms other methods significantly. With augmentations, JEDE achieves much better performance with 30.34 digit AP_{50} and 21.60 number AP_{50}. We further show the qualitative results in Figure 3.6. For SwinTextSpotter, we also use their pre-trained model to perform text detection directly as shown in the 4-th row of Figure 3.6. JEDE Augmented still achieves the best detection results overall. Pre-

Table 3.5: Performance comparison for B→S task.

Method	AP	AP_{50}	AP_{75}	AR	AR_{50}	AR_{75}
Faster R-CNN	1.49	3.80	0.98	3.04	3.46	3.46
Cascade R-CNN	1.46	2.94	1.33	3.85	4.01	4.01
TridentNet	0.14	0.65	0.00	0.84	0.84	0.84
Pose-guided R-CNN	1.51	4.02	1.07	3.60	3.83	3.83
JEDE Baseline	0.63	2.54	0.15	1.94	2.11	2.11
JEDE Augmented	**9.09**	**32.54**	**1.85**	**15.81**	**16.29**	**18.61**
Mask TextSpotter V3	-	0.98	0.16	-	**17.96**	3.03
SwinTextSpotter	0.00	0.00	0.00	0.06	0.10	0.11
Pose-guided R-CNN	0.20	0.78	0.05	0.48	0.50	0.50
JEDE Baseline	0.23	0.87	0.06	0.56	0.58	0.58
JEDE Augmented	**8.40**	**20.24**	**5.22**	**13.97**	14.02	**14.02**

trained SwinTextSpotter can achieve fair detection results, but cannot distinguish between jersey numbers and other texts. In some difficult conditions as shown in Figure 3.6, JEDE may fail, such as in the second image, "77" is not detected due to motion blur; and in the 4-th image, "12" is not detected due to small digit scale.

The other experiment is training on basketball and testing on soccer data (B→S). During testing, we resize the input image such that the longer edge is no more than 400 pixels, and the short edge is at least 240 pixels. `CopyPasteMix` is not applied for JEDE Augmented. The results are shown in Table 3.5. We observe worse results compared with the ones on S→B, since there is much less training data (but more testing data) for basketball domain (Table 3.1). Nevertheless, JEDE Augmented still achieves the best results among all the methods with number AP_{50} being over 20 times better than Mask TextSpotter V3. By comparing the JEDE Baseline and Augmented

results, it implies that `SwapDigit` significantly mitigate the problem of limited training data, suggesting the high practicality of the proposed data augmentation strategies. The cross-domain results demonstrate that JEDE models have better generalizability over other methods.

3.5.6 Ablation Study

We conduct a number of ablations to analyze JEDE, and show the digit detection results in this section unless otherwise specified. All the experiments are based on JEDE Baseline. Best results are in bold.

Backbone: Table 3.6 shows a comparison of JEDE results, number of parameters (in Million), giga floating point operations (GFlops), and inference frame per second (FPS) of various backbones. The metrics are measured on a Nvidia GTX 1080 Ti GPU with a batch size of 1, and the GFlops and FPS are averaged over all testing images. We observe that the results do not benefit from much deeper networks such as ResNet-101 and ResNeXt-101 due to overfitting on limited training data. The JEDE Augmented with ResNet-101 slightly outperforms it with ResNet-50 by 0.08 as shown in Table 3.2. It indicates that the proposed data augmentation methods are more effective than increasing the model size. We also observe notable speed and performance improvements by removing the anchor size of 32. Since small proposals are simple negative examples (background) that do not contribute much to the loss, the model acquires more effective proposals during training by removing small anchors. We also compute the inference speed of Mask TextSpotter V3 and SwinTextSpotter with the same settings since the jersey number detection task is time-sensitive. On average, Mask TextSpotter V3 achieves 5.04 FPS and SwinTextSpotter achieves 2.46 FPS, while JEDE with ResNet-50 backbone achieves 22.68 FPS as a comparison. **GT Minimum IoU:** As

Table 3.6: Ablation on backbone networks.

Backbone	AP	AR	Params	GFlops	FPS
ResNet-18	11.58	23.51	**67.4M**	52.5	**24.55**
ResNet-50	25.15	38.80	80.5M	60.9	20.37
ResNet-50†	**25.33**	**40.14**	80.5M	**52.0**	22.68
ResNet-101	24.50	38.85	93.2M	77.7	17.54
ResNeXt-101-32x8d	19.62	35.98	144.0M	119.2	10.76

† The smallest anchors of size 32 are removed.

discussed in Section 3.4.3.2, the hyper-parameter min_iou controls the Gaussian blob size of the digit centers. The larger min_iou is, the smaller the blob size is. Table 3.7 shows the results of using different values of min_iou. It can be observed that smaller min_iou gives better results because more training samples are included while training center heatmaps regression. **GT heatmap spatial size:** The spatial size of the GT heatmaps is important for predicting the center locations of digits. Hypothetically, higher resolution the heatmaps have, more accurate the prediction will be due to less coordinate discretization. We perform the sensitivity analysis of the heatmap size and show the results in Table 3.8. The input player features are interpolated to the desired size. As expected, using the largest size achieves slightly better AP. However, simply up-sampling features does not improve the results significantly. We further perform the experiment by pooling 56×56 player features directly. As a result, AP is significantly improved by 4.67 compared with using 14×14.

Keypoint s player features: We investigate the effectiveness of the pose-guided branch.

Table 3.7: GT Bounding box minimum overlap: Smaller value gives better results.

min_iou	AP	AR
0.1	**26.46**	**38.95**
0.3	25.79	38.69
0.5	26.12	38.77

Table 3.8: GT heatmap spatial size: Larger size gives better results.

Size	AP	AR
14×14	24.15	35.68
28×28	22.90	36.06
56×56	24.53	35.73
$56 \times 56^{\ddagger}$	**28.82**	**39.96**

‡ Directly pooled without interpolation.

Table 3.9: Input to pose-guided branch: fusion of both features gives better results.

Input Features	AP	AR
Keypoint	12.77	24.02
Player	20.81	31.64
Both	**23.55**	**34.81**

Table 3.10: Feature fusion methods: concatenation gives better results.

Fusion method	AP	AR
Multiply	20.79	33.50
Sum	21.40	33.76
Concatenate	**23.07**	**36.43**

There are two default input features, the keypoint heatmaps and pooled player features. We conduct ablation experiments by 1. only using the keypoint features, 2. only using the player features, and 3. using the fusion of both features. The results are shown in Table 3.9. Using both features significantly outperforms only using one set of features. By adding the keypoint features, AP is improved by 2.74 points. It validates the effectiveness of the pose-guided branch that pose information is helpful for localizing digits.

Fusion methods: We also investigate the fusion methods of the player features and key-

Table 3.11: Positional embeddings (PE): Concatenation w/ keypoint heatmaps gives better results.

Concat w/	AP	AR
No PE	23.07	36.43
Fused features	24.81	35.89
Keypoint heatmaps	**25.29**	**37.96**

Table 3.12: Normalization layers: GN provides better results using a small batch size.

Norm	AP	AR
None	21.44	32.14
BN	22.12	34.58
GN	**27.73**	**42.51**

point features, and show the results in Table 3.10. Concatenation has better performance because it provides more feature transformations for the branch to reduce the semantic difference between the player features and the keypoint heatmaps.

Table 3.13: Ablation on the input features for number length classification.

Input features	AP	AR
Center heatmaps	11.11	20.97
2nd last features	**12.75**	**23.80**
Fused features	12.66	20.16

Table 3.14: Digit RoI pooling resolution: larger resolution gives better results.

Pooling Res.	AP	AR
7×7	20.79	35.34
14×14	23.19	35.22
28×28	**24.99**	**36.60**

Positional embeddings: We examine the effectiveness of concatenating positional embeddings (PE) with different features, fused features or keypoint heatmaps, and show the results in Table 3.11. Adding the positional embeddings improves the results, and concatenating with keypoint heatmaps outperforms with fused features by 0.48 AP. One possible reason is that both PE and keypoint heatmaps provide spatial information and make learning easier.

Normalization layers: We further perform an ablation study on the normalization layers used in the pose-guided branch. Two commonly used feature normalization methods are selected, namely Batch Normalization (BN) [122] and Group Normalization (GN) [123], and the results are shown in Table 3.12. The results are improved by using normalization layers, and GN performs better than BN due to training with a small batch size of 4. We expect that better results can be achieved by using a larger batch size.

Number length classification: As discussed in Section 3.4.4.2, we use a MLP to predict the jersey number length. We investigate three possible features to be considered as its input: center heatmaps, the second last features in the center prediction head (features before the center heatmaps), and the fused features. The jersey number detection results are shown in Table 3.13. Using the 2nd last features gives the best AP and AR. Using the center heatmaps has worse performance due to the loss of contextual information of the player, which is useful for number length classification.

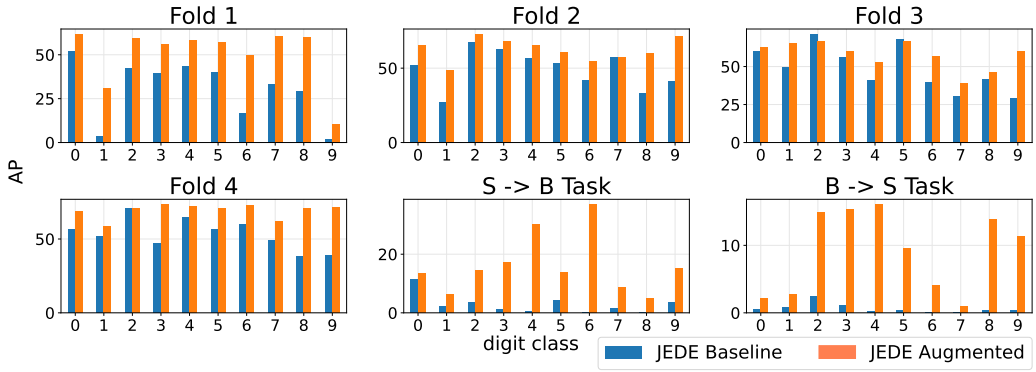


Figure 3.7: Per-class AP comparisons between JEDE Baseline and Augmented for each fold and cross-domain task.

Digit RoI pooling resolution: One of the hyper-parameters of the digit branch is the digit

Figure 3.8: Qualitative results for images in the wild. Sports from left to right, top to bottom are: soccer, lacrosse, rugby, American football, cricket, basketball, volleyball, ice hockey, handball, beach soccer, hockey, and water polo.

RoI pooling resolution. We conduct experiments with different resolutions and shown the results in Table 3.14. It can be observed that larger pooling size gives better performance. Higher resolution provides more fine-grained visual features that are helpful for differentiating similar digits like "1" and "7".

Table 3.15: Ablation on data augmentations.

Random Crop	Pre-train	`CopyPasteMix`	`SwapDigit`	AP	AR
				21.44	32.14
✓				23.16	35.90
	✓			30.37	40.73
		✓		39.24	57.13
			✓	30.36	44.57
	✓	✓		45.95	60.42
	✓	✓	✓	**49.80**	**63.35**

Augmentations: Besides the ablation on architectures, we also perform the effects of data augmentations on JEDE, including random crop with a maximum of 30% crop rate, pre-train on COCO and SVHN, `CopyPasteMix`, and `SwapDigit`. For JEDE Baseline, no augmentation is conducted to balance the uneven distribution of digits for fair comparison. The results are shown in Table 3.15. Each data augmentation method improves the results compared to JEDE Baseline. `CopyPasteMix` achieves the largest gain of AP by 17.80, while random crop only improves AP by 1.72. Moreover, combining `CopyPasteMix` and `SwapDigit` with pre-trained weights gives the best AP gain of 28.36, and AR gain of 31.21. The results demonstrate that data augmentation is one of the key factors for improving the performance given limited data. For jersey number detection task, our specially designed augmentation methods have stronger regularization capability than general methods like random crop.

Better results could be achieved by re-sampling [124], but the testing data distribution will still be very different from training data in the cross validation. This is the reason why `SwapDigit` is implemented for mitigating the problem of unbalanced data. By using the SVHN dataset which is well-balanced for digit annotations, the detection performance is much improved for those digit classes that are trained insufficiently. We provide a per-class comparison of JEDE Baseline and Augmented as shown in Figure 3.7. It can be observed that larger relative improvements are achieved for those digit classes that have a low AP.

We also evaluate and compare the player bounding box detection mean average precision (AP^{player}) and keypoint detection mean average precision (AP^{kpts}) for each experiment following the standard metrics [48]. We only report on the four categories of keypoints for which annotations are available in our dataset. The results are shown in Table 3.16. It can be observed that JEDE

Table 3.16: Comparison of JEDE Baseline and Augmented on player detection and human pose estimation results.

Experiment	Method	AP^{player}	AP^{kpts}
Fold 1	JEDE Baseline	76.52	94.88
	JEDE Augmented	**80.44**	**97.13**
Fold 2	JEDE Baseline	**81.44**	99.45
	JEDE Augmented	79.57	**99.58**
Fold 3	JEDE Baseline	**83.29**	91.27
	JEDE Augmented	75.49	**96.31**
Fold 4	JEDE Baseline	**80.59**	96.45
	JEDE Augmented	76.45	**98.03**
S→B	JEDE Baseline	**41.76**	49.61
	JEDE Augmented	41.42	**52.46**
B→S	JEDE Baseline	61.83	65.86
	JEDE Augmented	**70.41**	**83.05**

Augmented achieves better keypoint detection performance in each experiment, especially when less training data is available in the B→S task. The testing images for each fold are cropped frames with low variances, and thus the data distribution difference between training and testing are small. With data augmentations, we provide a stronger regularization by training with more instances at different scales. As a result, JEDE Augmented receives less training data that are similar to the testing data. This suggests why JEDE Augmented achieves slightly lower AP^{player} in some

experiments compared with the baseline.

Towards a universal jersey number detector: To fully exploit the capability of JEDE, we develop a slightly larger model using the best hyper-parameters found in the ablation studies. We use larger input image size with a maximum of 1589 pixels for the longer edge. We first pre-train the model on SVHN and COCO, and then train using `CopyPasteMix` and `SwapDigit`. To prevent "forgetting" [125] the pre-trained weights for player detection, we train on COCO and our whole dataset (including soccer and basketball data) concurrently with an equal sampling probability for 150k iterations ($3\times$ longer than all other experiments). The learning rate is decreased by 10 at the 120k-th iteration. Since we use all the data for training, we only show the qualitative results on images collected from the Internet. We choose several popular team sports for visualizations as in Figure 3.8. The qualitative results demonstrate the remarkable generalizability of JEDE across different sports, even though is it only trained on soccer and basketball domains. There are limitations of the proposed framework as shown in the last image in the figure, where the numbers on the players' heads are not detected. For uncommon poses and digit locations which do not present during training, JEDE can only reject low-confidence detections.

Limitations and solutions: JEDE achieves great performances on player detection, player pose estimation, and jersey number detection. However, the detections are not perfect as shown in the visualizations. We summarize the limitations and their solutions as follows:

- Our approach is data-driven, and the performance still suffer from the lack of extensive data. If more annotations on sports-related poses and jersey numbers are available, better results can be achieved.

- The jersey number detection performance relies on the accuracy of jersey number length

prediction. If the length is predicted incorrectly, the predicted jersey number will be wrong. As discussed in Section 3.5.4, if more training data of digits are provided, the number length predictor can be replaced with a sequence decoder like the one used in Mask TextSpotter V3 [81] and SwinTextSpotter [83].

- It is difficult for JEDE, object detection, or scene text detection framework to handle blur, occlusion, and small scale. JEDE can be deployed for real-time sports analysis. If the detection fails in a particular frame, we can still obtain correct detections in future frames. Moreover, in future work, JEDE can be extended for video analysis by integrating tracking for handling fast-changing conditions in sports fields.

Chapter 4

Fully Convolutional Scene Graph Generation

4.1 Introduction

Philosophers, linguists and artists have long wondered about the semantic content of what the mind perceives in images and speech [126, 127, 128, 129]. Many have argued that images carry layers of meaning [130, 131]. Considered as an engineering problem, semantic content has been modeled either as latent representations [132, 133, 134, 135], or explicitly as structured representations [136, 137, 138]. For a computer vision system to explicitly represent and reason about the detailed semantics, Johnson [139] *et al*. adopt and formalize *scene graphs* from computer graphics community. A scene graph serves as a powerful representation that enables many down-stream high-level reasoning tasks such as image captioning [140, 141], image retrieval [142, 139], Visual Question answering [143, 144] and image generation [142, 145].

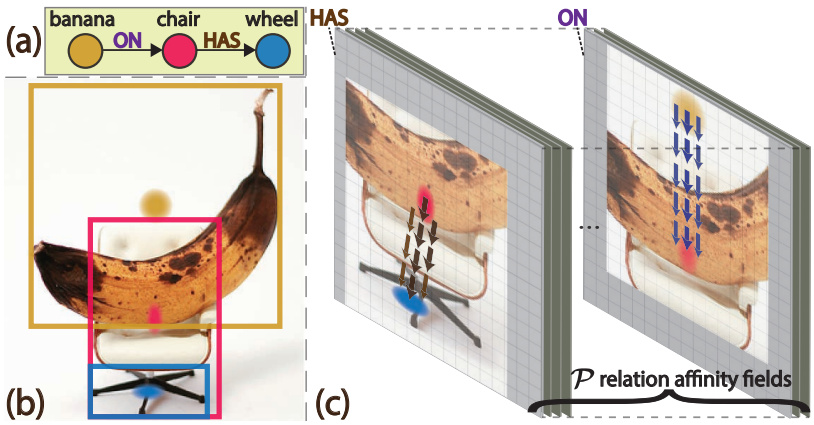


Figure 4.1: An example of scene graph generation. (a) The ground-truth scene graph of an image. (b) The ground-truth bounding boxes and their centers. (c) Our proposed relationship representation called relation affinity fields. (The image is `2353896.jpg` from Visual Genome [1].)

A *scene graph* is considered as an explicit structural representation for describing the semantics of a visual scene. The nodes in a scene graph represent the object classes and the edges represent the relationships between the objects. Figure 4.1(a) shows a simple example of a scene graph that represents the underlying semantics of an image. Each relationship between two objects is denoted as a triplet of `<subject, PREDICATE, object>` , i.e., `banana` $\xrightarrow{\text{ON}}$ `chair` and `chair` $\xrightarrow{\text{HAS}}$ `wheel` in Figure 4.1(a). Most of the SGG work [146, 147, 148, 138, 149] is build as a two-stage pipeline: object detection then scene graph generation. For the first stage of object detection, existing object detectors, *i.e.*, Fast/Faster R-CNN [39, 19], are used for object feature extraction from region proposals. For the second stage of scene graph generation, various approximation methods [148, 138, 149] for graph inference have been used. Some work [146, 150, 151, 152] have also investigated how to utilize external knowledge for improving the results. However, most previous work suffer from not only the long-tailed distribution of relationships [153, 154, 149], but also the highly biased prediction conditioned on object labels [155, 156, 157, 147]. Consequently, frequent predicates will prevail over less frequent ones, and unseen relationships can not

be identified. Moreover, the extensibility and inference speed of a SGG framework is crucial for accelerating down-stream tasks. Although few researchers have studied the efficiency and scalability in SGG [11, 158, 159], the high computational complexity impedes the practicality towards real-world applications. A natural question that arises is: *can we solve scene graph generation in a per-pixel prediction fashion?* Recently, anchor-free object detectors [102, 160, 161, 103] have become popular due to their simplicity and low cost. These methods treat an object as a single or many, pre-defined or self-learned keypoints. Relating object detection to human pose estimation, if an object can be modeled as a point (human "keypoint"), is it possible to represent a binary relationship as vectors (human "limb")?

In this chapter, we propose a novel fully convolutional scene graph generation model, *i.e.*, FCSGG, with state-of-the-art object detection results on Visual Genome dataset [1], as well as compelling SGG results compared with visual-only methods. We present a bottom-up representation of objects and relationships by modeling objects as points and relationships as vectors. Each relationship is encoded as a segment in a 2D vector field called *relation affinity field* (RAF). Figure 4.1(c) shows an illustration of RAFs for predicates `ON` and `HAS`. Both objects and relationships are predicted as dense feature maps without losing spatial information. For the first time, scene graphs can be generated from a single convolutional neural network (CNN) with significantly reduced model size and inference speed.

4.2 Related Work

We categorize the related work of SGG into the following directions: refinement of contextual feature, adaptation of external knowledge, and others.

Contextual feature refinement. Xu *et al*. [138] proposed an iterative message passing mechanism based on Gated Recurrent Units [162], where the hidden states are used for predictions. Followers [148, 149] studied better recurrent neural networks [114, 163, 164] for encoding object and edge context. Others trying to incorporate more spatial features into SGG. Li *et al*. [6] proposed the MSDN that merges features across multiple semantic levels, and later achieved message passing constrained on visual phrase [165]. Dai *et al*. [166] proposed a spatial module by learning from bounding box masks. Woo *et al*. [167] introduced the geometric embeddings by directly encoding the bounding box offsets between objects. Wang *et al*. [168] further studied the effects of relative positions between objects for extracting more discriminating features. Our method is fundamentally different from these methods as the relationships are grounded semantically and spatially directly into CNN features. Without any explicit iterative information exchange between nodes and edges, our model is able to predict objects and relationships in a single forward pass.

External knowledge adaptation. Beyond visual features, linguistic knowledge can serve as additional features for SGG [169, 136, 170, 150]. By adopting statistical correlations of objects, Chen *et al*. [146] utilized graph neural networks [171] to infer relationships. Gu *et al*. [172] and Zareian *et al*. [151, 152] explored the usefulness of knowledge or commonsense graphs for SGG. Tang *et al*. [147] proposed an de-biasing method by causal interventions of predictions. Lin *et al*. [173] investigated the graph properties and mitigated the long-tailed distributions of relationships. Compared with these methods, our proposed model relies only on visual features but still yields a strong performance.

Very few researchers have investigated alternatives either for object feature or relationship feature representations. Newell [174] and Zhang *et al*. [5] tried latent-space embeddings for rela-

tionship and achieved improvements. Different from most of the previous work, FCSGG reformulates and generalizes relationship representations from only visual-based features in near real-time, which is much faster than specifically designed SGG models for efficiency [158, 159].

4.3 Contributions of this Chapter

- We propose the first fully convolutional scene graph generation model that is more compact and computationally efficient compared to previous SGG models.
- We introduce a novel relationship representation called *relation affinity fields* that generalizes well on unseen visual relationships. FCSGG achieves strong results on zero-shot recall.
- Our proposed model outperforms most of the visual-only SGG methods, and achieves competitive results compared to methods boosted by external knowledge.
- We conduct comprehensive experiments and benchmark our proposed method together with several previous work on model efficiency, and FCSGG achieves near real-time inference.

4.4 Object Detection as Keypoint Estimation

In this section, we provide the preliminaries of modeling object detection as keypoint estimation in a single-scale dense feature map prediction fashion.

Our model is built upon a one-stage anchor-free detector, namely CenterNet [103]. Different from commonly used anchor-based R-CNN approaches for generating object proposals and features, it predicts three dense features that represent centers of object bounding boxes, center offsets, and object sizes. More specifically, an input image $\mathrm{I} \in \mathbb{R}^{3\times W\times H}$ will go through a backbone

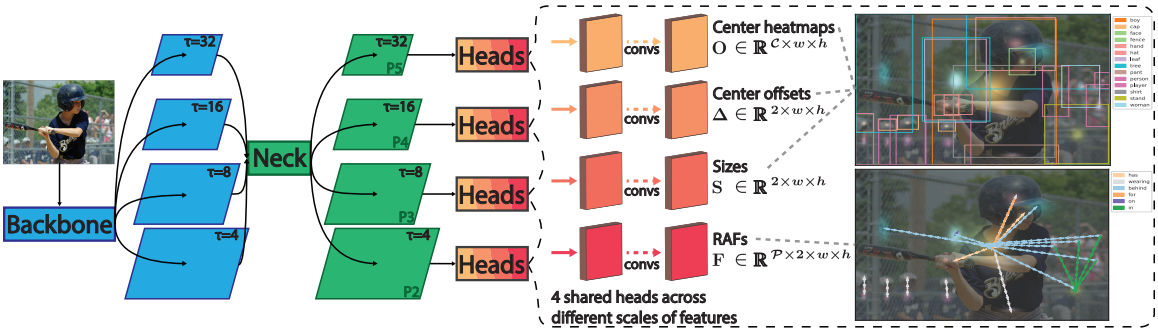


Figure 4.2: One example of our proposed fully convolutional scene graph generation architecture using four scales of features for prediction. We refer to the "backbone" as the feature extraction CNN like ResNet [2], and the "neck" as the network for generating multi-scale features like FPN [3], and the head as several convolutional layers (convs in figure). Shown in the right part, there are four output features per scale: $\mathbf{O}$, $\mathbf{\Delta}$, $\mathbf{S}$ for object detection and $\mathbf{F}$ for relationship detection. For single-scale prediction, the backbone features of $\tau = 4$ will be directly fed into the heads.

CNN generating feature maps of size $w \times h = \lfloor \frac{W}{\tau} \rfloor \times \lfloor \frac{H}{\tau} \rfloor$ where τ is the total stride until the last layer, and we set $\tau = 4$ unless specified otherwise. Then these features will be fed into three prediction heads, each of which consists of several convolutional layers. The three heads are for predicting object center heatmaps $\mathbf{O} \in \mathbb{R}^{\mathcal{C}\times w\times h}$ where $\mathcal{C}$ is the number of object classes in a dataset, object center offsets $\mathbf{\Delta} \in \mathbb{R}^{2\times w\times h}$ for recovering from downsampled coordinates, and object sizes $\mathbf{S} \in \mathbb{R}^{2\times w\times h}$, respectively (shown in the dashed block of Figure 4.2). We define the ground-truth (GT) objects in an image as $\mathbf{B} = \{b^i\}$ where $b^i = (x_0^i, y_0^i, x_1^i, y_1^i, c^i)$ is the object i of class c^i, (x_0^i, y_0^i) and (x_1^i, y_1^i) denote the coordinates of the left-top and right-bottom corners of its bounding box. The center of the bounding box is defined as $\mathbf{o}^i = (\mathbf{o}_x^i, \mathbf{o}_y^i) = ((x_0^i + x_1^i)/2, (y_0^i + y_1^i)/2)$, and the size of the object is defined as $\mathbf{s}^i = (x_1^i - x_0^i, y_1^i - y_0^i)$. To obtain the ground-truth center heatmaps at feature level, we divide coordinates by the stride τ and add Gaussian-smoothed samples following Law [102] and Zhou *et al.* [103]. Formally, the object center $\mathbf{o}^i$ will be modulated by a

bivariate Gaussian distribution along x-axis and y-axis on $\mathbf{O}_{c^i}$. The value around $\mathbf{o}^i$ is computed as

$$\mathbf{O}_{c^i,x,y} = \exp\big(- \frac{\|x - \lfloor \mathbf{o}^i_x/\tau \rfloor\|_2^2}{2\sigma_x^2} - \frac{\|y - \lfloor \mathbf{o}^i_y/\tau \rfloor\|_2^2}{2\sigma_y^2}\big), \tag{4.1}$$

where σ_x and σ_y controls the spread of the distribution. When multiple objects of the same class c contribute to $\mathbf{O}_{c,x,y}$, the maximum is taken as the ground truth. The center heatmaps are then supervised by Gaussian focal loss [102, 104, 103]. More details are provided in the supplementary file.

In addition to the supervision of center heatmaps, the center offset regression $\mathcal{L}_{\boldsymbol{\Delta}}$ and object size regression $\mathcal{L}_{\mathbf{S}}$ are used to recover object detections. For mitigating discretization error due to downsampling, the offset target is $\boldsymbol{\delta}^i = \mathbf{o}^i/\tau - \lfloor \mathbf{o}^i/\tau \rfloor$, and regressed via L1 loss as $\mathcal{L}_{\boldsymbol{\Delta}_{x,y}} = ||\hat{\boldsymbol{\Delta}}_{x,y} - \boldsymbol{\delta}^i||_1$ at center locations. For object size regression $\mathcal{L}_{\mathbf{S}}$, the target is feature-level object size $\mathbf{s}^i/\tau$, and the actual size can be recovered by multiplying the output stride. We also use L1 loss as $\mathcal{L}_{\mathbf{S}_{x,y}} = ||\hat{\mathbf{S}}_{x,y} - \mathbf{s}^i||_1$ at center locations. Both object size and offset regressors are class-agnostic such that there will be only one valid regression target at a particular location where $\mathbf{O}_{c,x,y} = 1$. If two object centers collide onto the same location, we choose the smaller object for regression. The overall object detection objective is

$$\mathcal{L}_{det} = \frac{1}{N} \sum_{c,x,y} \left(\mathcal{L}_{\mathbf{O}_{c,x,y}} + \lambda_{\boldsymbol{\Delta}} \mathcal{L}_{\boldsymbol{\Delta}_{x,y}} + \lambda_{\mathbf{s}} \mathcal{L}_{\mathbf{S}_{x,y}} \right), \tag{4.2}$$

where N is the total number of objects in the image, $\lambda_{\boldsymbol{\Delta}}$ and $\lambda_{\mathbf{s}}$ are hyper-parameters for weight balancing. We empirically set $\lambda_{\boldsymbol{\Delta}} = 1$ and $\lambda_{\mathbf{s}} = 0.1$ for all experiments. Until here, object centers, offsets, and sizes are all represented in single-scale feature maps. We will discuss a multi-scale prediction approach reducing regression ambiguity effectively in section 4.5.2.

4.5 Relation Affinity Fields

Newell and Deng [174] model objects as center points, and ground edges at the midpoint of two vertices then construct the graph via associative embeddings. The midpoint serves as a confidence measurement of presence of relationships. However, false detections and ambiguities arise when there are crowded objects in a region, or the associated objects of a relation are far away from each other. Another limitation is that it still needs feature extraction and grouping that cause low inference speed. Inspired and from a bottom-up 2D human pose estimation work called OpenPose [175], we migrate the concept of part affinity fields into scene graph generation. Our proposed method grounds relationships onto CNN features pixel by pixel, and mitigates above mentioned limitations.

Our model is conceptually simple: in addition to the outputs that are produced by the object detection network described in Section 4.4, we add another branch that outputs a novel feature representation for relationships called *relation affinity fields* (RAFs). Specifically, the RAFs are a set of 2D vector fields $\mathbf{F} = \{\mathbf{F}_p\} \in \mathbb{R}^{\mathcal{P} \times 2 \times h \times w}$, where $p \in \mathbb{R}^{\mathcal{P}}$ and $\mathcal{P}$ is the number of predicate classes in a dataset. Each 2D vector field $\mathbf{F}_p$ represents the relationships among all the object pairs of predicate p. Given our definition of objects as center points, the ground-truth RAFs are defined as vectors flow from the center of subject to the center of object. More formally, we define the binary relationships among objects $\mathbf{B}$ in the input image as $\mathbf{R} = \{r^{i \to j}\}$, where $r^{i \to j} = (b^i, p^{i \to j}, b^j)$ is the relationship triplet from subject b^i to object b^j with predicate $p^{i \to j}$. We define a "path" $\pi_p^{i \to j}$ that "propagates" $p^{i \to j}$ from subject center $\mathbf{o}^i$ to object center $\mathbf{o}^j$. For a point $\mathbf{p} = (x, y)$, its

ground-truth relation affinity field vector $\mathbf{F}_{p,x,y}$ is given as

$$\mathbf{F}_{p,x,y} = \begin{cases} \mathbf{e}^{i \to j} = \dfrac{\mathbf{o}^j - \mathbf{o}^i}{||\mathbf{o}^j - \mathbf{o}^i||_2} & \text{if } \mathbf{p} \in \pi_p^{i \to j} \\ 0 & \text{otherwise,} \end{cases} \tag{4.3}$$

and the path $\pi_p^{i \to j}$ is defined on a set of points between object centers forming a rectangular region:

$$\begin{aligned} \pi_p^{i \to j} = \{\mathbf{p} \,|\, 0 \leq \mathbf{e}^{i \to j} \cdot (\mathbf{p} - \mathbf{o}^i) \leq \epsilon_{\mathbf{e}^{i \to j}} \\ \text{and } |\mathbf{e}_\perp^{i \to j} \cdot (\mathbf{p} - \mathbf{o}^i)| \leq \epsilon_{\mathbf{e}_\perp^{i \to j}}\}, \end{aligned} \tag{4.4}$$

where $\epsilon_{\mathbf{e}^{i \to j}} = ||\mathbf{o}^j - \mathbf{o}^i||_2$ as the relationship "length" along the direction $\mathbf{e}^{i \to j}$, and $\epsilon_{\mathbf{e}_\perp^{i \to j}} = \min(a^i, b^i, a^j, b^j)$ as the relationship "semi-width" along $\mathbf{e}_\perp^{i \to j}$ (orthogonal to $\mathbf{e}^{i \to j}$) being the minimum of object centers' radii. Since vectors may overlap at the same point, the ground-truth RAF $\mathbf{F}_p$ averages the fields computed for all the relationship triplets containing that particular predicate p. It is given as $\mathbf{F}_p = \frac{1}{n_c(x,y)} \sum_{x,y} \mathbf{F}_{p,x,y}$, where $n_c(x, y)$ is the number of non-zero vectors at point (x, y). With the definition of ground-truth RAFs, we can train our network to regress such dense feature maps. The RAF regression loss $\mathcal{L}_{raf}$ can be estimated using a normal regression losses $\mathcal{L}_{reg}$ such as L1, L2 or smooth L1 [39]. Given the predicted RAFs $\hat{\mathbf{F}}$, the loss is defined as per-pixel weighted regression loss as

$$\mathcal{L}_{raf} = \mathbf{W} \cdot \mathcal{L}_{reg}(\hat{\mathbf{F}}, \mathbf{F}), \tag{4.5}$$

where $\mathbf{W}$ is a pixel-wise weight tensor of the same shape of $\mathbf{F}$. The weights $\mathbf{W}$ are determined and divided into three cases (Figure 4.3):

1. $\mathbf{W}_{p,x,y} = 1$ if (x, y) is exactly on the line segment between objects having the relationship p
2. $\mathbf{W}_{p,x,y} \in (0, 1)$ if the distance between (x, y) and the line segment is small and the value is negative correlated to the distance

3. otherwise where $\mathbf{F}_{p,x,y} = 0$.

We provide ablation study on the choice of losses and the weight tensor in Section 4.6.3. Finally, the complete loss for training our proposed model can be written as $\mathcal{L} = \mathcal{L}_{det} + \mathcal{L}_{raf}$.

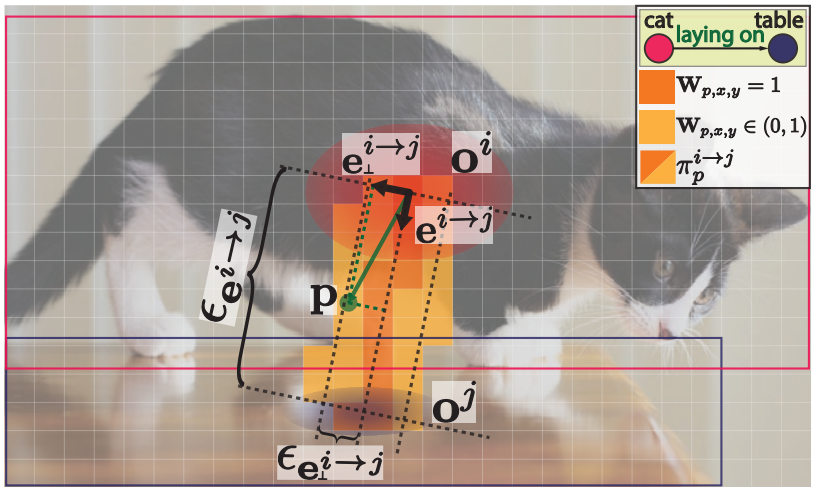


Figure 4.3: An example of GT relation affinity field of predicate `LAYING ON` based on equations 4.3 and 4.4. A non-zero unit vector is only defined on locations inside $\pi^{\texttt{cat}\to\texttt{table}}_{\texttt{LAYING ON}}$.

Our proposed RAFs encode rich information of both spatial and semantic features as dense feature maps, and enable end-to-end joint training of object detection and relation detection. To extract relationship from predictions, path integral over RAFs is performed which will be described below.

4.5.1 Inference

We compute path integrals over RAFs along the line segments connecting pairs of detected object centers as the scores of relationships. Specifically, for two candidate object centers $\hat{\mathbf{o}}^i$ and $\hat{\mathbf{o}}^j$ with predicted class scores $\hat{h}^i$ and $\hat{h}^j$, we gather the predicted RAFs $\hat{\mathbf{F}}$ along the path between $\hat{\mathbf{o}}^i$ and $\hat{\mathbf{o}}^j$, and compute the mean of their projections onto $\hat{\mathbf{e}}^{i \to j} = (\hat{\mathbf{o}}^j - \hat{\mathbf{o}}^i)/||\hat{\mathbf{o}}^j - \hat{\mathbf{o}}^i||_2$.

The path integral scores $\mathbf{E}^{i \to j}$ are identified as the confidences of existence of relationships:

$$\mathbf{E}^{i \to j} = \frac{\hat{h}^i \cdot \hat{h}^j}{m^{i \to j}} \sum_{p \in \mathbb{R}^{\mathcal{P}}} \sum_{(x,y) \in \pi^{i \to j}} \mathbf{F}_{p,x,y} \cdot \hat{\mathbf{e}}^{i \to j}, \tag{4.6}$$

where $m^{i \to j} = |\pi^{i \to j}|$ is the number of points in $\pi^{i \to j}$. Since our RAFs are object-class-agnostic, we multiply the class scores of the objects and the path integral score as the overall classification score for the relationship predicate. The integral will be performed spatially for each predicate channel, so $\mathbf{E}^{i \to j}$ represents the confidences of predicted relationship triplet $\hat{r}^{i \to j}$ for all predicates. Note that the integral could be negative that indicates an opposite relationship of the object pairs, and those negative integral values can be simply negated as $\mathbf{E}^{j \to i} = -\mathbf{E}^{i \to j}$. Finally, both scores of $\mathbf{E}^{i \to j}$ and $\mathbf{E}^{j \to i}$ will be used for ranking the predicted relationships. We also experiment with a simple re-weighting step known as frequency bias [149] by multiplying $\mathbf{E}^{i \to j}$ with $1.0001^{n_c(\hat{r}^{i \to j})}$, where $n_c(\hat{r}^{i \to j})$ counts the occurrence of triplet $r^{i \to j}$ in training set. The path integral procedure is presented in Algorithm 2. In practice, the operations are performed using matrix multiplication in stead of FOR LOOP for fast inference.

4.5.2 Multi-scale Prediction

Since object centers are downsampled to feature level, their centers could collide onto the same pixel location. Regression ambiguity may rise due to single-scale feature representations. We address this problem by utilizing multi-scale prediction and shared detection heads.

Though Zhou *et al*. [103] argued that only a very small fraction (<0.1% in COCO [48] dataset) of objects have center collision problem at stride of 4, the size and offset regression targets need better assignment strategy since there is only one valid target per pixel. We follow the work of FPN [3], RetinaNet [104] and FCOS [160], and assign the ground-truth bounding boxes to different

Algorithm 2 Path Integral over Relation Affinity Fields

Input: Object centers $\{\hat{\mathbf{o}}^i\}$, Relation Affinity Fields $\hat{\mathbf{F}}$

Output: Relations $\hat{\mathbf{R}} = \{\hat{r}^{i\to j} = (i, \mathbf{E}^{i\to j}, j) | i \neq j\}$

$\hat{\mathbf{R}} \leftarrow \{\}$

for each center pair $(\hat{\mathbf{o}}^i, \hat{\mathbf{o}}^j) | i \neq j$ **do**

  $\pi^{i\to j} \leftarrow$ LINSPACE($\hat{\mathbf{o}}^i, \hat{\mathbf{o}}^j$) $\in \mathbb{R}^{m^{i\to j}\times 2}$

  $\hat{\mathbf{F}}_{\pi^{i\to j}} \leftarrow$ INDEX($\hat{\mathbf{F}}$, $\pi^{i\to j}$) $\in \mathbb{R}^{\mathcal{P}\times m^{i\to j}\times 2}$

  $\mathbf{E}^{i\to j}, \mathbf{E}^{j\to i} \in \mathbb{R}^{\mathcal{P}} \leftarrow$ Equation 4.6

  $\hat{\mathbf{R}} \leftarrow \hat{\mathbf{R}} \bigcup \{\hat{r}^{i\to j} = (i, \mathbf{E}^{i\to j}, j)\}$

  $\bigcup \{\hat{r}^{j\to i} = (j, \mathbf{E}^{j\to i}, i)\}$

end for

return SORTED($\hat{\mathbf{R}}$)

levels based on scales. Building upon the backbone features, we construct multi-level feature maps $\{\mathrm{P}_k\}$ where P_k is of stride 2^k. We refer the network component of generating multi-scale features as the "neck" (the green box in Figure 4.2), such as FPN [3]. We define a valid range $[l_k, u_k] \subset [0, \mathrm{L}_{max}]$ for objects in each scale, where L_{max} is the maximum size of longer edge allowed for training and testing. Only bounding boxes of area within $[l_k^2, u_k^2]$ are qualified for the k-th scale training. We experiment different number of scale levels and input image size. For smaller input image of L_{max} = 512, we build 4-scale features [3] $\{\mathrm{P}_2, \mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5\}$ (as shown in Figure 4.2) with valid ranges {[0, 32^2], [32^2, 64^2], [64^2, 128^2], [128^2, 512^2]}; for larger input image of L_{max} = 1024 (shorter edge is at least L_{min} = 640), we use 5-scale features [104, 160] $\{\mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5, \mathrm{P}_6, \mathrm{P}_7\}$ with area ranges {[0, 64^2], [64^2, 128^2], [128^2, 256^2], [256^2, 512^2], [512^2, 1024^2]}. If there is still more

than one target at the same location, we simply choose the smallest object for regression.

In terms of the multi-scale RAFs training, the GT assignment is based on the distances between object centers. For high-level semantic features like P_5, the feature map can capture large objects, as well as relationships among distant objects. We select the relationships of "length" $\epsilon_{\mathbf{e}^{i\to j}} \in [l_k, u_k]$ as valid samples for training the k-th scale. The exact ranges for 4-scale or 5-scale RAFs are the same as the settings for bounding boxes. Finally, the weights of detection heads are shared across different feature scales for efficiencies and performance improvements. During inference, we gather outputs from each scale based on the corresponding valid range, then merge and rank all relationship triplets. Figure 4.2 illustrates the details of our proposed architecture using a four-scale feature setup as an example with shared detection heads. Our experiments (Section 4.6) show that the multi-scale GT and scale-aware training resolve the aforementioned ambiguity problem thus improve the results over single-scale prediction.

4.6 Experiments

Dataset. We use the Visual Genome (VG) [1] dataset to train and evaluate our models. We followed the widely-used preprocessed subset of VG-150 [138] which contains the most frequent 150 object categories ($\mathcal{C}$ = 150) and 50 predicate categories ($\mathcal{P}$ = 50). The dataset contains approximately 108k images, with 70% for training and 30% for testing. Different from previous works [176, 148, 149], we do not filter non-overlapping triplets for evaluation.

General settings. We experiment on two settings, one for small input size (L_{max} = 512) and one for larger size (L_{max} = 1024). The model is trained end-to-end using SGD optimizer with

Table 4.1: Recall and no-graph constraint recall @K evaluation results on VG-150. ⋆ denotes the methods evaluated on other datasets, such that VTransE is evaluated on VG-200 [5] and FactorizableNet on a smaller set following [6]. † denotes the methods with updated re-implementation results. - denotes the results that are not reported in the corresponding work.

Recall @K / No-graph Constraint Recall @K			AP_{50}	Predicate Classification		Scene Graph Classification		Scene Graph Detection	
				R@20/50 /100	ng-R@20/50/100	R@20/50/100	ng-R@20/50/100	R@20/50/100	ng-R@20/50/100
External Knowledge		VCTree [148]	-	60.1/ 66.4/ 68.1	-	35.2/ 38.1/ 38.8	-	22.0/ 27.9/ 31.3	-
		KERN [146]	-	-/ 65.8/ 67.6	-/ 81.9/ 88.9	-/ 36.7/ 37.4	-/ 45.9/ 49.0	-/ 27.1/ 29.8	-/ 30.9/ 35.8
		GPS-Net [173]	-	67.6/ 69.7/ 69.7	-	41.8/ 42.3/ 42.3	-	22.3/ 28.9/ 33.2	-
		MOTIFS-TDE [147, 149]	28.1	33.6/ 46.2/ 51.4	-	21.7/ 27.7/ 29.9	-	12.4/ 16.9/ 20.3	-
		GB-NET-β [151]	-	-/ 66.6/ 68.2	-/ 83.5/ 90.3	-/ 37.3/ 38.0	-/ 46.9/ 50.3	-/ 26.3/ 29.9	-/ 29.3/ 35.0
Visual Only		VTransE⋆ [5]	-	-	-	-	-	-/ 5.5/ 6.0	-
		FactorizableNet⋆ [158]	-	-	-	-	-	-/ 13.1/ 16.5	-
		IMP† [138, 149]	20.0	58.5/ 65.2/ 67.1	-	31.7/ 34.6/ 35.4	-	14.6/ 20.7/ 24.5	-
		Pixels2Graphs [174]	-	-	-/ 68.0/ 75.2	-	-/ 26.5/ 30.0	-	-/ 9.7/ 11.3
		Graph R-CNN [159]	23.0	-/ 54.2/ 59.1	-	-/ 29.6/ 31.6	-	-/ 11.4/ 13.7	-
		VRF [154]	-	-/ 56.7/ 57.2	-	-/ 23.7/ 24.7	-	-/ 13.2/ 13.5	-
		CISC [168]	-	42.1/ 53.2/ 57.9	-	23.3/ 27.8/ 29.5	-	7.7/ 11.4/ 13.9	-
FCSGG (Ours)		HRNetW32-1S	21.6	27.6/ 34.9/ 38.5	32.2/ 46.3/ 56.6	12.3/ 15.5/ 17.2	13.5/ 19.3/ 23.6	11.0/ 15.1/ 18.1	12.4/ 18.2/ 23.0
		HRNetW48-1S	25.0	24.2/ 31.0/ 34.6	28.1/ 40.3/ 50.0	13.6/ 17.1/ 18.8	14.2/ 19.6/ 24.0	11.5/ 15.5/ 18.4	12.7/ 18.3/ 23.0
		ResNet50-4S-FPN$_{\times 2}$	23.0	28.0/ 35.8/ 40.2	31.6/ 44.7/ 54.8	13.9/ 17.7/ 19.6	14.8/ 20.6/ 25.0	11.4/ 15.7/ 19.0	12.2/ 18.0/ 22.8
		HRNetW48-5S-FPN$_{\times 2}$	28.5	28.9/ 37.1/ 41.3	34.0/ 48.1/ 58.4	16.9/ 21.4/ 23.6	18.6/ 26.1/ 31.6	13.5/ 18.4/ 22.0	15.4/ 22.5/ 28.3
		HRNetW48-5S-FPN$_{\times 2}$-f	28.5	33.4/ 41.0/ 45.0	37.2/ 50.0/ 59.2	19.0/ 23.5/ 25.7	19.6/ 26.8/ 32.1	16.1/ 21.3/ 25.1	16.7/ 23.5/ 29.2

the batch size of 16 for 120k iterations. The initial learning rate is set to 0.02 and decayed by the factor of 10 at 80k$^{\text{th}}$ and 100k$^{\text{th}}$ iteration. We adopt standard image augmentations of horizontal flip and random crop with multi-scale training. During testing, we keep the top 100 detected objects for path integral.

Metrics. We conduct comprehensive analysis following three standard evaluation tasks: Predicate Classification (PredCls), Scene Graph Classification (SGCls), and Scene Graph Detection (SGDet). We report results of recall@K (R@K) [136], no-graph constraint recall@K (ng-R@K) [174, 149], mean recall@K (mR@K) [146, 148], no-graph constraint mean recall@K (ng-

Table 4.2: The SGG results on mean recall@K and no-graph constraint mean recall@K.

Mean Recall @K /	Predicate Classification		Scene Graph Classification		Scene Graph Detection	
Ng Mean Recall @K	mR@20/50/100	ng-mR@20/50/100	mR@20/50/100	ng-mR@20/50/100	mR@20/50/100	ng-mR@20/50/100
VCTree [148]	14.0 / 17.9 / 19.4	-	8.2 / 10.1 / 11.8	-	5.2 / 6.9 / 8.0	-
KERN [146]	- / 17.7 / 19.4	-	- / 9.4 / 10.0	-	- / 6.4 / 7.3	-
GPS-Net [173]	- / - / 22.8	-	- / - / 12.6	-	- / - / 9.8	-
MOTIFS-TDE [147, 149]	18.5 / 25.5 / 29.1	-	9.8 / 13.1 / 14.9	-	5.8 / 8.2 / 9.8	-
GB-NET-β [151]	- / 22.1 / 24.0	-	- / 12.7 / 13.4	-	- / 7.1 / 8.5	-
HRNetW32-1S	4.0 / 5.5 / 6.3	5.4 / 9.7 / 13.6	1.9 / 2.5 / 2.8	2.7 / 4.4 / 6.2	1.7 / 2.4 / 2.9	2.2 / 3.6 / 4.9
HRNetW48-1S	3.7 / 5.2 / 6.1	5.2 / 9.5 / 14.7	2.2 / 2.9 / 3.4	3.5 / 6.3 / 9.4	1.8 / 2.6 / 3.1	2.7 / 4.7 / 6.9
ResNet50-4S-FPN$_{\times 2}$	4.2 / 5.7 / 6.7	6.5 / 11.3 / 16.6	2.2 / 2.9 / 3.3	3.6 / 6.0 / 8.3	1.9 / 2.7 / 3.3	3.0 / 4.9 / 6.8
HRNetW48-5S-FPN$_{\times 2}$	4.3 / 5.8 / 6.7	6.1 / 10.3 / 14.2	2.6 / 3.4 / 3.8	4.1 / 6.4 / 8.4	2.3 / 3.2 / 3.8	3.7 / 5.7 / 7.4
HRNetW48-5S-FPN$_{\times 2}$-*f*	4.9 / 6.3 / 7.1	6.6 / 10.5 / 14.3	2.9 / 3.7 / 4.1	4.2 / 6.5 / 8.6	2.7 / 3.6 / 4.2	3.8 / 5.7 / 7.5

mR@K), zero-shot recall@K (zsR@K) [136] and no-graph constraint zero-shot recall@K (ng-zsR@K) [147] for all three evaluation tasks. We do not train separate models for different tasks.

4.6.1 Implementation Details

We conduct experiments on different backbone and neck networks. Each of the detection heads consists of four 3×3 convolutions followed by batch normalization and ReLU, and one 1×1 convolution with the desired number of output channels in all our experiments unless specified otherwise. For convenience, our models are named as BACKBONE - # OF OUTPUT SCALES - NECK - OTHER OPTIONS.

ResNet [2, 3]. We start by using ResNet-50 as our backbone and build a 4-scale FPN for multi-scale prediction. Since the tasks of object detection and RAFs prediction are jointly trained, the losses from the two tasks could compete with each other. We implement a neck named "FPN$_{\times 2}$" with two parallel FPNs, such that one FPN is used for constructing features for object detection

heads (center, size and offset), and the other is for producing features for RAFs. We name this model as ResNet50-4S-FPN$\times 2$.

HRNet [8]. We then experiment on a recent proposed backbone network called HRNet that consists of parallel convolution branches with information exchange across different scales. For single-scale experiments, we use HRNetV2-W32 and HRNetV2-W48; and for multi-scale prediction, we adopt its pyramid version called HRNetV2p. We omit their version number for the rest of the paper. We test several models: HRNetW32-1S, HRNetW48-1S and HRNetW48-5S-FPN$\times 2$. We also experiment on adding frequency bias for inference as discussed in Section 4.5.1, and the model used is called HRNetW48-5S-FPN$\times 2$-*f*.

4.6.2 Quantitative Results

Table 4.3: Comparisons of SGG results on zero-shot Recall@K, and our results on no-graph constraint zero-shot Recall@K.

Zero-shot Recall @K	PredCls		SGCls		SGDet	
Method	zsR@50/100		zsR@50/100		zsR@ 50/100	
MOTIFS-TDE [147]	14.4 / 18.2		3.4 / 4.5		2.3 / 2.9	
VTransE-TDE [147]	13.3 / 17.6		2.9 / 3.8		2.0 / 2.7	
VCTree-TDE [147]	14.3 / 17.6		3.2 / 4.0		2.6 / 3.2	
Knyazev *et al*. [157]	- / 21.5		- / 4.2		- / -	
FCSGG (Ours)	zsR @50/100	ng-zsR @50/100	zsR @50/100	ng-zsR @50/100	zsR @50/100	ng-zsR @50/100
HRNetW32-1S	8.3 / 10.7	12.9 / 19.2	1.0 / 1.2	2.3 / 3.5	0.6 / 1.0	1.2 / 1.6
HRNetW48-1S	8.6 / 10.9	12.8 / 19.6	1.7 / 2.1	2.9 / 4.4	1.0 / 1.4	1.8 / 2.7
ResNet50-4S-FPN$\times 2$	8.2 / 10.6	11.7 / 18.1	1.3 / 1.7	2.4 / 3.8	0.8 / 1.1	1.0 / 1.7
HRNetW48-5S-FPN$\times 2$	7.9 / 10.1	11.5 / 17.7	1.7 / 2.1	2.8 / 4.8	0.9 / 1.4	1.4 / 2.4
HRNetW48-5S-FPN$\times 2$-*f*	7.8 / 10.0	11.4 / 17.6	1.6 / 2.0	2.8 / 4.8	0.8 / 1.4	1.4 / 2.3

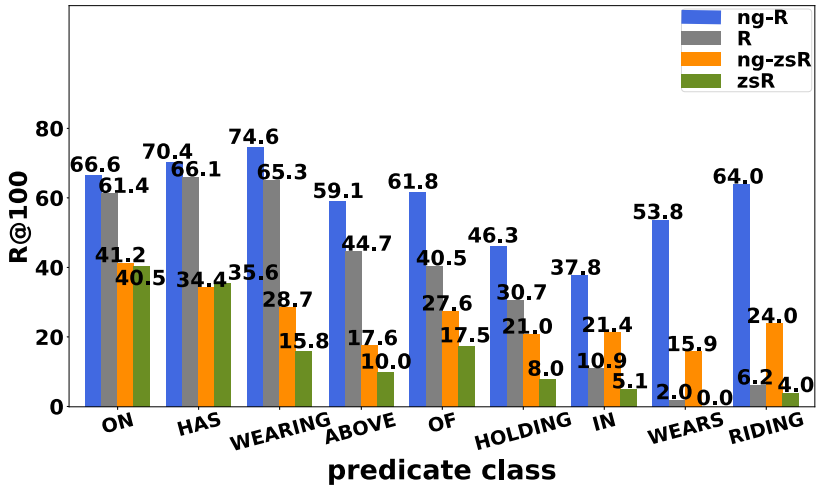


Figure 4.4: FCSGG per-predicate PredCls@100 results for selected predicates using HRNetW48-5S-FPN$\times$2-*f*.

We first compare results of R@K and ng-R@K with various of state-of-the-art (SOTA) models, and divide them into two categories:

1. models that only use visual features derived from the input image like our proposed model
2. models that not only use visual features, but also use features like language embeddings, dataset statistics or counterfactual causality, *etc*.

The results are shown in Table 4.1. Our best model achieves 28.5 average precision at IoU = 0.5 (AP_{50}) for object detection. Though our SGG results do not outperform the SOTA approaches, we achieve the best scene graph detection results among visual-only models. Specifically, Pixels2Graphs [174] and our models are the only models without using Faster R-CNN [19] as object detector, and we achieves 13.8 / 17.9 gain on SGDet ng-R@50 / 100 compared with their results using RPN [19]. We then report mean recall and no-graph constraint mean recall results shown in Table 4.2. We still obtain competitive results, especially on ng-mR. By comparing (m) R@K and ng-(m) R@K directly, we observe more gain on our results than other methods'.

Zero-shot recall (zsR) [155, 156, 157, 136, 170, 147, 177] is a proper metric for evaluating the model's robustness and generalizability for generating scene graphs. It computes recall on those subject-predicate-object triplets that do not present during training. There are in total of 5971 unique zero-shot triplets from the testing set of VG-150. The results and comparisons are listed in Table 4.3. We also compute the per-predicate recall@100 for predicate classification task using HRNetW48-5S-FPN$\times$2-*f*, and show the comparisons in Figure 4.4. We observe similar behavior with our results on recall, such that the results on no-graph constraint zero-shot recall are significantly better than zero-shot recall. Even for unseen triplets, purely based on visual features, FCSGG is still capable of predicting meaningful RAFs which proves its generalization capability. In other words, when constructing scene graphs from RAFs, our approach does not highly depend on the object classes but only focuses on the context features between the objects. When comparing with other reported results on zsR, we achieve slightly lower results than those much larger models. For example of PredCls task, ResNet50-4S-FPN$\times$2 achieves 10.6 zsR@100 with only 36 million (M) number of parameters and inference time of 40 milliseconds (ms) per image, while VCTree-TDE [147] achieves 17.6 zsR@100 with 360.8M number of parameters and inference time of 1.69 s per image. For comparison, ResNet50-4S-FPN$\times$2 is *10 times* smaller and *42 times* faster than VCTree-TDE.

Limitations. It should be noticed that FCSGG also has some "disadvantages" over Faster R-CNN-based methods on easier tasks such as PredCls and SGCls. During evaluations with given GT bounding boxes or classes, our RAFs features will not change, while R-CNN extracted object/union-box features will change which leads to better results. When using visual-only representation of relationships, it is hard for the network to distinguish predicates between `WEARS` / `WEARING` (by comparing R and ng-R in Figure 4.4) or `LAYING ON` / `LYING ON`, which is common in VG

dataset. In this sense, incorporating external knowledge gives FCSGG a large potential in improving results. Comparing the model HRNetW48-5S-FPN$\times 2$ and its frequency-biased counterpart HRNetW48-5S-FPN$\times 2$-*f*, we find noticeable improvement by using training set statistics. This simple cost-free operation can improve R@20 by 2.6, and we expect better results from fine-tuning hyper-parameters. However, the focus of this work is not perfectly fitting on a dataset, but improving generalization of relationship based on visual features. More sophisticated ensemble methods or extensions are beyond the scope of this paper.

4.6.3 Ablation Study

4.6.3.1 RAF regression Loss

We experience the same difficulty of training from sparsely annotated scene graphs as discussed by Newell *et al*. [174]. The network has the potential of generating reasonable triplets not covered in the ground-truth, and our results on zero-shot recall prove the argument. To reduce the penalty on these detections, we investigate the design methodology of RAF regression loss $\mathcal{L}_{raf}$ (Equation 4.5).

We refer the loss applied at locations having GT RAFs defined as positive loss $\mathcal{L}^{+}_{raf}$, and we test different regression losses. As for locations where $\mathbf{F}_{p,x,y} = 0$, we apply so called negative loss $\mathcal{L}^{-}_{raf}$ using L1 for regression. $\mathcal{L}^{-}_{raf}$ will be multiplied by a factor β for adjusting the penalty. Spatially, $\mathcal{L}_{raf}$ can be re-written as $\mathcal{L}_{raf} = \mathcal{L}^{+}_{raf} + \beta\mathcal{L}^{-}_{raf}$. Table 4.4 shows the effects of different losses and penalty factor on the performance. We observe better performance using L1 and β = 10. However, when only supervise on $\mathcal{L}^{+}_{raf}$ loss, the model has comparable mean recall results and it can detect more semantic and rare relationships. On the other hand, adding $\mathcal{L}^{-}_{raf}$ loss will push the

Table 4.4: Ablations on losses used for positive samples and regularization factor on negative samples of RAFs. AP_{50} and SGDet results are reported using HRNetW32-1S.

$\mathcal{L}^{+}_{raf}$	β	AP_{50}	R@50	zR@50	mR@50
L1	0	21.57	6.22	0.40	2.28
L1	1	21.52	9.80	0.56	2.33
L1	10	21.56	15.05	0.60	2.36
Smooth L1	0	20.15	5.00	0.30	2.51
Smooth L1	1	19.65	7.46	0.57	2.45
Smooth L1	10	20.63	11.82	0.61	2.83
L2	0	19.62	4.82	0.26	2.34
L2	1	21.60	10.76	0.68	2.57
L2	10	21.62	2.89	0.57	2.50

model more biased to dominating predicates like `ON` and `HAS`.

4.6.3.2 Architecture Choices

Table 4.5: Comparisons of FPN *s* FPN$\times 2$, and Multi-scale batch normalization *s* group normalization. AP_{50} and SGDet results are reported using ResNet50-4S.

Neck	Norm	AP_{50}	R@50	zR@50	mR@50
FPN	GN	22.75	11.29	0.71	2.95
FPN	MS-BN	22.10	13.23	0.75	2.67
FPN$\times 2$	GN	22.74	11.96	0.78	3.00
FPN$\times 2$	MS-BN	22.60	12.01	0.80	2.88

By comparing our results between single-scale and multi-scale models, we see substantial performance gain on both object detection and scene graph generation from Table 4.1 4.2 4.3.

We also observe HRNet has better results over ResNet due to its multi-scale feature fusions. For investigating the entanglement of object features and contextual features producing RAFs, we compare the results of FPN and FPN$_{\times 2}$ using ResNet-50 as backbone shown in Table 4.5. As observed in [160], the regression range differs across different levels. Therefore, to improve the performance of shared fully-convolutional heads, we replace each batch normalization (BN) [122] layer in the head with a set of BN layers, each of which is only applied for the corresponding scale. We name this modified BN as multi-scale batch normalization (MS-BN). We also experiment on group normalization (GN) [123] which stabilizes the training as well. We show the comparisons of MS-BN and GN (Section 4.5.2) in the same table. We observe better mR by using FPN$_{\times 2}$ and better recall and zsR by using MS-BN. We expect more improvement with a larger batch size with MS-BN.

Table 4.6: Model size and speed comparisons for SGDet.

Method	#Params (M)	Input Size	s / image
Pixels2Graphs [5]	94.8	512×512	3.55
VCTree-TDE [147]	360.8	600×1000	1.69
MOTIFS-TDE [147]	369.5	600×1000	0.87
KERN [146]	405.2	592×592	0.79
MOTIFS [147]	367.2	600×1000	0.66
FactorizableNet [158]	40.4	600×1000	0.59
VTransE-TDE [147]	311.6	600×1000	0.55
GB-NET-β [151]	444.6	592×592	0.52
Graph R-CNN [159]	80.2	800×1024	0.19
FCSGG (Ours)			
HRNetW32-1S	47.3	512×512	0.07
HRNetW48-1S	86.1	512×512	0.08
ResNet50-4S-FPN$\times 2$	36.0	512×512	0.04
HRNetW48-5S-FPN$\times 2$	87.1	640×1024	0.12
HRNetW48-5S-FPN$\times 2$-f	87.1	640×1024	0.12

4.6.3.3 Model Size and Speed

We also conduct experiments on the model size and inference speed. Few work benchmarked on efficiency of scene graph generation previously [158, 151]. Though scene graphs are powerful, it is almost not possible to perform SGG and down-stream tasks in real-time due to significantly increased model complexity. FCSGG alleviates the computational complexity effectively. Our experiments are performed on a same NVIDIA GeForce GTX 1080 Ti GPU with inference batch size of 1. For comparisons, we include several previous work by running corresponding open-source codes under the same settings. The results are shown in Table 4.6. Both the number of parameters and inference time are considerably lower for FCSGG models. It is worth noting that the

computation overhead is from the backbone network. The path integral (Algorithm. 2) is performed pair-wisely for all 100 kept objects across five scales, which results in $\binom{100}{2} \times 2 \times 5 = 49500$ maximum number of candidate relationships for an image. The inference time for path integral is almost invariant over the number of instances as analyzed by Cao *et al*. [175]. We believe that object relationships exist universally, especially geometric ones. By grounding the full graph in RAFs as intermediate features, richer semantics can be retained for down-stream tasks. More importantly, convolution is hardware-friendly, and the model size is kept small for deployment on edge devices. We anticipate that real-time mobile SGG can be performed in the near future.

Chapter 5

RepSGG: Novel Representations of Entities and Relationships for Scene Graph Generation

5.1 Introduction

To understand a scene, it is important to infer underlying properties of entities and the relationships between them. For a computer vision system to explicitly represent and reason about the detailed semantics, Johnson [139] *et al*. adopt and formalize scene graphs from computer graphics community. A scene graph is an explicit graph representation for modeling a visual scene, where entities are the nodes, and pairwise relationships are represented as edges. A relationship between two entities is denoted as a triplet of `<subject, predicate, object>`. In

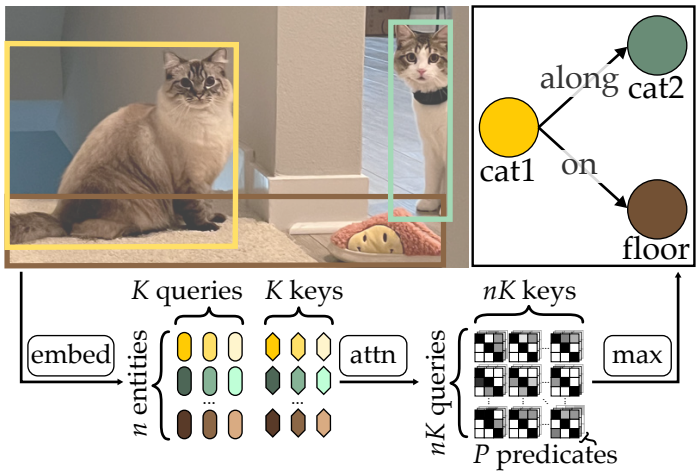


Figure 5.1: Illustration of RepSGG. For n detected entities, each entity is represented by K subject queries and K object keys. The attention weights between queries and keys are projected as the predicate classification scores in the shape of $P \times nK \times nK$, where P is the number of predicates in a dataset. The final predicate classification is reduced to the shape of $P \times n \times n$ by max-pooling, and top predictions are collected as the scene graph. $K = 3$ and $n = 3$ in the example.

the context of this paper, the term "entity" denotes an instance of an object, while the term "object" specifically indicates an entity with semantic significance. Serving as a powerful representation, scene graph enables many down-stream high-level reasoning tasks such as image captioning [140, 141], image retrieval [142, 139], Visual Question answering [143, 144, 178] and image generation [142, 145]. Since SGG is built upon object detection where many off-the-shelf detectors [39, 19, 179, 20, 103, 180] can be used, limited attention has been directed towards the investigation of better feature representations for entities and relationships. Currently, there are mainly three types of entity visual feature representations:

1. *Box-based*: spatially-pooled features extracted from bounding boxes. Most SGG methods [138, 147] are based on R-CNN detectors [39, 19, 20] which use RoIPooling [39] or RoIAlign [20] for feature extraction.

2. *Point-based*: single-pixel features extracted from bounding box centers. These methods [174, 7, 9] utilize anchor-free detectors [179, 103, 180] to ground entities and relationships in a regression fashion.

3. *Query-based*: fixed-size learnable embeddings. These methods [181, 182, 183] build upon DETR [184] where message passing and matching are performed between entity and relationship embeddings.

Each type has its own merits and drawbacks. While preserving the spatial appearance of entities, box-based features are computationally expensive and use more memory. Point-based features are often regressed and extracted at the center of an entity's bounding box (referred to as entity center for the rest of the paper). Although such models achieve fast inference speed, their performance is relatively low due to the fact that point-based features are less semantically meaningful with limited cardinality. Query-based representation relies on a fixed number of learnable embeddings for decoding entities or relations, which considerably simplifies the SGG tasks. However, it suffers from training difficulties and lower performance compared to box-based methods.

Another issue of box-based, point-based and query-based entity representations lies in their predetermined granularities. Box-based features could be coarse and redundant, while point-based and query-based features are insufficient to represent entities with different semantics. For example, if there are two relation for the same entity `person`, `<person, eating, pizza>` and `<person, on, street>`, it is important to capture the context around the mouth and hands of the person for the first triplet, and around the feet and street for the second one. Box-based representation keeps the information of the entity `person`, but loses the fine-grained features around the mouth, hands and feet due to the low pooling resolution (*e.g.* 7×7). The same issue will arise for

point-based and query-based representations. Increasing the pooling resolution, feature dimension, or number of queries could help, but the computation complexity will increase dramatically.

Regarding the relation representations, most works use compositional contextual features to perform predicate classification. Few researchers [9, 7] have explored different ways to represent relations in a regression fashion. These methods represent entities as keypoints (*e.g.*, entity centers), and relationships as points [174] or vectors [7, 9]. By regressing and grounding entities and relations geometrically, these methods achieve faster inference speed which is useful for down-stream tasks. However, regression with handcrafted targets [7, 9] is deficient, especially on sparsely-annotated datasets [1]. Consequently, the performance of regression is inferior compared to predicate classification.

In this work, we seek new insights into alternative representations of entities and relationships for scene graph generation. We propose a novel architecture called RepSGG that is built upon the FCOS [180] entity detector with a specialised transformer-based [185, 184, 186] relationship encoder. As shown in Fig. 5.1, an entity class is represented with a set of learnable subject and object embeddings, named rep-embeddings, to learn diverse semantics. The subject and object embeddings are progressively augmented by dynamically sampled visual features and inter-embedding attentions. First, each embedding is updated by visual features sampled at semantically-dependent representative points (rep-points); then, a two-way cross-attention (subject-to-object and vice-versa) is performed to update both subject and object embeddings. For relationship prediction, subject and object embeddings are treated as queries and keys, with relationships quantified as the projected attention weights between these queries and keys.

Besides the proposed architecture, we also investigate the long-tailed problem in SGG.

Inspired by logit adjustment [187], we propose a run-time performance-guided logit adjustment (PGLA) to achieve per-instance label-dependent loss modification. We measure and update the performance (*e.g.*, recall or precision) of predicate predictions per mini-batch, per iteration during training. Instead of adding a bias term to logits as in [187], we perform the affine transformation on the logits. We also measure run-time logit differences among predicates, named confusion logits, to further enlarge inter-class logit margins. For each predicate, the among of the adjustment will be determined by its frequency in the training set, run-time performance, and confusion logits.

5.2 Related Work

The task of scene graph generation involves numerous aspects, and our focus lies on entity and relation representations, as well as the long-tailed problem.

5.2.1 Feature Representations

As discussed in Section 5.1, the entity representation is either box-based, point-based, or query-based. Traditional SGG methods [138, 149, 148] utilize a pre-trained detector [19, 20] to extract a set of entity bounding boxes and their corresponding feature maps via feature pooling (RoIPool [19] or RoIAlign [20]). The visual features for each entity, commonly referred to as appearance features, are represented as a tensor of shape $c \times h \times w$, where c is the number of channels of the feature, and $h \times w$ is the feature spatial size. Those features are used to construct visual context for predicate classification. To incorporate the relative position between entities, researchers use the geometric layout encoding [167], union of bounding boxes mask encoding [166], or geometric constraints [168] for a better visual representation.

One-stage anchor-free entity detectors [102, 103, 180] have recently gained popularity due to their simplicity and efficiency. In these works, entities are directly regressed at pixels in feature maps, where the entity center or corners are selected as the ground-truth targets. Instead of pooling features of various shapes, extracting features at multiple pixels is much faster and consumes less memory. Several works [174, 7, 9] explore such point-based entity representation for SGG. Pixel2Graph [174] grounds edges at the midpoints between the bounding box centers of subjects and objects (referred to as subject and object centers for the rest of the paper). FCSGG [7] uses relation affinity fields to encode the relations as 2D vectors "flow" from the subject to object centers. CoRF [9] extends the concept of fields by composing more regression targets per pixel.

Recent works in scene graph generation have explored transformer-based models to improve the performance. Several works [147, 188, 189] start to replace the RNN-based context decoders [149, 148] with multi-head self-attention [185]. Subsequently, other works [190, 191, 192] explore ways to construct subject, object and predicate queries with variants of transformers. With the success of DETR [184], more works [181, 182, 183] study the query-based representations of entities and relations. For DETR-like approaches, there are a fixed number of learnable entity and predicate queries, which will be decoded as output triplets in an end-to-end manner.

The strengths and weaknesses of different types of entity representation vary based on their granularity and flexibility. For example, box-based features are extracted via RoIAlign [20] with a fixed shape of $d \times h \times w$, such as $256 \times 7 \times 7$ for SGG tasks. Although preserving entities' spatial configuration, box-based features may lose semantic details due to the pooling operation. Furthermore, the fact that features are pooled into the same shape regardless of actual sizes of entities may result in the loss of semantic details in relationship inference. Another drawback of

the box-based representation is that it is computationally expensive to compute $\mathcal{O}(n^2)$ relationships for n entity proposals. Sampling candidate entities and relationships is commonly used during training. On the other hand, the point-based methods significantly reduce the computational cost by using features of shape $d \times 1$. By reformulating the SGG in a per-pixel regression fashion, point-based methods [7, 9] achieve much faster inference speed. However, the performance is relatively lower due to the coarse entity representations and handcrafted relationship targets. The query-based entity representation provides a way to perform object detection and SGG in an end-to-end manner. It exhibits greater capability in capturing semantics compared to convolutional regression, achieving better performance than point-based methods. Nevertheless, challenges arise for query-based methods due to factors like feature cardinality, the constraint of a fixed number of learnable queries, and increased complexity in both model design and post-processing. It is also difficult for query-based methods to perform predicate classification and scene graph classification due to the end-to-end prediction manner.

This work proposes a novel entity representation by using a set of semantically representative embeddings, which are augmented by visually representative points (rep-points) [193, 161] progressively. Different sets of rep-points are sampled dynamically w.r.t. entity reference points (centers) to update subject and object embeddings, respectively. Cross-attention between subject and object embeddings are also performed to achieve message passing. This approach allows for each entity instance to be represented as distinct queries and keys, which is more flexible than box-based representation and more fine-grained than point-based and query-based representations. Additionally, the proposed method eliminates the need for composite or predicate queries [190, 183, 181], as the predicate of a relationship can be computed as the multi-head attention weights between the

subjects (as queries) and objects (as keys). Unlike triplet classification, modeling relationships as attention weights preserves the directional information among subjects and objects, and captures more semantics.

5.2.2 Long-tailed Distributions

Long-tailed data distribution has been a key challenge in visual recognition [194], and it has been addressed in the recent literature on SGG [195]. In order to tackle this problem, various approaches have been proposed, such as data re-sampling [196, 197, 198, 199], de-biasing [200, 147, 201, 202, 203], and loss modification [204, 205, 173, 206, 207, 208, 209]. De-biasing methods require pre-trained biased models for initialization and then finetune the model. Loss modification methods generally assign a weight vector to the cross-entropy loss for predicate classification, with higher weights to tail classes and lower weights to head classes.

In this work, instead of re-weighting the loss function, another type of approach is applied by directly modifying the classification logits [187, 210, 211, 212]. A run-time performance-guided logit adjustment strategy is presented which offers a dynamic and effective control over the relative contributions of labels in the loss.

5.3 Contributions of this Chapter

1. We introduce RepSGG, a novel SGG paradigm in which entities are expressed as queries and keys, and relationships are represented as their attention weights. Significantly different from most existing SGG approaches, it explores a natural approach of capturing visual and semantic features progressively, and encapsulating relationships as attention weights which

encode the edge confidence and directionality effectively.

2. We propose a run-time performance-guided logit adjustment (PGLA) strategy to mitigate the long-tailed problem. PGLA is a simple, yet effective, model-agnostic, and cost-free solution that achieves a more balanced performance on unbalanced data. The choice of loss (*e.g.*, binary cross-entropy or cross-entropy) and performance metric (*e.g.*, recall or precision) are task-dependent, and this adaptability can extend to a range of settings and tasks.

3. We perform extensive experiments on the Visual Genome and Open Images V6 datasets to demonstrate the effectiveness of the proposed approach. Beyond standard SGG metrics, we also report the zero-shot mean recall (zs-mR) for our method and several state-of-the-art methods. RepSGG exhibits superior robustness and generalization capabilities on out-of-distribution data.

5.4 Technical Approach

In this section, we first provide the preliminaries of modeling entity detection in a per-pixel prediction fashion. We then introduce the RepSGG architecture, consisting of an entity detector, an entity encoder, a relationship encoder, and a relationship output layer. An illustration of RepSGG is shown in Fig. 5.2. Finally, we present several training strategies, including PGLA, addressing the challenges posed by long-tailed distributions and sparsely annotated data.

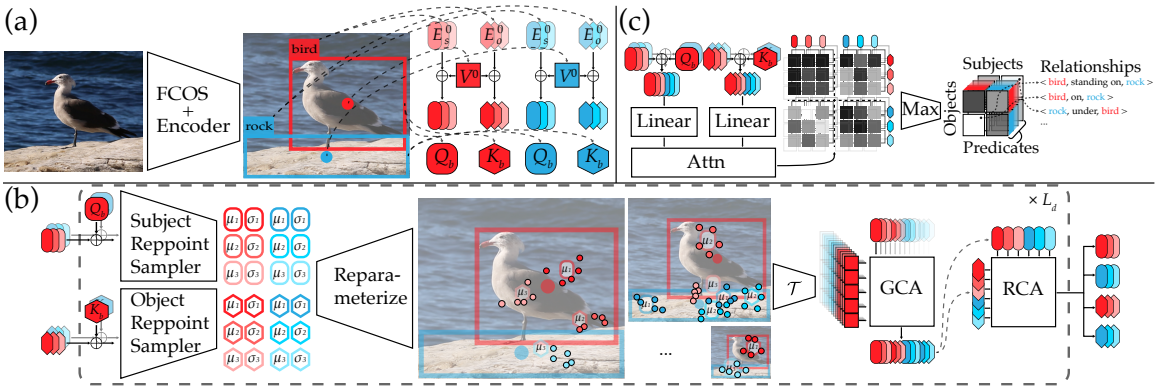


Figure 5.2: An illustration of RepSGG. (a) Entity detection and encoder: firstly, the FCOS entity detector detects a `bird` (colored in red) and `rock` (colored in blue). For each entity, K subject and object rep-embeddings ($K = 3$ in this illustration) are retrieved based on the entity's class label as $\mathbf{E}_s^0$ (shaped as a rounded rectangle) and $\mathbf{E}_o^0$ (shaped as a hexagon), respectively. The entity encoder generates entities' visual features as $\mathbf{V}^0$, and semantic-specific bounding box embeddings $\mathbf{Q}_b$ and $\mathbf{K}_b$. (b) Relationship encoder: the initial *queries* $\mathbf{Q}^0$ (representing subjects) are generated by adding $\mathbf{V}^0$ and $\mathbf{E}_s^0$ with $\mathbf{Q}_b$ acting as positional embeddings, and likewise for the *keys* $\mathbf{K}^0$ (representing objects). Semantic-specific visual features are sampled around entities dynamically based on input queries and keys via rep-point samplers (5 samples per rep-embedding in this illustration), and then utilized to update queries and keys via a GCA layer to gather more visual context. Subsequently, the cross-attention between queries and keys are performed via a RCA layer to further capture semantic features. (c) Relationship output layer: the pair-wise relationship scores are computed as the sigmoid activation of raw attention weights between the linear projections of queries and keys. The group-wise maximum scores are then taken as the predicate classification scores.

5.4.1 Entity Detection

Our model is built upon a one-stage anchor-free detector, namely FCOS [180]. Different from commonly used anchor-based R-CNN approaches for generating object proposals and features, entity detections are decoded from regressed dense features. Specifically, an input image $\mathbf{I} \in \mathbb{R}^{H^0 \times W^0 \times 3}$ will go through a backbone CNN (*e.g.*, ResNet-50 [2]) followed by a feature pyramid network (FPN) [3] to generate 5 scale levels of visual feature maps. Feature maps of different levels have different down-sampled spatial size w.r.t. the original image size, and thus are used for detecting entities of different sizes. To generate entity outputs, 3 fully convolutional detection heads shared between 5 scale levels are used, generating dense feature maps that provide entity classifi-

cation, bounding box regression, and center-ness scores, respectively. At each spatial location on feature maps of each level, FCOS directly predicts the entity category and the relative offsets from the four sides of the bounding box to the location. Locations with a final score weighted by the classification and center-ness scores over 0.2, followed by a non-maximum suppression (NMS) operation, are considered as positive detections. The entity detections are gathered as $\mathcal{B} : \{b^i\}_{i=1}^n$, where $b^i = (x_0^i, y_0^i, x_1^i, y_1^i, z^i, c^i)$, (x_0^i, y_0^i) and (x_1^i, y_1^i) denote the coordinates of the top-left and bottom-right corners of the bounding box, $z^i \in \{0, 1, 2, 3, 4\}$ is the scale level at which the detection is decoded, $c^i \in \{0, ..., C-1\}$ is the predicted entity label for a dataset containing C classes, and n is the number of detected entities. For SGG tasks, the top 100 detections are kept. Since the positive training targets are defined around the center of bounding boxes, the center is considered as the reference point of an entity in this paper. We maintain the FCOS architecture and training/testing settings unchanged, focusing instead on grounding entity features in a compact form to enable efficient relationship inference.

5.4.2 Entity Encoder

To encourage information exchange among different spatial locations and scales, a deformable transformer encoder [186] is used to further encode entity features. It consists of L_e deformable transformer encoder layers, which transform the backbone FPN features without changing their shapes. At each location, the deformable attention is performed by querying the FPN features at that location to features dynamically sampled around the location across all 5 scale levels. Such mechanism allows efficient multi-level feature aggregation, which is important for relationship inference as it requires more spatial context around entities. The output features of the entity encoder are considered as the visual features of the image. Following [186], the 2D positional embed-

dings [185] are generated and added with a learnable scale-level embedding. Note that the visual features and positional embeddings are multi-scale features. To merge levels of features, they are resized via bilinear interpolation to the shape of the largest feature map respectively. The interpolated features are then stacked along the level dimension, producing the visual features $\mathbf{V} \in \mathbb{R}^{5\times H\times W\times d}$ and positional embeddings $\mathbf{PE} \in \mathbb{R}^{5\times H\times W\times d}$, where $H = \lfloor H^0/8 \rfloor$, $W = \lfloor W^0/8 \rfloor$ and $d = 256$.

Entities can have different characteristics and roles depending on the context. For instance, detecting the relationship <man, on, street> requires the visual context around the feet of the man to determine the predicate on. While for <man, holding, apple>, it relies on the visual context surrounding the hands of the man. To account for such contextual variations, we initiate the representation of an entity with a distinct set of representative embeddings (rep-embeddings) in the semantic space, which we subsequently employ for relationship inference. Concretely, we construct subject rep-embeddings $\mathbf{E}_s$ and object rep-embeddings $\mathbf{E}_o$ of shape $C \times K \times d$, where K is the number of entity's class-specific embeddings. $\mathbf{E}_s$ and $\mathbf{E}_o$ are fixed-size learnable parameters initialized randomly, and are learnt through training. For every entity class, there are K distinct subject embeddings and K distinct object embeddings, respectively. These class-specific and semantic-specific embeddings serve as feature prototypes to characterize an entity from a particular class being a subject or object. By having distinct embeddings for entities being subjects and objects, our method can better capture the nuanced relationships and contextual information present in complex scenes. We conduct an ablation study using identical rep-embeddings for subjects and objects ($\mathbf{E}_s = \mathbf{E}_o$) in Section 5.5.5, where we show having distinct rep-embeddings achieves better results compared with identical ones.

5.4.3 Relationship Encoder

In this section, we describe the process of constructing semantic-specific entity features based on rep-embeddings ($\mathbf{E}_s$ and $\mathbf{E}_o$), visual features $\mathbf{V}$, positional embeddings $\mathbf{PE}$, and entity detections $\mathcal{B}$. For each detected entity, the relationship encoder generates a subject and object feature representations, respectively. Then, it employs the attention mechanism [185] to aggregate local visual features and capture dependencies between all subjects and objects.

The relationship encoder is composed of a stack of L_d encoder layers, where the initial inputs are formed by fusing visual features and bounding box embeddings with subject rep-embeddings and object rep-embeddings, respectively. The outputs are encoded subject and object features of the same shape as inputs. Each layer has a rep-point sampler, two group cross-attention layers, and a two-way relational cross-attention layer. To simplify the terms and make them compatible with the attention mechanism, we refer to the subject embeddings as *queries*, and object embeddings as *keys* for the rest of the paper.

5.4.3.1 Initial Queries and Keys

Rep-embeddings only provide identities of entity classes and semantics, without considering visual and spatial contexts. To integrate such information, the entity-specific visual and spatial features are sampled from the entity reference points, and fused with rep-embeddings to create subject and object features as the initial queries and keys to the relationship encoder.

Let the reference point $\boldsymbol{p} \in [0, 1]^3$ be the normalized coordinates, where $(0, 0, 0)$ and $(1, 1, 1)$ indicate the top-left corner at the lowest scale level, and bottom-right corner at the highest scale level of the features, then for $n \times m$ reference points $\mathbf{P} = \{\boldsymbol{p}^{i,1}, \ldots, \boldsymbol{p}^{i,m}\}_{i=1}^{n}$, the point

sampler function is defined as

$$\mathcal{T}(\,\cdot\,,\mathbf{P}) : \mathbb{R}^{5\times h\times w\times d} \times \mathbb{R}^{n\times m\times 3} \to \mathbb{R}^{n\times m\times d}, \tag{5.1}$$

which is achieved by bilinear interpolation. To prepare semantic-agnostic entity features, the normalized centers of bounding boxes $\mathbf{P}^0 \in \mathbb{R}^{n\times 1\times 3}$ are used as the reference points derived from the entity detections $\mathcal{B}$. The point sampler is applied on $\mathbf{V}$ and $\mathbf{PE}$ to get the corresponding features at the entities' reference points as:

$$\begin{aligned} \mathbf{V}^0 &= \mathcal{T}(\mathbf{V}, \mathbf{P}^0) \\ \mathbf{PE}^0 &= \mathcal{T}(\mathbf{PE}, \mathbf{P}^0) \\ \mathbf{P}^0 &= \left\{ \left(\frac{x_0^i + x_1^i}{2W}, \frac{y_0^i + y_1^i}{2H}, \frac{z^i}{4} \right) \right\}_{i=1}^{n}. \end{aligned} \tag{5.2}$$

For assigning semantics to entities, the subject and object embeddings are gathered from the corresponding indices of predicted entity labels $\{c^i\}_{i=1}^n$ as $\mathbf{E}_s^0 \in \mathbb{R}^{n\times K\times d}$ and $\mathbf{E}_o^0 \in \mathbb{R}^{n\times K\times d}$. The subject and object embeddings are class-dependent learnable parameters which capture the semantics of an entity class being the subject and object. Subject queries $\mathbf{Q}^0$ and object keys $\mathbf{K}^0$ are then constructed by adding the corresponding embeddings with the visual features $\mathbf{V}^0$:

$$\begin{aligned} \mathbf{Q}^0 &= \mathbf{E}_s^0 + \mathbf{V}^0 \\ \mathbf{K}^0 &= \mathbf{E}_o^0 + \mathbf{V}^0. \end{aligned} \tag{5.3}$$

By merging instance-specific visual features with class-specific embeddings, the queries (or keys) retain semantic similarities within their entity classes while also diversifying the instance-wise entity representations.

The bounding box also plays a crucial role in determining the spatial relationship between entities. Hence, the bounding box coordinates are mapped into embeddings. First, the positional

embeddings $\mathbf{PE}$ of the top-left and bottom-right corners are sampled via (5.1). Two learnable embeddings indicating "top-left corner" and "bottom-right corner" are added with the corresponding corners' positional embeddings, respectively. The two corner embeddings are then concatenated and fed to a fully-connected layer, resulting in the box embeddings. Lastly, two learnable embeddings are added with the box embeddings to construct subject and object box embeddings respectively, denoted as $\mathbf{Q}_b \in \mathbb{R}^{n \times d}$ and $\mathbf{K}_b \in \mathbb{R}^{n \times d}$.

5.4.3.2 Rep-point Sampler

For rep-embeddings to capture more visual context, we propose a dynamic approach for sampling features from representative points (rep-points) and message passing via attentions. Two rep-point samplers are implemented to sample subject and object rep-points, since queries and keys serve distinct roles in conveying subject and object semantics respectively. Specifically, a multi-layer perceptron (MLP) is used as the sampler to predict the points of interest for each entity. To achieve semantic-specific sampling, the MLP weights are split into K groups (in practice, group 1D convolution is used), and k-th group is applied on the k-th slice of the inputs (queries or keys) along the K dimension of rep-embeddings. To further increase the diversity of sampling and prevent overfitting on few points, instead of directly predicting the coordinates, the distribution parameters of offsets w.r.t. the reference points are predicted, following the variational autoencoder (VAE) and reparameterization trick [213]. Let the input queries to the l-th layer be $\mathbf{Q}^{l-1}$, the subject rep-point offsets are defined as

$$\begin{aligned} \Delta \mathbf{P}_s^l &= \boldsymbol{\mu}_s^l + \boldsymbol{\sigma}_s^l \odot \boldsymbol{\epsilon}, \\ \boldsymbol{\mu}_s^l, \boldsymbol{\sigma}_s^l &= \text{MLP}(\mathbf{Q}^{l-1} + \mathbf{Q}_b) \end{aligned} \tag{5.4}$$

where means $\boldsymbol{\mu}_s^l \in \mathbb{R}^{n\times K\times 3}$ and standard deviations (stds) $\boldsymbol{\sigma}_s^l \in \mathbb{R}^{n\times K\times 3}$ are the outputs of the subject rep-point sampler, $\odot$ is the element-wise product, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_3)$. To estimate robust parameters, m points are randomly sampled per parameter during training, namely $\boldsymbol{\epsilon} \in \mathbb{R}^{n\times K\times m\times 3}$. Similarly, the object rep-point offsets are obtained as $\Delta\mathbf{P}_o^l$. By adding the sampled rep-points offsets to the reference points, the sampled points are obtained as

$$\begin{aligned} \mathbf{P}_s^l &= \mathbf{P}_s^{l-1} + \Delta\mathbf{P}_s^l,\ l = 1\dots L \\ \mathbf{P}_o^l &= \mathbf{P}_o^{l-1} + \Delta\mathbf{P}_o^l,\ l = 1\dots L, \end{aligned} \tag{5.5}$$

where $\mathbf{P}_s^0 = \mathbf{P}_o^0 = \mathbf{P}^0$ with expanded shapes of $n \times 1 \times 1 \times 3$. The offsets are accumulated through encoder layers from the original entity centers $\mathbf{P}^0$, so relevant features can be aggregated as the encoder layer goes deeper. Accordingly, rep-point visual features and positional embeddings are sampled via (5.1):

$$\begin{aligned} \mathbf{V}_s^l &= \mathcal{T}(\mathbf{V}, \mathbf{P}_s^l),\ \mathbf{PE}_s^l = \mathcal{T}(\mathbf{PE}, \mathbf{P}_s^l), \\ \mathbf{V}_o^l &= \mathcal{T}(\mathbf{V}, \mathbf{P}_o^l),\ \mathbf{PE}_o^l = \mathcal{T}(\mathbf{PE}, \mathbf{P}_o^l). \end{aligned} \tag{5.6}$$

The rep-point sampler provides a probabilistic mapping from visual cues to the semantic space, which helps finding visually relevant features with semantic significance. Since our task is prediction rather than generation, to achieve deterministic SGG evaluation results, the stochastic behavior of rep-point samplers is transformed into a deterministic one by sampling within a range with a fixed step size during inference. In particular, the random noise vector $\boldsymbol{\epsilon}$ for each offset is replaced by a set $\{-\xi, -\xi+1, ..., 0, ..., \xi\}^3$, where ξ is the range of consideration which is set to 3 by default. In other words, rep-points are sampled within the "3σ" range with a step size of "σ" along the width, height, and scale dimensions in a combinatorial manner. For a subject rep-point sampler for the k-th rep-embedding, at layer l, the rep-points are derived the Cartesian product $\prod_{dim=1}^{3}(\boldsymbol{\mu}_s^{l,k,dim} - 3\boldsymbol{\sigma}_s^{l,k,dim}, \cdots, \boldsymbol{\mu}_s^{l,k,dim} + 3\boldsymbol{\sigma}_s^{l,k,dim})$. In total, there are $7^3 = 343$ rep-

points sampled per mean. Comparisons of performance and inference speed w.r.t. ξ are provided in Section 5.5.

5.4.3.3 Group Cross-Attention

The group cross-attention (GCA) captures the visual features that correspond to each subject and object rep-embedding by computing their attention scores, respectively. The application of GCA involves performing separate interactions between queries and subject rep-point features, as well as between keys and object rep-point features. The subject GCA for the i-th entity is defined as

$$\begin{aligned}
\mathbf{Q}^{l,i} &= \text{GCA}(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) \\
&= \text{softmax}(\boldsymbol{q}\boldsymbol{k}^T/\sqrt{d_G})\ \boldsymbol{v} \\
\boldsymbol{q} &= \text{Linear}(\mathbf{Q}^{l-1,i} + \mathbf{Q}_b^i) \in \mathbb{R}^{h_G \times K \times d_G} \\
\boldsymbol{k} &= \text{Linear}(\mathbf{V}_s^{l,i} + \mathbf{PE}_s^{l,i}) \in \mathbb{R}^{h_G \times K \times m \times d_G} \\
\boldsymbol{v} &= \text{Linear}(\mathbf{V}_s^{l,i}) \in \mathbb{R}^{h_G \times K \times m \times d_G},
\end{aligned} \tag{5.7}$$

where i indexes the entity, Linear$(\cdot)$ is a fully-connected layer ($\boldsymbol{q}$, $\boldsymbol{k}$, and $\boldsymbol{v}$ are projected with different parameters), h_G is the number of attention heads, and d_G is the dimension of each head. GCA is performed independently among groups in parallel, where the cross-attention between a rep-embedding and its corresponding sampled rep-point features are performed. Following the transformer architecture [185], multi-head outputs are concatenated and projected with a fully-connected layer, and a residual connection [2] with layer normalization [214] is added. Likewise, another GCA layer with different parameters is performed for keys as $\mathbf{K}^l = \text{GCA}(\text{Linear}(\mathbf{K}^{l-1} + \mathbf{K}_b)$, $\text{Linear}(\mathbf{V}_o^l + \mathbf{PE}_o^l), \text{Linear}(\mathbf{V}_o^l))$. GCA allows each rep-embedding to focus on different visual features, carrying the relevant ones along the way.

5.4.3.4 Two-Way Relational Cross-Attention

In GCA, queries and keys are updated by their corresponding sampled features. In a two-way relational cross-attention (RCA) layer, queries are updated by keys, and vice versa. Firstly, the raw attention weights $\mathbf{A}^l$ are computed between flattened $\mathbf{Q}^l$ and $\mathbf{K}^l$. Since the projections of queries and keys are different, the attention weights are not symmetric and can be normalized along different dimensions. Softmax is then applied on $\mathbf{A}^l$ along the dimension of keys, and along the dimension of queries to obtain two-way attention weights. Finally, queries and keys are updated by multiplying the corresponding attention weights with values. The two-way relational cross-attention is formally defined as following:

$$
\begin{aligned}
\mathbf{Q}^l, \mathbf{K}^l &= \text{RCA}(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}_q, \boldsymbol{v}_k) \\
\boldsymbol{q} &= \text{Linear}(\mathbf{Q}^l + \mathbf{Q}_b) \\
\boldsymbol{k} &= \text{Linear}(\mathbf{K}^l + \mathbf{K}_b) \\
\boldsymbol{v}_q &= \text{Linear}(\mathbf{Q}^l) \\
\boldsymbol{v}_k &= \text{Linear}(\mathbf{K}^l) \\
\mathbf{A}^l &= \boldsymbol{q}\boldsymbol{k}^T / \sqrt{d_R} \in \mathbb{R}^{h_R \times n_q \times n_k} \\
\mathbf{A}_k^l &= \text{softmax}(\mathbf{A}^l)\ s.t.\ \textstyle\sum_{j=1}^{n_k} \mathbf{A}_k^{l,*,*,j} = 1 \\
\mathbf{A}_q^l &= \text{softmax}(\mathbf{A}^l)\ s.t.\ \textstyle\sum_{i=1}^{n_q} \mathbf{A}_q^{l,*,i,*} = 1 \\
\mathbf{Q}^l &= \mathbf{A}_k^l \boldsymbol{v}_k,\ \mathbf{K}^l = (\mathbf{A}_q^l)^T \boldsymbol{v}_q,
\end{aligned}
\tag{5.8}
$$

where $n_q = n_k = n \times K$, h_R is the number of attention heads, and d_R is the dimension of each head, "$*$" denotes any index along the specific dimension. Additionally, two MLPs are used for projecting the output queries and keys respectively, following the feed-forward network design in

[185]. Without abuse of notation, the notations of output queries $\mathbf{Q}^l$ and keys $\mathbf{K}^l$ of RCA layers remain the same. After L_d relationship encoder layers, the outputs $\mathbf{Q}^{L_d}$ and $\mathbf{K}^{L_d}$ are obtained which are used for predicate prediction.

5.4.4 Relationships as Attention Weights

For most SGG methods using either box-based, or query-based representation, predicate classification is performed on triplet features in different forms of feature fusions. For example, box-based methods use the concatenation of pairwise entity features and their union-box features, and query-based methods use learnable triplet embeddings to perform classification directly. Neither of them can capture the directional information of scene graphs explicitly which could cause learning bias and overfitting on dominant visual configurations. A very simple case is that there is a common triplet <man, on, street> in the dataset, the learnt model will likely predict on if it detects man and street concurrently. Conversely, triplets such as <man, standing on, street> and <street, under, man> are considered as incorrect predictions and are treated as negative examples during training, despite being semantically valid triplets. The presence of rare, bidirectional [215], and unannotated relationships [199] hinders the learning of representative semantics for SGG methods.

Similar to the RCA discussed in Section 5.4.3.4, $\mathbf{Q}^{L_d}$ and $\mathbf{K}^{L_d}$ are projected with h_A heads, and each head has d_A dimensions. Unnormalized attention weights $\mathbf{A}^{L_d} \in \mathbb{R}^{h_A \times n_q \times n_k}$ are computed between projected queries and keys. Different from RCA, the multi-head attention weights are not multiplied by projected values. The asymmetric nature of dot-product attention serves as a natural metric for quantifying the relationship between subjects and objects. Moreover, attention weights of each head captures distinct semantics, similar to the way feature channels oper-

ate. Therefore, the attention weights can be mapped to the predicate classification $\mathbf{Y} \in \mathbb{R}^{P \times n_q \times n_k}$ via a fully-connected layer, where P is the number of predicate classes of a dataset. In parallel, a binary relation mask $\mathbf{H} \in \mathbb{R}^{n_q \times n_k}$ is also predicted to classify if a relationship exists between a pair of rep-embeddings. The relation mask is used for suppressing low-quality predictions. Formally, $\mathbf{Y}$ and $\mathbf{H}$ are obtained as

$$\begin{aligned} \mathbf{Y} &= \text{Linear}(\mathbf{A}^{L_d}) \\ \mathbf{H} &= \text{Linear}(\mathbf{A}^{L_d}) \\ \mathbf{A}^{L_d} &= \boldsymbol{q}\boldsymbol{k}^T/\sqrt{d_A} + \boldsymbol{b}_A \\ \boldsymbol{q} &= \text{Linear}(\mathbf{Q}^L + \mathbf{Q}_b) \\ \boldsymbol{k} &= \text{Linear}(\mathbf{K}^L + \mathbf{K}_b), \end{aligned} \tag{5.9}$$

where $\boldsymbol{b}_A$ is an added bias term. $\mathbf{Y}$ represents relationships between subject and object rep-embeddings instead of subjects and objects. To get pairwise relationships between entities, $\mathbf{Y}$ is re-arranged to $\mathbb{R}^{P \times n \times n \times K^2}$. During training, Gumbel-Softmax [216] is applied over the last dimension of $\mathbf{Y}$ to sample the rep-embedding pairs with the largest logits, and $\mathbf{Y}$ is reduced to the shape of $\mathbb{R}^{P \times n \times n}$. The attention matrices $\mathbf{Y}$ now represent the raw edge (relationship) scores of a fully-connected scene graph between all n entities over P predicates. An annealing schedule is applied that changes the Gumbel-Softmax temperature from 10 to 0.5 gradually through the first 30% iterations. During inference, the maximum logits over the last dimension are chosen. The relation mask $\mathbf{H}$ undergoes the same re-arrangement operation, followed by selecting the maximum value for both training and inference. The final predicate classification score is defined as $(\sigma(\mathbf{H}) \cdot \sigma(\mathbf{Y}))^\zeta$, where $\sigma(\cdot)$ is the sigmoid function, and ζ is a hyper-parameter to balance between predicate scores and entity detection scores. During evaluation, the triplet score is computed by multiplying the predicate classification

score, subject detection score, and object detection score. We empirically set $\zeta = 2$ for evaluating on the Visual Genome [1] dataset where recall is the primary metric. For the Open Images [217] dataset where precision is the main metric, we find that assigning more weights to entity detection scores yields improved results, effectively lowering the ranking of false positive detections. We set $\zeta = 0.1$ for evaluation on the Open Images dataset.

5.4.5 Training

In this section, we discuss the training losses and strategies to address the challenges posed by long-tailed sparsely-annotated data. The hyper-parameters and losses for entity detection remain the same as in FCOS [180].

5.4.5.1 Losses

Due to the potential existence of multiple relationships (directional or bidirectional) between two entities, the scene graph frequently exhibits a multi-graph structure. The predicate classification is considered as a multi-label multi-class classification problem. Consequently, we use the binary cross-entropy (BCE) instead of softmax to supervise the predicate classification and relation mask. To balance well-learned and hard examples, the focal loss (FL) [104] for BCE is used. Specifically, given the predicted logits $\hat{\mathbf{Y}}$ and ground-truth labels $\mathbf{Y} \in \{0,1\}^{P \times n \times n}$, the focal BCE is defined as

$$\text{FL}(\hat{\mathbf{Y}}, \mathbf{Y}) = \begin{cases} -\frac{\alpha}{N_{pos}} \sum\limits_{p,i,j} (1 - \sigma(\hat{\mathbf{Y}}^{p,i,j}))^{\gamma} \log(\sigma(\hat{\mathbf{Y}}^{p,i,j})), & \mathbf{Y}^{p,i,j} = 1 \\ -\frac{1-\alpha}{N_{pos}} \sum\limits_{p,i,j} \sigma(\hat{\mathbf{Y}}^{p,i,j})^{\gamma} \log(1 - \sigma(\hat{\mathbf{Y}}^{p,i,j})), & \mathbf{Y}^{p,i,j} = 0, \end{cases} \tag{5.10}$$

where α is a class-balance weighting factor, γ is the focal factor, p indexes the predicate classes, i indexes subjects, j indexes objects, and $N_{pos} = \sum_{p,i,j} \mathbf{Y}^{p,i,j}$ is the number of ground-truth triplets. As the predicate prediction forms a fully-connected graph among n entities, the ground-truth $\mathbf{Y}$ is inherently sparse with few ones. We select $\alpha = 0.75$ to prioritize generating larger logits for positive predicates, rather than penalizing negative predicates. For the focal factor γ, we set it differently based on the predicate frequency of training data. Let the predicate priors be $\boldsymbol{\eta}$, *e.g.*, the empirical predicate class frequencies in the training dataset, we compute a predicate-specific $\boldsymbol{\gamma}$ to replace γ in (5.10) as

$$\boldsymbol{\gamma} = \gamma \cdot \frac{\boldsymbol{\eta} - \min(\boldsymbol{\eta})}{\max(\boldsymbol{\eta}) - \min(\boldsymbol{\eta})}, \tag{5.11}$$

where $\gamma = 2$, $\min(\cdot)$ and $\max(\cdot)$ are operations to get the minimum and maximum value respectively. For the tail predicates, $\boldsymbol{\gamma}^p$ is small so that it encourages the logits to be larger. For the head predicates, $\boldsymbol{\gamma}^p$ becomes larger and down-weights the loss. For supervising the relation mask, we empirically select $\alpha = 0.75$ and $\gamma = 2$.

As discussed in [174, 7, 199], the sparsity of data annotations for relations, coupled with the presence of numerous unannotated ones, leads to semantic ambiguity and poses challenges during training. Therefore, simply considering triplets without ground-truth annotations as negative is not optimal. For training the relation mask $\hat{\mathbf{H}} \in \mathbb{R}^{n \times n}$ among n entities, we sub-sample the negative triplets with a ratio of 10:1 in proportion to the number of ground-truth triplets. Additional, we employ a margin ranking loss for predicted relation classification $\hat{\mathbf{Y}}$. Instead of supervising negative samples (where $\mathbf{Y}^{p,i,j} = 0, \forall p$) with labels of zero in BCE, per-predicate margins are calculated and used as the upper bounds for negative samples' logits. The margin ranking loss $\mathcal{L}_\eta$

is defined as

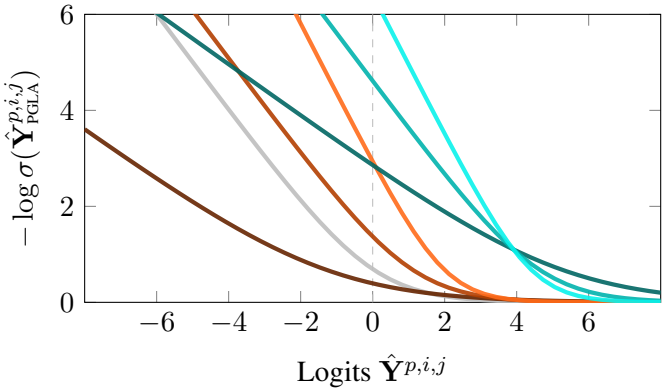
$$\mathcal{L}_\eta(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{N_{neg}} \sum_{p,i,j} \max(\sigma(\hat{\mathbf{Y}}^{p,i,j}) - \eta^p),\ 0),$$
$$\eta^p = \min\{\sigma(\hat{\mathbf{Y}}^{p,i,j}) \mid \mathbf{Y}^{p,i,j} = 1,\ \forall\ i, j\}, \tag{5.12}$$

where N_{neg} is the number of negative samples, and η^p is the margin for predicate p between positive and negative samples. To ensure a normalized effect across different scenes (images), the margins are computed on a per-image basis. By employing this loss, unannotated triplets are neither excessively penalized, which could lead to training ambiguity, nor overly encouraged, which could result in lower ranks for positive triplets.

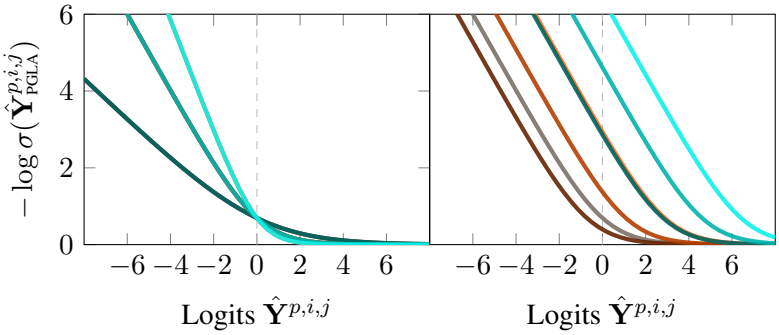
For sampling useful subject and object rep-points, margin ranking losses are applied to the subject and object offset means $\boldsymbol{\mu}_s^l$ and $\boldsymbol{\mu}_o^l$, for $l = 1, ..., L$. For the rep-point coordinates reparameterized by the mean, their margins are the top-left and bottom-right corner coordinates of union bounding boxes of triplets involving the entity. Consequently, the loss is nonzero when a mean rep-point is outside of a corresponding union bounding box.

5.4.5.2 Performance-Guided Logit Adjustment

Building upon the concept of the logit adjustment (LA) [187], we introduce the run-time performance-guided logit adjustment (PGLA) as a novel approach to enhance the performance on long-tail problems. In logit adjusted softmax cross-entropy [187], the logits from a classifier are added with a bias term $\log \boldsymbol{\eta}$ to create pairwise margins between classes. Due to the intricacies involved in entity detection and scene graph generation, employing a fixed bias throughout the training process is suboptimal. Hence, we extend the logit adjustment to a more general form of affine transformation by adding a weight term, and leveraging the training statistics to more



(a) For positive targets, PGLA assigns more loss on tail predicates over head predicates in general. It also assigns more loss on hard predicates (*e.g.*, low recall observed during training), and less on well-classified predicates (*e.g.*, high recall regardless of being a head or tail predicate).

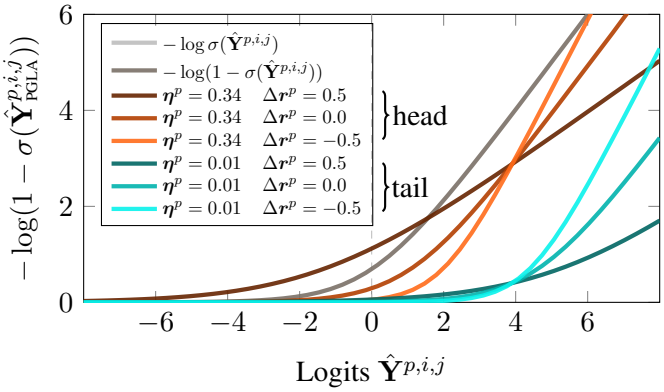


(b) Left: individual loss effects of $\mathbf{W}$, where a positive predicate with a relatively high recall receives less loss when misclassified (logits less than 0), but more loss to suppress over-confident predictions. Right: individual loss effects of $\mathbf{B}$, where predicates of lower frequencies or performance receive more loss.

accurately quantify the class margins.

In the context of the scene graph generation task, recall is used to guide the strength of adjustment, and PGLA is only applied to logits of positive relations. For the pair of subject i and object j, the predicted logits are adjusted as

$$\hat{\mathbf{Y}}^{:,i,j}_{\text{PGLA}} = \begin{cases} \mathbf{W} \odot \hat{\mathbf{Y}}^{:,i,j} + \mathbf{B} & (\exists\, p)\ \mathbf{Y}^{p,i,j} = 1 \\ \hat{\mathbf{Y}}^{:,i,j} & \text{otherwise,} \end{cases} \tag{5.13}$$



(c) BCE loss on PGLA-adjusted logits. (a) Loss on a positive predicate p where $\mathbf{Y}^{p,i,j} = 1$. (b) Loss on a positive predicate p with $\mathbf{B} = \mathbf{0}$ (left), and with $\mathbf{W} = \mathbf{1}$ (right). (c) Loss on a negative predicate p where $\mathbf{Y}^{p,i,j} = 0$. The legend in (c) is shared across (a) - (c), and gray lines in figures represent the BCE loss on original logits.

Figure 5.3: BCE loss on PGLA-adjusted logits. (a) Loss on a positive predicate p where $\mathbf{Y}^{p,i,j} = 1$. (b) Loss on a positive predicate p with $\mathbf{B} = \mathbf{0}$ (left), and with $\mathbf{W} = \mathbf{1}$ (right). (c) Loss on a negative predicate p where $\mathbf{Y}^{p,i,j} = 0$. The legend in (c) is shared across (a) - (c), and gray lines in figures represent the BCE loss on original logits.

where $\mathbf{W} \in \mathbb{R}^P$ and $\mathbf{B} \in \mathbb{R}^P$ are the weight and bias factors, respectively, and the operation denoted by ":" selects all elements along the specified dimension. By setting $\mathbf{W} = \mathbf{1}$ and $\mathbf{B} = \log \boldsymbol{\eta}$, (5.13) yields logit adjustment [187]. The weight and bias factors for (5.13) are calculated as

$$\begin{aligned} \mathbf{W} &= -\tanh(\Delta\boldsymbol{r}) + 1, \\ \mathbf{B} &= -\tanh(\Delta\boldsymbol{r}/\lambda) \cdot \log(P^{-1}) + \log \boldsymbol{\eta}, \end{aligned} \tag{5.14}$$

where $\Delta\boldsymbol{r} = \boldsymbol{r} - \bar{\boldsymbol{r}}$, $\boldsymbol{r}$ is the measured recall, $\bar{\boldsymbol{r}}$ is the mean of the recall, λ is a hyper-parameter set to 1 by default, and $\log(P^{-1})$ is the log probability of the uniform distribution over P predicates. The hyper-parameter λ controls the sensitivity of PGLA w.r.t. the recall differences. A smaller value of λ increases the sensitivity, and more losses are enforced for predicates with low recall. The $\tanh(\cdot)$ function limits $\Delta\boldsymbol{r}$ or $\Delta\boldsymbol{r}/\lambda$ within the range of [-1, 1], and there exist alternative functions that serve the same purpose.

The effect of the long-tailed problem on losses can be described as follows: tail predicates play the role of negative classes when training head predicates, and constantly receive losses for being classified as negatives during training; on the other hand, head predicates receive more losses for being positive and less losses for being classified as negatives during training. To address this effect, there are various considerations to be taken into account regarding $\mathbf{W}$ and $\mathbf{B}$. Overall, the impacts of $\mathbf{W}$ and $\mathbf{B}$ w.r.t. the BCE loss with adjusted logits $\hat{\mathbf{Y}}_{\text{PGLA}}$ are illustrated in Fig. 5.3. In the case where a predicate p achieves a relatively higher recall ($\Delta \boldsymbol{r}^p > 0$), we decrease the value of $\mathbf{W}^p$ to encourage the network to generate larger logits when p is positive ($\mathbf{Y}^{p,*,*} = 1$), and smaller logits when p is negative ($\mathbf{Y}^{p,*,*} = 0$). This adjustment of $\mathbf{W}$ tries to push the predictions towards the saturation regions of the sigmoid function so that it is easier to distinguish between positive and negative classes. Simultaneously, as the recall $\boldsymbol{r}^p$ increases relatively, $\mathbf{B}^p$ increases and less loss is assigned to predicate p, allowing us to focus on other predicates with lower recall. In addition to utilizing the recall, the bias factor per predicate will be adjusted based on its prior distribution as well. A tail predicate is assigned with a smaller bias factor, and receive larger losses when being positive and smaller loss when being negative.

Despite achieving the goal of assigning class margins dynamically to address the long-tailed problem, it is critical to note that the similarities between predicates have not been taken into account. Therefore, we introduce a training statistic named "confusion logits", denoted as $\mathbf{D} \in \mathbb{R}^{P \times P}$, which tracks pairwise predicate logit differences if the predictions are incorrect. The

confusion logit between the ground-truth predicate p and an arbitrary predicate $\hat{p}$ is computed as

$$\begin{aligned}
\mathbf{D}^{p,\hat{p}} &= \underset{(p,i,j)\in\Omega}{\text{mean}}\{\boldsymbol{d}^{i,j}_{logits} \cdot \boldsymbol{d}^{p,\hat{p}}_{\eta}\} \\
\Omega &\coloneqq \{(p,i,j) \mid \mathbf{Y}^{p,i,j} = 1\} \\
\boldsymbol{d}^{i,j}_{logits} &= \text{ReLU}(\hat{\mathbf{Y}}^{\hat{p},i,j} - \hat{\mathbf{Y}}^{p,i,j}) \\
\boldsymbol{d}^{p,\hat{p}}_{\eta} &= \tanh(\text{ReLU}(\log \boldsymbol{\eta}^{\hat{p}} - \log \boldsymbol{\eta}^{p})),
\end{aligned} \tag{5.15}$$

where $\text{ReLU}(\cdot)$ [218] is used for selecting positive values only. The confusion logits for a GT predicate p are computed only with respect to incorrectly-predicted predicates ($\hat{\mathbf{Y}}^{\hat{p},i,j} > \hat{\mathbf{Y}}^{p,i,j}$) with larger priors ($\boldsymbol{d}^{p,\hat{p}}_{\eta} > 0$ if $\boldsymbol{\eta}^{\hat{p}} > \boldsymbol{\eta}^{p}$). Large value of $\mathbf{D}^{p,\hat{p}}$ means that p is often mis-classified as $\hat{p}$. The confusion logits not only quantify the effects of long-tailed data, but also the semantic similarities between predicates. For similar tail predicates like `across` and `along`, their confusion logits can be large as well. During training, per-instance PGLA is applied instead by utilizing the corresponding confusion logits of a specific ground-truth predicate p, and (5.13) is modified as

$$\hat{\mathbf{Y}}^{:,i,j}_{\text{PGLA}} = \begin{cases} \mathbf{W} \odot \hat{\mathbf{Y}}^{:,i,j} + \mathbf{B} + \mathbf{D}^{p,:} & (\forall\, p)\ \mathbf{Y}^{p,i,j} = 1 \\ \hat{\mathbf{Y}}^{:,i,j} & \text{otherwise.} \end{cases} \tag{5.16}$$

Ideally, recall of each predicate should be close to the mean recall for a balanced classifier. However, head classes tend to exhibit significantly higher recall compared to tail classes due to long-tailed training data. Therefore, we evaluate the recall per predicate differently. The exponential moving average (EMA) of predicate recall per mini-batch is computed to estimate the performance change over time. For each image in the mini-batch, the histogram of ground-truth relationships per predicate is obtained as $\boldsymbol{n}$. Next, based on the top $\text{sum}(\boldsymbol{n})$ triplets and predicate priors, ranking targets $\boldsymbol{\kappa}$ are set differently per predicate. For a predicate p, the top-$\boldsymbol{\kappa}^p$ triplets is used for computing the recall, where tail predicates are assigned with smaller $\boldsymbol{\kappa}^p$ while head predicates are assigned with

larger values. As an illustration, the ranking target $\boldsymbol{\kappa}^{p_0}$ for the rarest predicate p_0 is $\boldsymbol{n}^{p_0}$, where the predicate should be ranked within the top-$\boldsymbol{n}^{p_0}$ in a mini-batch. For the second rarest predicate p_1, it should be ranked within the top-$(\boldsymbol{n}^{p_0}+\boldsymbol{n}^{p_1})$, and so forth. By setting distinct ranking targets for each predicate, it forces tail predicates to rank higher than head ones. Finally, the per-predicate recall $\boldsymbol{r}$ is calculated. A EMA momentum specific to each predicate is assigned as $\boldsymbol{\rho} = 0.9999^{-\log \boldsymbol{\eta}}$. Assume the batch size is 1, the process of calculating recall at iteration t is detailed in Algorithm 3. The confusion logits $\mathbf{D}_t$ is calculated and updated per iteration via EMA as well. Consequently, the PGLA can be performed given the run-time values of $\mathbf{W}_t$, $\mathbf{B}_t$, and $\mathbf{D}_t$ in (5.16).

Algorithm 3 Recall Calculation at training iteration t

Input: $\hat{\mathbf{Y}}, \mathbf{Y}, \boldsymbol{r}_{t-1}, \boldsymbol{\eta}$

Output: $\boldsymbol{r}_t$

$\boldsymbol{r}_t \leftarrow \mathbf{0}_P$

$\boldsymbol{\nu} \leftarrow \arg\mathrm{sort}(\boldsymbol{\eta})$ ▷ indices of sorted predicate

$\boldsymbol{n} \leftarrow \sum_{i,j} \mathbf{Y}^{:,i,j}$ ▷ No. of GTs per predicate

$\boldsymbol{\kappa} \leftarrow \mathrm{cumsum}(\boldsymbol{n}[\boldsymbol{\nu}])$ ▷ cumulative sum of No. of GTs

for $p \leftarrow \boldsymbol{\nu}^1$ to $\boldsymbol{\nu}^P$ **do**

  $\hat{R}_{\mathrm{top}-\boldsymbol{\kappa}^p} \leftarrow \arg\max_{\boldsymbol{\kappa}^p}(\hat{\mathbf{Y}})$ ▷ top $\boldsymbol{\kappa}^p$ triplets

  $R^p \leftarrow \mathrm{nonzero}(\mathbf{Y}^p)$ ▷ GT triplets

  $\boldsymbol{r}_t^p \leftarrow \boldsymbol{r}_t^p + \mathrm{match}(\hat{R}_{\mathrm{top}-\boldsymbol{\kappa}^p}, R^p)$ ▷ accumulate matches

end for

$\boldsymbol{r}_t \leftarrow \boldsymbol{r}_t \oslash \boldsymbol{n}$ ▷ element-wise division

$\boldsymbol{r}_t \leftarrow (1-\boldsymbol{\rho}) \cdot \boldsymbol{r}_t + \boldsymbol{\rho} \cdot \boldsymbol{r}_{t-1}$ ▷ EMA

return $\boldsymbol{r}_t$

5.5 Experiments

5.5.1 Datasets and Evaluation

Datasets. We evaluate our methods on the Visual Genome (VG) [1] dataset. We use the widely-used pre-processed subset VG150 introduced by [138] for evaluation, which contains the most frequent 150 entities (C = 150) and 50 predicates (P = 50). The VG150 dataset contains

Table 5.1: Comparisons of R@K and mR@K results on VG150 between the proposed methods and SOTA methods. Methods are grouped from top to bottom as: point-based, query-based, and box-based methods. FCSGG [7] uses HRNet [8] as backbone, and CoRF [9] uses Swin-S [10]. The best results are bold, and the second-best results are underlined.

	Predicate Classification						Scene Graph Classification						Scene Graph Detection					
	R@20/50/100			mR@20/50/100			R@20/50/100			mR@20/50/100			R@20/50/100			mR@20/50/100		
FCSGG [7]	33.4	41.0	45.0	4.9	6.3	7.1	19.0	23.5	25.7	2.9	3.7	4.1	16.1	21.3	25.1	2.7	3.6	4.2
CoRF [9]	-	45.4	-	-	10.1	-	-	18.7	-	-	3.9	-	-	18.6	-	-	3.9	-
RelTR [183] ‡	**63.1**	64.2	-	20.0	21.2	-	29.0	36.6	-	7.7	11.4	-	21.2	27.5	-	6.8	10.8	-
TraCQ [182] ‡	-	-	-	-	-	-	-	-	-	-	-	-	19.7	28.3	35.7	12.0	13.8	14.6
SGTR [181] §	-	-	-	-	-	-	-	-	-	-	-	-	-	24.6	28.4	-	12.0	15.2
SGTR [181, 196] §*	-	-	-	-	-	-	-	-	-	-	-	-	-	20.6	25.0	-	15.8	20.1
VCTree [148] ‖	60.1	**66.4**	**68.1**	-	-	-	**35.2**	38.1	38.8	-	-	-	22.0	27.9	31.3	-	-	-
BGNN [196] †*	-	59.2	61.3	-	30.4	32.9	-	37.4	38.5	-	14.3	16.5	-	31.0	35.8	-	10.7	12.6
PPDL [208] †*	-	41.6	43.6	-	33.3	36.2	-	24.8	26.2	-	20.2	22.0	-	13.6	16.5	-	12.2	14.4
PCPL [206] ‖ *	-	50.8	52.6	-	35.2	37.8	-	27.6	28.4	-	18.6	19.6	-	14.6	18.6	-	9.5	11.7
RTPB [212] †*	-	45.6	47.5	30.3	36.2	38.1	-	24.5	25.5	19.1	21.8	22.8	-	19.7	23.4	12.7	16.5	19.0
DT2-ACBS [197] §*	-	23.3	25.6	27.4	35.9	39.7	-	16.2	17.6	18.7	24.8	27.5	-	15.0	16.3	**16.7**	**22.0**	**24.4**
IETrans [199] †*	-	48.0	49.9	-	37.0	39.7	-	30.0	30.9	-	19.9	21.8	-	23.6	27.8	-	12.0	14.9
FGPL [209] †*	-	-	-	**30.8**	37.5	40.2	-	-	-	**21.9**	**26.2**	27.6	-	-	-	11.9	16.2	19.1
RepSGG ‡	55.2	62.7	65.0	16.8	22.2	24.4	34.7	**44.0**	**49.9**	10.5	14.5	17.3	**23.6**	**31.1**	**36.3**	7.2	10.0	12.3
$\text{RepSGG}_{\text{PGLA, }\lambda=0.1}$ ‡*	40.3	46.7	48.7	23.1	29.8	33.1	23.5	30.6	35.0	12.6	17.5	21.5	16.1	21.8	26.0	9.4	13.1	16.1
$\text{RepSGG}_{\text{PGLA}}$ ‡*	24.3	27.8	28.8	29.2	**39.7**	**43.7**	13.8	17.9	20.3	16.7	22.9	**28.0**	9.0	12.4	14.9	11.1	15.6	19.2

Backbone network: †ResNeXt-101-FPN §ResNet-101 ‖VGG-16 ‡ResNet-50 * debiasing technique is used

approximately 108k images, with 70% for training (≈ 82k) and the remaining 30% for testing (≈ 26k), following the same protocols [149, 148, 206, 197, 196, 208, 212, 199, 209, 181, 182, 183]. We also evaluate on the Open Images V6 dataset (OIV6) [217], which contains 126k training images, 5k testing images, 301 entities and 30 predicates for SGG tasks [196, 181, 191, 219]. For OIV6, as some entity and predicate classes are absent from the testing set, we exclusively train on the classes

that are actually present. Consequently, we use 212 entities and 21 predicates that remain in the training set.

Evaluation. We evaluate our methods following three standard evaluation tasks: 1) predicate classification (PredCls): predict predicates given ground-truth entity classes and bounding boxes; 2) scene graph classification (SGCls): predict predicates and entity classes given ground-truth entity bounding boxes; 3) scene graph detection (SGDet): predict predicates, entity classes, and entity bounding boxes. For VG150, we report results of recall@K (R@K) [136], mean recall@K (mR@K) [146, 148], zero-shot recall@K (zs-R@K) [136] for all the three evaluation tasks. In addition, we further evaluate zero-shot mean recall@K (zs-mR@K) to assess methods' ability to generalize to unseen long-tailed testing distributions. For OIV6, following previous works [220, 196], we report results of R@K, weighted mean AP of relationship detection (wmAP$_{rel}$), weighted mean AP of phrase detection (wmAP$_{phr}$), and the weighted score as score$_{wtd}$=$0.2\times$R@50$+0.4\times$wmAP$_{rel}+$ $0.4\times$ wmAP$_{phr}$. Considering the down-weighting effect of these metrics on tailed predicates, we also provide mean recall@K results.

5.5.2 Implementation Details

ResNet-50 [2] is used as the backbone network and the same hyper-parameters are used following [110, 180]. The entity detector is initialized with the weights pre-trained on COCO dataset [48, 221]. Specifically, the pre-trained weights are trained for 90k iterations with a batch size of 16, an initial learning rate of 0.01 which is decreased at the 60k-th and 80k-th iteration by a factor of 0.1 sequentially, and the weight decay of 0.0001. Images are resized such that their shorter edge is sampled from [640, 800] with a step of 32, and their longer edge does not exceed 1333 pixels. Random horizontal flip with a probability of 0.5 and random crop are used for data augmentations.

Specifically, a relative random ratio is selected from [0.9, 1] to crop along each axis respectively. To train RepSGG, the same multi-scale training is adopted following FCOS [180], except that we set the shorter edge range as [480, 800], and random crop ratio range as [0.8, 1]. The repeat factor sampling [222] with the factor of 0.02 is applied to sample more images that contain tail predicates. We first train the FCOS detector on VG150 or OIV6 for 90k iterations. The whole architecture is then trained for additional 90k iterations while freezing the backbone and entity detection heads. Finally, the entire model is jointly trained for 10k iterations, and this procedure is referred to as fine-tuning throughout the rest of the paper. Training is performed on 4 Nvidia A100 GPUs with a batch size of 32. The AdamW [223] optimizer is used with a initial learning rate of 10^{-5} which is decayed at the 80k-th iteration by a factor of 0.1, and the weight decay of 10^{-4}. The learning rates of the backbone and rep-point samplers are multiplied by a factor of 0.1. A single model is trained for all tasks, rather than separate models for each task. For the encoder, L_e is set to 1 with the same hyper-parameter setting as used in [186]. The relationship encoder is configured with $L_d = 1$, $K = 4$, $h_G = h_R = 8$, $h_A = 128$, $d_G = d_R = 32$, and $d_A = 64$ as the default settings. For the rep-point samplers, m is uniformly sampled from [1, 100].

5.5.3 Quantitative Results

5.5.3.1 Visual Genome

To compare with methods using different entity representations and those using debiasing techniques respectively, we train one model without PGLA (RepSGG) and another with PGLA (RepSGG$_{\text{PGLA}}$). In addition, since RepSGG and RepSGG$_{\text{PGLA}}$ are two extreme cases of using debiasing methods, we also add a 3rd model RepSGG$_{\text{PGLA, }\lambda=0.1}$ with balanced R@K and mR@K

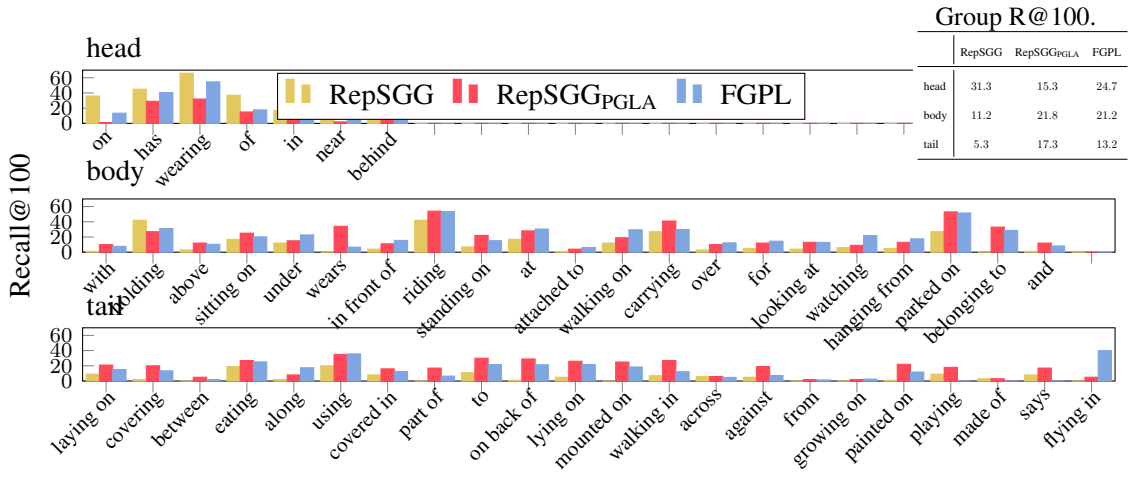


Figure 5.4: Per-predicate SGDet R@100 comparison between RepSGG, RepSGG$_{PGLA}$, and FGPL on VG150 dataset. RepSGG$_{PGLA}$ performs better on body and tail groups. The overall standard deviation of R@100 is 14.6 (RepSGG), 12.3 (RepSGG$_{PGLA}$), and 13.6 (FGPL) respectively, which also implies that RepSGG$_{PGLA}$ achieves a more balanced performance.

results for comparison. R@K is the main metric when comparing methods without debiasing, while mR@K is the main one for debiasing methods. We compare our methods with the state-of-the-art (SOTA) scene graph generation models as shown in Table 5.1. We divide the methods for comparison into 3 groups in Table 5.1 from top to bottom: point-based, query-based, and box-based, regardless of whether long-tailed techniques are used. Notably, point-based methods seek faster inference, and often do not involve debiasing techniques. Query-based methods inherently possess a certain level of debiasing capability due to the direct triplet prediction design. However, they are typically limited to performing only on SGDet. Box-based methods focus on designing debiasing methods with off-the-shelf entity detectors.

First of all, RepSGG and RepSGG$_{PGLA}$ attain state-of-the-art performance across 8 out of 18 metrics, surpassing other methods (VCTree, DT2-ACBS, FGPL) that achieve state-of-the-art results over only 3 metrics. Solely comparing methods that have different entity or predicate representations, RepSGG outperforms point-based methods FCSGG [7] and CoRF [9] on all metrics

by a large margin. Comparing with query-based methods, RepSGG outperforms RelTR [183], TraCQ [182], and SGTR [181] on most R@K metrics across all 3 tasks, while RepSGG$_{\text{PGLA}}$ also outperforms RelTR and TraCQ on mR@K metrics. As query-based methods like SGTR and TraCQ directly predict triplets consisting of entity and predicate predictions, they typically perform better on the SGDet task. Nevertheless, RepSGG$_{\text{PGLA, }\lambda=0.1}$ achieves comparable performance on R@K and mR@K w.r.t. SGTR, with higher mR@K and slightly lower R@K. Under the condition of no debiasing, the performance improvements on R@K for RepSGG indicates that the proposed entity and predicate representations are superior to point-based and query-based methods. Compared with box-based VCTree [148], RepSGG achieves higher R@K on SGCls and SGDet tasks with slightly lower recall on PredCls. In terms of debiasing methods, RepSGG$_{\text{PGLA}}$ outperforms the SOTA methods on PredCls mR@50, PredCls mR@100, and SGCls mR@100, while achieving comparable results on other metrics. RepSGG achieves 43.7 mR@100 on PredCls, which is 3.5 higher than the box-based FGPL [209]. FGPL is a complicated long-tail learning technique which requires a biased model with several fine-tuned hyper-parameters, while PGLA is much simpler yet effective for mitigating the long-tailed problem.

We further compare the per-predicate and group R@100 with FGPL for the SGDet task as shown in Fig. 5.4. Predicates are sorted in descending order based on their frequency in the training set, and divided into 3 groups following [196]. RepSGG$_{\text{PGLA}}$ achieves higher recall on body and tail classes, resulting in a more balanced performance over FGPL. In VG150 testing set, there are only 29, 37, and 12 triplets involving `playing`, `made of`, and `says`, respectively. While FGPL failed retrieving these triplets, our method achieves significantly better results, even thought these triplets are extremely rare both during training and testing. Moreover, we observe consider-

ate improvements on fine-grained predicates with similar semantics. RepSGG$_{\text{PGLA}}$ achieves better recall over predicates `sitting on`, `standing on`, `parked on`, `laying on`, `lying on`, and `painted on`. It reveals the discriminatory capability of our model among hard-to-distinguish predicates. We conjecture that the rep-point samplers and the GCA layers capture more visual context compared with box-based methods. Furthermore, by comparing the results of RepSGG$_{\text{PGLA}}$ and FGPL on a pair of visually indistinguishable predicates, `wearing` and `wears`, it becomes evident that FGPL still exhibits the bias towards the more frequent predicate `wearing`, resulting in significantly lower performance on `wears`. In contrast, RepSGG$_{\text{PGLA}}$ achieves comparable results on `wearing` and `wears`, indicating that the proposed PGLA strategy offers a more effective and balanced learning process. Notably, without debiasing techniques, RepSGG achieves excellent performance on those tail predicates (`across`, `playing`, `made of`, and `says`) as well. It demonstrates that the proposed entity and relationship representations inherently capture more informative semantics.

We further conduct analysis on zero-shot performance. In this setting, the objective is to retrieve triplets that are not encountered during training, but are present during testing. The zero-shot performance in scene graph generation is essential for achieving generalizability, robustness, adaptability, and cost-effectiveness in real-world applications, while also serving as a key metric for model evaluation and benchmarking. In Table 5.2, we report the zero-shot recall and zero-shot mean recall results on the PredCls task and compare with the SOTA methods. We collect the results by implementing the zs-mR@K evaluation following [147]. RepSGG$_{\text{PGLA}}$ outperform the SOTA methods on mR@50 and mR@100 by a large margin. Among methods for comparison, IETrans [199] achieves good results by re-labeling predictions and labeling unannotated samples

Table 5.2: PredCls results of zero-shot mean recall (zs-mR@K) and zero-shot recall (zs-R@K) on VG150 compared to state-of-the-art methods. The best results are bold, and the second-best results are underlined.

	PredCls Zero Shot Relationship Retrieval					
	mR@20/50/100			R@20/50/100		
BGNN [196]	1.9	3.2	4.9	2.0	3.5	4.6
BA-SGG [202]	3.1	5.3	6.7	3.0	6.0	8.0
Motifs-TDE [147]	5.3	9.3	11.4	8.3	<u>14.3</u>	**18.0**
FGPL [209]	**11.0**	14.3	15.9	**9.4**	13.0	<u>14.6</u>
IETrans [199]	**11.0**	**14.5**	<u>17.0</u>	6.5	10.0	12.0
RepSGG	4.1	7.1	8.9	<u>8.9</u>	**14.6**	**18.0**
$\text{RepSGG}_{\text{PGLA}}$	<u>10.3</u>	**17.2**	**20.0**	6.3	9.3	11.1

from biased models for training. Without extra data for training, $\text{RepSGG}_{\text{PGLA}}$ outperforms IETrans by 3.0 on zs-mR@100. RepSGG also achieves the SOTA zero-shot recall performance over zs-R@50 and zs-R@100. Motifs-TDE employs a debiasing technique to achieve a zs-R@100 of 18.0, whereas RepSGG achieves the same results without using debiasing techniques. It demonstrates that RepSGG generalizes significantly better to compositions of entities and relationships in unseen contexts.

5.5.3.2 Open Images

Since the precision is one of the evaluation metrics for OIV6, we conduct experiments using precision as the PGLA metric besides recall. The model trained with precision-guided logit

Table 5.3: Comparisons with the state-of-the-art methods on OI V6. R@50 in the table is micro-Recall@50 [11]. The best results are bold, and the second-best results are underlined.

	mR@50	R@50	wmAP$_{rel}$	wmAP$_{phr}$	score$_{wtd}$
Motifs [149]	32.68	71.63	29.91	31.59	38.93
RelDN [220]	33.98	73.08	32.16	33.39	40.84
VCTree [148]	33.91	74.08	34.16	33.11	40.21
G-RCNN [159]	34.04	74.51	33.15	34.21	41.84
GPS-Net [173]	35.26	74.81	32.85	33.98	41.69
BGNN [196]	40.45	74.98	33.51	34.15	42.06
SGTR [181]	42.61	59.91	<u>38.73</u>	<u>36.98</u>	42.28
CSL [219]	41.72	75.44	34.30	35.38	42.86
SS R-CNN [191]	50.73	75.70	**41.14**	**43.24**	**48.89**
RepSGG	<u>62.68</u>	<u>77.70</u>	30.01	29.58	39.38
RepSGG$_{\text{PGLA/R}}$	**64.26**	76.40	31.64	31.27	40.44
RepSGG$_{\text{PGLA/P}}$	53.87	70.25	32.53	32.47	40.05
RepSGG$_{\text{X101}}$	56.32	**77.83**	36.73	36.61	<u>44.90</u>

adjustment is denoted as RepSGG$_{\text{PGLA/P}}$, and the model trained with recall-guided logit adjustment is renamed to RepSGG$_{\text{PGLA/R}}$. We also train a model with the ResNeXt-101-32$\times$8d [43] backbone, denoted by RepSGG$_{\text{X101}}$ without using PGLA. The experimental results on OIV6 [217] are shown in Table 5.3. We observe that all RepSGG models outperforms other methods on mR@50 by a large margin. RepSGG$_{\text{PGLA/R}}$ achieves a mR@50 of 64.26, marking a 13.53-point increase, or a 26.7% improvement over the previous SOTA method SS R-CNN [191]. RepSGG$_{\text{PGLA/P}}$ achieves better results on wmAP$_{rel}$ and wmAP$_{phr}$ in comparison to RepSGG and RepSGG$_{\text{PGLA/R}}$, highlighting the effectiveness of precision-guided PGLA for precision-oriented tasks. As a compromise, the recall performance is lower compared with PGLA/R. With a larger backbone network, RepSGG$_{\text{X101}}$

achieves the highest R@50 performance of 77.83, and the second-best score$_{wtd}$ of 44.90, which is a 2.62-point improvement over SGTR. Our methods achieves lower precision-oriented metrics like wmAP$_{rel}$ and wmAP$_{phr}$. This is because OIV6 has very sparse annotations, while our model has great generalization power as shown in Table 5.2. OIV6 has 2.76 relationship annotations per image on average in the training set, while VG150 has 5.97. As a result, most detections will be considered as false positives, leading to lower precision. We then collect the results on other SGG tasks to further analyze the performance as shown in Table 5.4. The results on R@50, mR@50, and wmAP$_{rel}$ show significant improvements on PredCls and SGCls tasks. In the PredCls setting, all RepSGG models achieve R@50 and wmAP$_{rel}$ over 90. Both Table 5.1 and 5.3 demonstrate that our model's primary bottleneck lies in entity detection rather than predicate prediction. Consequently, there are fewer improvements observed in SGDet and SGCls tasks compared to PredCls.

Table 5.4: Comparisons of RepSGG models without PGLA, with recall-guided LA, and with precision-guided LA on PredCls and SGCls tasks.

	PredCls			SGCls		
	mR@50	R@50	wmAP$_{rel}$	mR@50	R@50	wmAP$_{rel}$
RepSGG	69.40	<u>97.33</u>	<u>93.19</u>	<u>66.59</u>	**90.56**	<u>54.21</u>
RepSGG$_{\text{PGLA/R}}$	**80.33**	96.69	89.08	**71.95**	<u>90.06</u>	**55.46**
RepSGG$_{\text{PGLA/P}}$	<u>73.44</u>	95.36	92.51	55.09	74.99	51.30
RepSGG$_{\text{X101}}$	66.97	**97.48**	**93.60**	63.04	85.81	52.09

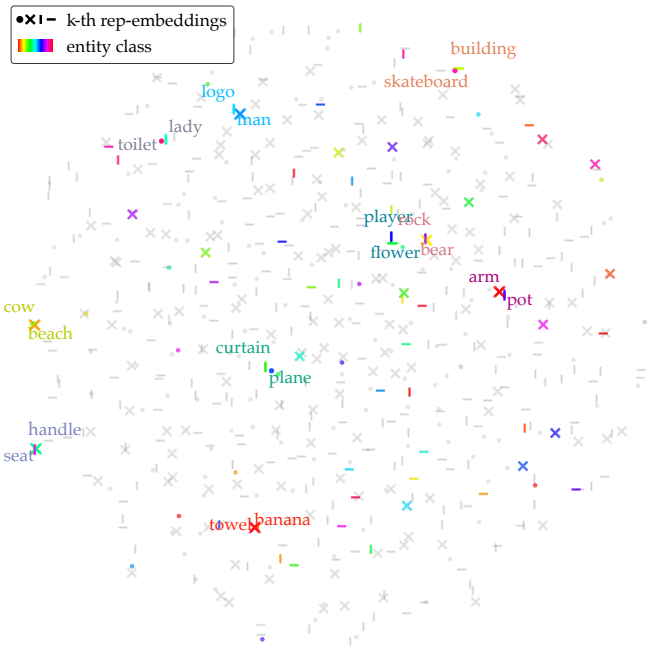


Figure 5.5: The t-SNE visualization of subject rep-embeddings $\mathbf{E}_s$, projected to $\mathbb{R}^{C\times K\times 2}$ with pairwise cosine similarity. There are $C \times K = 150 \times 4 = 600$ points in total, and each point represents a subject rep-embedding in the projected 2D space. The top-10 similar pairs are labeled. Rep-embeddings of the same entity class share a color. Only the entity classes involved in the top-10 pairs are colored, while the others are displayed in gray.

5.5.4 Qualitative Analysis

We want to qualitatively examine what the model learns from the data, especially on the rep-embeddings and rep-points. We visualize the weights of subject rep-embeddings $\mathbf{E}_s$ of the trained RepSGG$_{\text{PGLA}}$ (with $K = 4$, $L_e = 1$, and $L_d = 1$) via t-SNE [224] as shown in Fig. 5.5. The top 10 pairs of similar rep-embeddings are highlighted, while the remaining rep-embeddings belonging to the involved entities are colored. We use the pairwise cosine similarity as the distance metric for t-SNE where distances represent the similarities between rep-embeddings. It reveals that the rep-embeddings between different entity classes are well separated. Rep-embeddings of the same entity class are distinctly separated as well. Interestingly, most pairs do not exhibit explicit

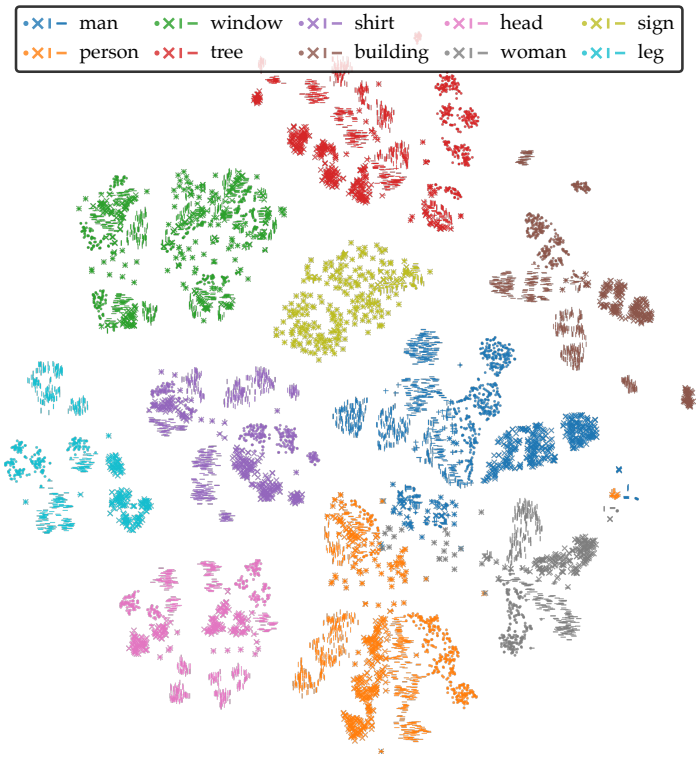


Figure 5.6: The t-SNE visualization results on the output subject queries of the relationship encoder ($\mathbf{Q}^{L_d}$) for 10 frequent entity classes.

semantic part affinities. We hypothesize that the initial rep-embeddings serve as "anchors" which are evenly distributed in the embedding space, and capture more semantic information as they progress through the relationship encoder.

We further validate the hypothesis by visualizing the output queries $\mathbf{Q}^{L_d}$. To eliminate entity mis-classification, we collect the output queries on the PredCls task from the first 1000 testing images. As shown in Fig. 5.6, the inter-class rep-embeddings are distinguishable. Entities with similar semantics, such as `woman`, `man`, and `person`, form more compact clusters. The compactness and distinctness of rep-embeddings differ within an entity class. For `building`, 4 types of rep-embeddings are well-separated with high variance where each type is responsible for a different semantic concept. For `sign`, the distribution of rep-embeddings is more compact, indicating that

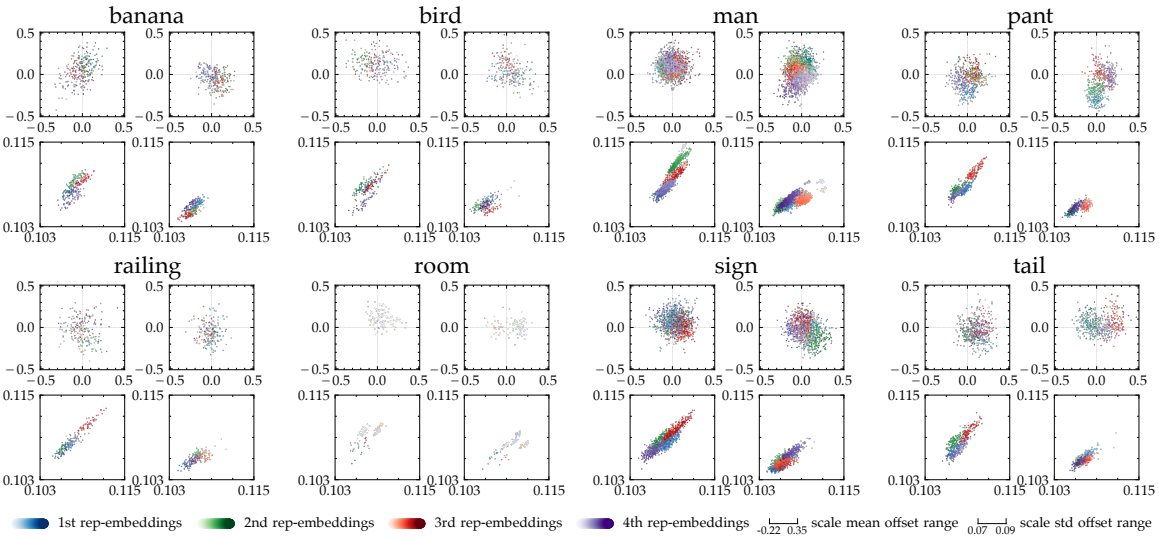


Figure 5.7: Visualizations on predicted subject and object rep-point mean and std offsets. For each entity class, the offsets of subject means, object means, subject stds, and object stds are shown on the top-left, top-right, bottom-left, and bottom-right of the sub-figure. The coordinates represent spatial offsets w.r.t. the ground-truth centers, while the color saturation denotes the scale offsets w.r.t. the entity feature scale. The more saturated the color is, the larger the scale offset is, and vice versa.

the semantics and relationships associated with `sign` are mostly homogeneous.

We also collect the subject and object rep-point offset parameters $\boldsymbol{\mu}_s^1$, $\boldsymbol{\sigma}_s^1$, $\boldsymbol{\mu}_o^1$, and $\boldsymbol{\sigma}_o^1$, which are shown in Fig. 5.7. The variation in parameters is evident, primarily between different entity classes, and between subject and object offsets. The distribution of parameters implies the locations of relevant semantic features to a certain extent. It is likely that the spatial means follow a bivariate Gaussian distribution especially for `man` and `sign`, which suggests that rep-points are sampled around the bounding box centers. Entities with larger location variances within bounding boxes, such as `banana`, `bird`, and `railing`, exhibit more dispersed distributions of spatial mean offsets. The distributions of subject and object mean offsets for `man` are noticeably distinct. The subject means are more tightly clustered around centers, indicating that when `man` is a subject, the features representing the entirety of a `man` are more significant. Conversely, when `man` is an object,

the features representing semantic parts become more important as object means are much more distributed. The scale mean offsets also reveal different patterns among entities. For entities with large bounding boxes like `room`, the scale means are primarily negative, indicating that the sampled rep-points are from lower scales of features. For entities in medium and small sizes, the sampled rep-points are mostly from the same or higher scales of features. In terms of the standard deviations (stds) of offsets, we observe the spatial collinearity, which is a natural occurrence in training data with diverse entity sizes. It is notable that `railing` has larger spatial and scale stds, reflecting the uncertainty associated with the varying lengths of railings. The scale stds normally remain within a narrower range (0.07 to 0.09) compared to the spatial stds (0.103 to 0.115).

5.5.5 Ablation Studies

To further investigate the proposed methods, we perform ablation studies on VG150 dataset. Unless specified, we use the same hyper-parameters as discussed in Section 5.5.2, and PGLA is applied but fine-tuning is not applied.

5.5.5.1 Analysis of RepSGG

We investigate the effects of different numbers of rep-embeddings, deformable encoder layers, and relationship encoder layers. The results on mR@100 and zs-mR@100 are listed in Table 5.5. First, adding the deformable encoder or relationship encoder improves the overall performance. It is also important to note that similar behaviors can be observed when increasing the values of K, L_e, or L_d, as the performance improves and then plateaus at certain values of K, L_e, or L_d, respectively. The combination of $K = 4$, $L_e = 1$, and $L_d = 1$ achieves a balanced trade-off

Table 5.5: Ablation studies of number of rep-embeddings K, number of encoder layers L_e, and number of decoder layers L_d in RepSGG$_{\text{PGLA}}$. Results on mR@100 and zs-mR@100 are collected for three SGG tasks.

K	L_e	L_d	PredCls @100		SGCls @100		SGDet @100	
			mR	zs-mR	mR	zs-mR	mR	zs-mR
1	1	1	**42.5**	18.5	20.0	6.5	15.7	5.5
4	1	1	42.0	**20.6**	19.9	6.0	**15.9**	5.4
7	1	1	41.1	18.8	19.6	6.2	15.7	5.6
10	1	1	40.7	20.0	19.1	6.6	15.4	**5.9**
4	0	0	39.1	16.2	18.0	5.9	13.8	5.0
4	1	0	39.8	18.2	19.1	6.3	15.0	5.1
4	0	1	41.0	16.8	19.4	6.1	15.7	5.5
4	2	2	40.7	18.2	19.5	5.9	15.2	5.4
10	3	3	41.0	19.6	**20.0**	**7.1**	15.1	5.5

Table 5.6: Ablation studies of GCA and RCA.

GCA	RCA	PredCls @100		SGCls @100		SGDet @100	
		mR	zs-mR	mR	zs-mR	mR	zs-mR
		39.8	18.2	19.1	6.3	15.0	5.1
✓		40.3	17.3	19.0	5.5	14.4	4.8
	✓	41.8	18.7	**19.9**	**6.6**	15.7	**5.7**
✓	✓	**42.0**	**20.6**	**19.9**	6.0	**15.9**	5.4

between the performance and model complexity. Based on our experiments, four rep-embeddings per entity class ($K = 4$) are sufficient to capture possible relationships, and using $K = 10$ po-

Table 5.7: Ablation studies on using separate subject and object rep-embeddings ($\mathbf{E}_s \neq \mathbf{E}_o$) vs. identical rep-embeddings ($\mathbf{E}_s = \mathbf{E}_o$).

L_e	L_d	Separate	PredCls @100		SGCls @100		SGDet @100	
			mR	zs-mR	mR	zs-mR	mR	zs-mR
0	0		39.1	15.5	**18.2**	5.5	13.8	**5.1**
0	0	✓	39.1	**16.2**	18.0	**5.9**	13.8	5.0
1	1		40.2	17.0	19.1	4.6	14.4	5.0
1	1	✓	**42.0**	**20.6**	**19.9**	**6.0**	**15.9**	**5.4**

tentially causes overfitting on few re-embeddings. Likewise, increasing the number of deformable encoder or relationship encoder layers does not lead to significant performance improvement. When both deformable encoder and relationship encoder are removed, ReSGG$_{\text{PGLA}}$ still achieves a high PredCls mR@100 of 39.3, which outperforms many state-of-the-art box-based methods listed in Table 5.1. This suggests the advantage of our proposed relationship representation over traditional softmax classification. Treating relationships as attention weights naturally incorporates more semantic information, effectively capturing the distinction between subject and object entities.

We further conduct the analysis of GCA and RCA modules in the relationship encoder. We use the model with $K = 4$, $L_e = 1$, and $L_d = 1$. The model without either GCA or RCA is equivalent to the one with hyper-parameters $K = 4$, $L_e = 1$, and $L_d = 0$ as shown in Table 5.5. We further train 3 separate models, one with GCA only, one with RCA only, and one with both modules. When GCA is not equipped, we use the visual features sampled at entity centers as the inputs to RCA, without using rep-point samplers. The ablation results are shown in Table 5.6. When both GCA and RCA are not used, the model achieves relatively lower results compared with when both are used except SGCls zs-mR@100. No significant performance drop is observed when

removing GCA or RCA. RCA is more important than GCA based on the results, as the RCA-only model achieves better performance on most metrics. We conjecture that RCA exchanges semantic information among rep-embeddings, while GCA only exchanges visual features locally. When both GCA and RCA are applied, we obtain the best results on most metrics except on SGDet zs-mR@100.

We also investigate the effectiveness of using separate rep-embeddings to represent subjects and objects. Specifically, we keep $K = 4$ for this experiment, and conduct 2 groups of ablation studies, with either $L_e = L_d = 0$ or $L_e = L_d = 1$, and train with separate rep-embeddings ($\mathbf{E}_s \neq \mathbf{E}_o$) or identical rep-embeddings ($\mathbf{E}_s = \mathbf{E}_o$). The results are shown in Table 5.7. In the absence of both the entity encoder and relationship encoder, there is minimal performance difference observed between the two settings. It is difficult for the model to predict relationships without the entity encoder and relationship encoder, which are responsible for capturing more visual and semantic context. However, upon adding a single layer of both the entity encoder and relationship encoder, significant performance improvements are observed across all metrics and tasks. This clearly illustrates the benefits of using separate rep-embeddings for subjects and objects in distinguishing semantics in relationship inference. We hypothesize that predicting certain relationships becomes challenging for the model when it lacks information about which entities serve as subjects or objects.

5.5.5.2 Analysis of PGLA

To investigate the effects different loss-related configurations proposed in the paper, we conduct the ablation experiments on Logit Adjustment [187], the proposed PGLA, the margin rank-

Table 5.8: Ablation studies of loss and training configurations.

LA	PGLA	$\mathcal{L}_\eta$	finetune	PredCls @100		SGCls @100		SGDet @100	
				mR	zs-mR	mR	zs-mR	mR	zs-mR
				27.1	9.7	15.9	5.1	11.4	4.1
✓				39.4	16.2	19.4	5.5	13.7	4.4
	✓			40.3	<u>18.4</u>	<u>21.4</u>	5.8	15.0	5.1
	✓	✓		<u>41.2</u>	**20.1**	<u>21.4</u>	<u>6.6</u>	<u>15.3</u>	<u>5.6</u>
	✓	✓	✓	**43.7**	**20.1**	**27.7**	**7.1**	**18.7**	**5.9**

ing loss $\mathcal{L}_\eta$ proposed in Section 5.4.5.1, and fine-tuning. As shown in Table 5.8, RepSGG$_{\text{PGLA}}$ achieves higher mean recall on all metrics compared with RepSGG trained with LA, which confirms the effectiveness of PGLA. RepSGG$_{\text{PGLA}}$ also achieves a more significant improvement on zs-mR@100, providing evidence that PGLA is more resilient to under-fitting or over-fitting. Using $\mathcal{L}_\eta$ with PGLA brings more improvements, as it effectively suppresses unlikely relationships and avoids excessively penalizing potentially unannotated ones. Finally, with additional fine-tuning the entire model, we manage to further increase the performance on mean recall.

We study the individual and combinatorial effects of $\mathbf{W}$, $\mathbf{B}$, and $\mathbf{D}$ in (5.16). We simply set $\mathbf{W} = \mathbf{1}$ when $\mathbf{W}$ is not applied, and $\mathbf{B}$ or $\mathbf{D}$ to zero when they are not applied. As shown in Table 5.9, introducing either component improves the performance on mean recall. The bias term $\mathbf{B}$ has the most significant effect over the results with the largest improvement compared with individual application of $\mathbf{W}$ or $\mathbf{D}$. This can be attributed to the fact that class margins are primarily determined by the bias. Similarly, $\mathbf{D}$ serves as an additional adjustment for class margins, which has more effect over $\mathbf{W}$ as a result. Applying $\mathbf{B}$ alone already yields the highest mR, but considering

Table 5.9: Effects of $\mathbf{W}$, $\mathbf{B}$, and $\mathbf{D}$ in PGLA.

			PredCls @100		SGCls @100		SGDet @100	
W	**B**	**D**	mR	zs-mR	mR	zs-mR	mR	zs-mR
			27.1	9.7	15.9	5.0	11.4	4.3
✓			28.5	9.4	17.4	5.4	12.8	4.5
	✓		**42.0**	17.7	<u>21.3</u>	6.5	15.2	<u>5.8</u>
		✓	30.5	10.0	18.8	5.7	13.6	4.8
	✓	✓	40.5	16.8	20.5	6.9	<u>15.3</u>	5.7
✓		✓	30.5	10.2	17.8	5.9	13.5	4.7
✓	✓		<u>41.6</u>	<u>18.7</u>	**21.4**	<u>7.0</u>	**16.0**	**5.9**
✓	✓	✓	41.2	**20.1**	**21.4**	**7.1**	<u>15.3</u>	<u>5.8</u>

zs-mR is also crucial for real applications of scene graphs. Combing $\mathbf{W}$ and $\mathbf{B}$ enhances the zs-mR performance, and combing all 3 components further improves the zs-mR performance with a slightly decrease on mR.

The hyper-parameter λ in (5.14) allows for adjusting the sensitivity to run-time recall, and we explore its effects on recall and mean recall. We evaluate on all 3 SGG tasks with different values of λ, and the results are shown in Fig. 5.8. As anticipated, increasing λ results in higher mR@100 but lower R@100, whereas decreasing λ leads to lower mR@100 but higher R@100. Although better trade-offs may exist, exploring them falls beyond the scope of this paper.

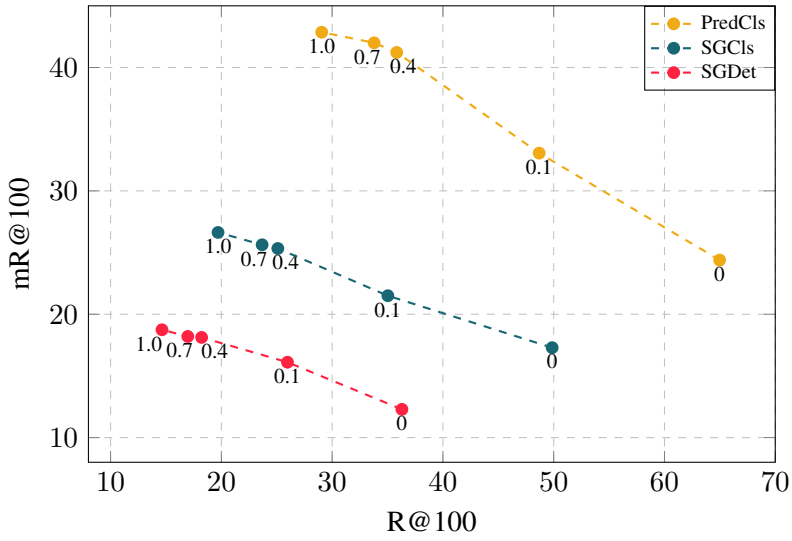


Figure 5.8: Effects of λ in (5.14) on R@100 and mR@100. The value of λ used for each model is annotated, where $\lambda = 0$ denotes "PGLA is not applied", and $\lambda = 1$ denotes the default PGLA.

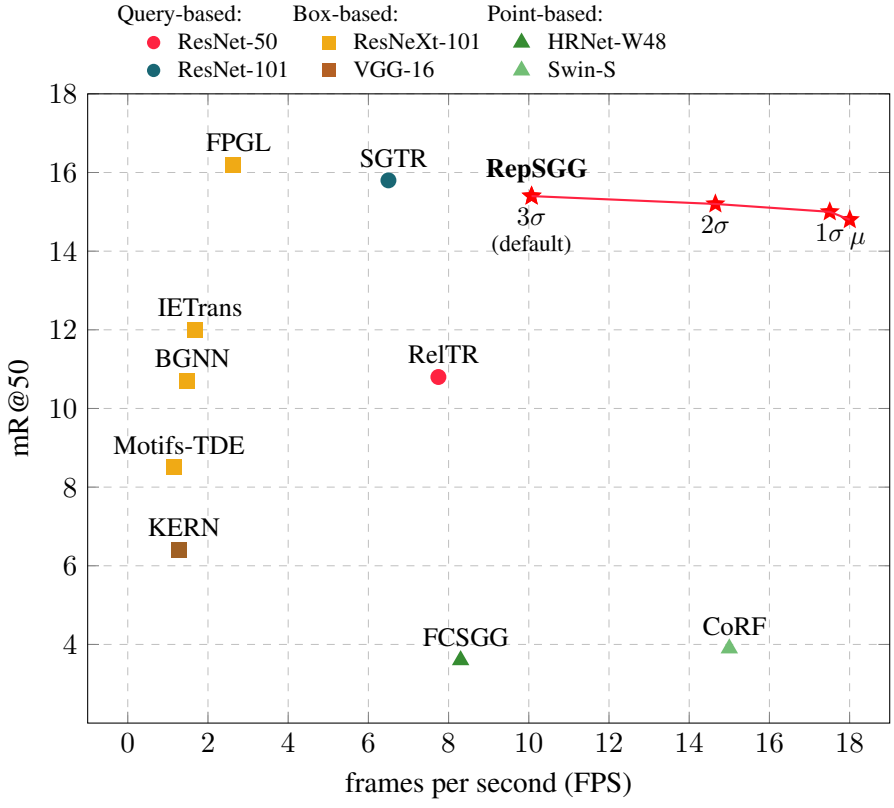


Figure 5.9: Inference speed and mR@50 benchmark on the SGDet task.

5.5.5.3 Analysis of Inference Speed

Beside performance improvements, another set of benefits of RepSGG are its fast inference speed and flexible inference configurations. In Section 5.4.3.2, the details of inference are discussed where we sample rep-points within "3σ". We have the option to perform inference with fewer samples by changing the hyper-parameter ξ during inference. We conduct experiments using samples within 3σ, 2σ, σ, and just the samples at the means μ. We also benchmark several methods discussed in Section 5.2 as comparisons. We measure the average of frames per second (FPS) over all testing images for all selected models, which is evaluated on a single Nvidia GTX 1080 Ti GPU with a batch size of 1. The settings from different models follow their original papers and implementations, so the inference speed depends on not only the architectures, but also inference configurations such as image size, mixed precision, *etc*.

The results of mR@50 and FPS are collected on the SGDet task as shown in Fig. 5.9. Remarkably, the proposed methods achieve better trade-offs between performance and speed. No crucial performance drop is observed when less rep-points are sampled. By sampling within 3σ, our model achieves competitive results compared to the state-of-the-art box-based FPGL and query-based SGTR methods, while being approximately 4.1$\times$ and 1.6$\times$ faster, with only 4.9% and 2.5% mR@50 performance drop, respectively. By only sampling at the means, RepSGG$_{\text{PGLA}}$ achieves 14.8 mR@50 with a nearly real-time inference speed of 18 FPS, a speedup of approximately 6.8$\times$ and 2.7$\times$, with only 8.6% and 6.3% mR@50 performance drop, compared to FPGL and SGTR, respectively.

5.6 Limitations and Future Work

RepSGG excels on PredCls and SGCls tasks, showcasing an advantage over query-based methods [181, 182, 191]. Nevertheless, its performance on SGDet does not outperform the state-of-the-art methods, because neither it fully leverages ground-truth information as effectively as box-based methods, nor it is directly optimized on the SGDet task like query-based methods. Giving the GT bounding boxes, we must map the corner coordinates to appropriate feature levels and identify the best-matched pixels responsible for the detections. However, the best-matched features do not correspond as accurately to actual ground-truth features as achieved by RoIAlign [20]. A hybrid representation of box, point, and query features could help improve the performance. Additionally, it is important to note that our model is exploratory and primarily focuses on visual features. The integration of multi-modal features, such as depth maps [225], language [136], video [226], and knowledge graph [151], can be seamlessly incorporated into our RepSGG architecture as additional queries and keys. The architecture also offers potential for extension to other relationship-related tasks, such as human-object interaction (HOI) detection [227], video-based HOI detection [228], video SGG [229], panoptic SGG [230], and panoptic video SGG [231].

The proposed PGLA serves as a general approach for leveraging performance evaluation to attain a more balanced performance. It can also be adapted for addressing other long-tailed problems (*e.g.*, in visual recognition), utilizing different metrics (such as accuracy), and incorporating other loss functions (such as cross-entropy) with minor adjustments.

Chapter 6

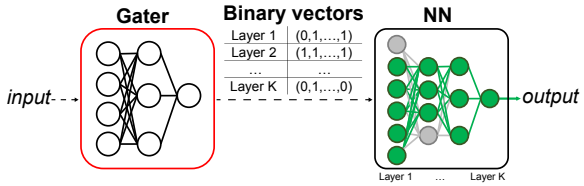
Dynamically Throttleable Neural Networks

6.1 Introduction

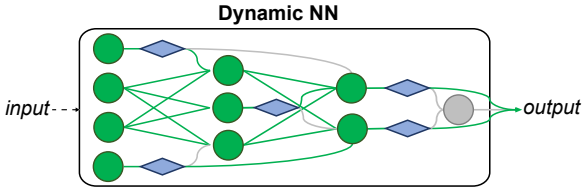
Recently, deep neural networks (DNNs) are prevailing in computer vision applications on edge devices and autonomous vehicles where real-time response is needed. The computation power and memory budget are drastically different across devices. Even on the same device, the runtime performance varies given different battery conditions, operation temperatures, *etc*. Researchers try to design lightweight neural networks [232, 233, 234, 235, 236, 237] or perform neural architecture search (NAS) [238, 239, 240, 241, 242] with computation complexity constraints. Others seek ways of network pruning [243, 244, 245, 246], model distillation and compression [247, 248, 249, 250] or quantization [251, 252, 253, 254, 255] to reduce the memory footprint. These approaches typically provide offline-trained static models with a constant allocation of computation and memory

resource. After training and deploying, the model is fixed such that the whole network has to be executed. However, the conditions in real-world setting are often different, whereby the run-time inference is neither optimal from an accuracy or efficiency perspective. The problem lies in that the naive training approaches can only produce static models with a specialized trade-off between performance and resource utilization. By leveraging run-time filter selection inspired from Dropout [256, 257], this paper presents an adaptive system named dynamically throttleable neural network (DTNN) to tackle challenges in highly dynamic deployment environments. DTNN consists of a context-aware controller and a throttleable neural network. We use the term "throttleable" to suggest the ability of the model to adaptively balance performance and resource use in response to a control signal named *utilization*. TNNs flexibly support diverse and dynamic configurations under different resource constraints without re-training or re-deploying. The contextual controller learns the policy to generate the input-dependent utilization as the control signal for TNNs. For example, the controller predicts lower utilization for "easy" input data (*e.g.*, video sequences of static objects), and higher utilization for "challenging" one (*e.g.*, video sequences of moving pedestrians). Optimizing the best trade-off between performance metrics and utilization is carried out via deep contextual reinforcement learning [258, 259, 260]. In short, the controller determines how much to throttle, and the TNN determines how to throttle.

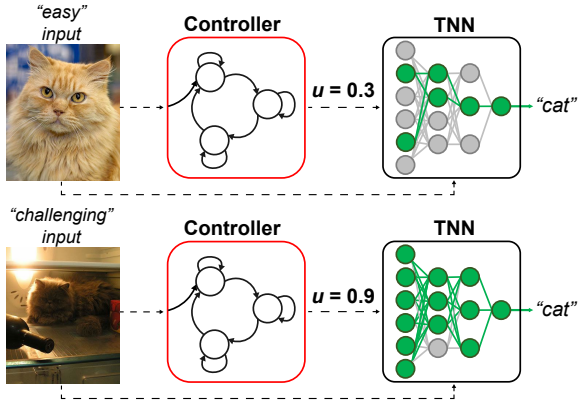
A major benefit of DTNN is the separation of the controller and TNN. Any controller that provides a single scalar can be used, such as heuristic and learnable policies. That is, the controller can be as simple as a fixed utilization parameter, or a state-machine with trainable policies. Moreover, by using a *single* utilization parameter u to conditionally gate part of the network, the complexity of the controller is much reduced while retaining highly flexible inference paths. In



(a) An example diagram of network with conditional gating mechanism [261, 262]. The "gater" generates the gating functions as binary masks and applies for all computation nodes.



(b) An example diagram of adaptive control of neural networks [259, 260, 263, 264]. The execution of a computation node is determined by a control node ◆. As a result, only a subset of the inference graph is executed.



(c) Our proposed dynamically throttleable neural network. Based on the contextual information of the input data, the controller decides how much utilization of TNN is needed. Different from Figure 6.1a, gating is controlled by one scalar, the utilization u, instead of vectors. The dynamic execution is decoupled instead of entangled as in Figure 6.1b. Our DTNN enables much more robust and flexible configurations.

Figure 6.1: Conceptual architectures: (a) an ordinary gating network, (b) an ordinary dynamic network, (c) the proposed DTNN. A computation node ○ can represent a single kernel, a group of kernels, a layer or a group of layers based on different designs. An executed computation node is drawn as ● while an unexecuted or gated node is drawn as ●.

addition to the input data, TNN only takes an extra input, the utilization, for task-specific predictions. In comparison, standard gating networks (Figure 6.1a) produce a large number of vectors as control signals that make it hard to train and fine-tune the gating network. Another advantage of

DTNN is its high flexibility and modularity. TNN consists of neural kernels which are structured into "blocks", while conditional computation is applied at the level of individual block. The gating strategies consider different dimensions of the network ("width-wise" or "depth-wise") as well as the order of gating ("independent" or "nested"). The training approach for TNNs essentially enforces sparsity in the kernels such that there is no catastrophic accuracy loss as run-time utilization decreases in comparison to a naively trained network. TNNs are trained by maximizing task performance metric over all the utilization settings, while optionally minimizing the computational cost for an application under conditions that can change over time.

6.2 Related Work

We focus on the research related to conditional computation and adaptive control of DNNs [265]. Design of efficient architectures [233, 234, 266, 267, 268], NAS [238, 240, 269, 270], and static model compression [243, 244, 245, 247, 248, 249, 250, 271, 272, 273, 274, 275] are different from our approach such that we have multiple operational points within a single architecture for performance and efficiency trade-off, rather than a single static model. In fact, most architectures can be easily converted into a throttleable one without losing the peak performance which will be demonstrated later in Section 6.5 and 6.5.6.2.

6.2.1 Conditional Computation

Our work builds on conditional computation [280] where portions of the model are executed for prediction to reduce overall computational load. Different from static or post-training pruning, sub-models are generated on the fly to achieve dynamic model size or computation re-

Table 6.1: Theoretical comparisons of the DTNN and related work on conditional computation. The proposed method supports both width-wise and depth-wise gating. It can achieve static inference using fixed utilization, or per-input dynamic inference with a learnable contextual controller without fine-tuning or re-training the throttleable neural network.

	Architecture Support				Reported Experimental Results			
	Width-wise	Depth-wise	Inference		Image classification	Object detection	Video classification	Embedded
	# OP	# OP	Static	Dynamic				
DTNN	$\mathcal{O}(2^{CL})$	$\mathcal{O}(2^{L})$	✓	✓	✓	✓	✓	✓
DEN [276]	$\mathcal{O}(2^{CL})$	$\mathcal{O}(2^{CL})$	✓		✓			
GaterNet [262]	$\mathcal{O}(2^{CL})$	$\mathcal{O}(2^{CL})$		✓	✓			
Slimmable [277]	$\mathcal{O}(N)$		✓		✓	✓		
NestedNet [278]	$\mathcal{O}(N)$		✓		✓			
Spasov & Liò [260]	$\mathcal{O}(2^{CL})$			✓	✓			
RNR [263]	$\mathcal{O}(2^{CL})$			✓	✓			
D^{2}NN [259]		$\mathcal{O}(N)$	✓		✓			
BlockDrop [261]		$\mathcal{O}(2^{L})$		✓	✓			
SkipNet [279]		$\mathcal{O}(2^{L})$		✓	✓			
ConvNet-AIG [264]		$\mathcal{O}(2^{L})$		✓	✓			

OP: the theoretical maximum number of operational points or sub-models within the architecture (the actual # OP depends on implementations).

Static: the sub-model is manually selected from several available configurations. Dynamic: the sub-model is selected dynamically based on the input.

C: the maximum number of channels among all layers.

L: the number of layers within the architecture.

N: the number of configurations within the architecture.

DTNN can have at most $\mathcal{O}(2^{CL})$ operational points when applying both width-wise and depth-wise gating within the architecture.

source. An intuitive idea is the early prediction where the inference stops at an early stage of the network once a criterion or "confidence" is satisfied. Such examples include Adaptive Computation Time (ACT/SACT) [281], BranchyNet [282], and Dynamic Time Recurrent Visual Attention (DT-RAM) [283], where essentially gating is applied for all the kernels after the decision layer. Convolutional neural mixture model (CNMM) [284], as an example, is the ensemble of convolutional neural networks (CNNs) that are sampled during inference with "early-exit" classifiers. However, vision tasks in real-world environments are challenging and complex, depending only on low-level features is not practical. This work leans more towards encouraging sparse activations throughout the entire network rather than discarding high-level semantic features.

Beyond early prediction, there are other methods that reduce computations conditionally. Shazeer *et al*. [285] proposed the Mixture-of-Experts layer that learns to rank and select the top several sub-networks. Similarly, in [286], a gating module is proposed to select among multiple branches of networks of features for each input. NestedNet [278] constructs a nested architecture with several levels of sparsity. Stochastic Depth [287], Skipnet [279], ConvNet-AIG [264] and BlockDrop [261] are similar approaches that learn to bypass ResNet [2] blocks based on the input. Similar ideas also appear in recent work on neural architecture search, such as EfficientNet [288] that studies the model scaling of depth, width and resolution, and OFA [242] that searches sub-networks for specialized edge devices. Dynamic channel pruning [289, 290, 263] can also be categorized as conditional computation which performs selective convolutions during run-time.

6.2.2 Adaptive Control

More recently, the research is moving towards dynamically configuring the network topology at run-time based on input. In this line of research, no parallel networks are explicitly defined,

instead, a single sub-network is selected from the super-network by partially activating model components such as filters and layers for each input. For example, Slimmable Neural Networks [277] can scale the network width to pre-defined configurations. GaterNet [262] uses a separate gating network to generate sparse binary masks for the backbone network in an input-dependent manner. Odena *et al*. [291] introduce a "Composer" module to select the computational graph. D^2NN [259] uses reinforcement learning to jointly learn the parameters of computation nodes and control nodes. Similarly, Spasov *et al*. [260] propose a channel based selection method by casting the gating function as a multi-armed bandit problem. Ahn *et al*. [276] consider the dynamic network as an estimator-selector framework for multi-task learning such that the candidate network is optimized for one specialized task.

6.2.3 Summary

The key difference for TNNs as compared to previous work is the flexible integration of the network modules with the control decision nodes. Compared with recent work theoretically, the TNN itself provides a much more flexible model selections and applications (Table 6.1). TNNs subsume these models [261, 278, 277]. In particular, [277, 278] are subsumed under our nested width-wise gating strategy with limited number of operational levels. BlockDrop [261] is only applicable to ResNet-type networks with skip-connections which can be summarized as our nested depth-wise gating strategy. In terms of adaptive control, [259, 260, 262, 291] generate sequences of control nodes as a form of vectors. The control and backbone networks are tightly coupled before fine-tuning training is further applied. Our approach, on the other hand, introduces the single-scalar utilization parameter to control the backbone network that is user-friendly and semantically meaningful. With the utilization parameter, the control and backbone network can be trained and

optimized independently. Furthermore, our two-phase training does not require fine-tuning the network jointly. The TNN backbone can work by itself and preserve a monotonic performance without the controller. Other adaptive control approaches can not guaranteed this performance since their control is either integrated into the network or limited by complicated gating functions. Whereas most previous papers show results for image classification only, we demonstrate our approach with insights and provide results for object detection, video classification, and hardware performance.

6.3 Contributions of this Chapter

We are contributing to the body of research on conditional computation and adaptive control (Section 6.2) with DTNN to improve model performance and address efficient inference with tight resource budgets such as memory, power, and compute capability. For vision-based systems, this is particularly important, as the workload and processing throughput are demanding and dynamic. To the best of our knowledge, we offer the following contributions in this chapter:

1. A novel architecture called throttleable neural networks with plug-and-play throttleable blocks comprised of convolutional or fully-connected filters, and the usage of model capacity is controlled by a single utilization parameter.

2. A lightweight context-aware controller trained to regulate TNNs' performances and computing needs with learnable control policies.

3. Comprehensive experimental results on image classification, object detection and video-based hand gesture recognition tasks that demonstrate the effectiveness of TNNs, while most of the previous work only shows results for image classification.

4. A use case of DTNN for hand gesture recognition system which outperforms the vanilla architectures and all fixed utilization settings.

5. Physical power and run-time measurements on an embedded GPU that demonstrate the efficacy and practicality of TNNs.

6.4 Technical Approach

6.4.1 Objective and problem setting

Our goal is to enable TNN with dynamic performance during inference while having the same or similar number of parameters with vanilla ones. Furthermore, TNN should be modular such that (1) they are easily trained with gating policies that can be fixed or learned, (2) the controller can be trained separately without re-training the TNN, (3) the framework is largely model-agnostic.

A neural network is a function $\mathrm{T}_{\Phi}(\mathbf{x})$ parameterized by Φ that maps an input data $\mathbf{x}$ to an output $\hat{\mathbf{y}}$. For example, $\mathbf{x}$ can be an image or a video sequence, and $\hat{\mathbf{y}}$ is the prediction of class label for a classification problem. We define a *throttleable neural network* as a function of two variables, $\mathrm{T}_{\Phi}(\mathbf{x}, u)$, where $u \in (0, 1]$ is a control utilization parameter that indicates how much "computational effort" the network should exert. We emphasize that u is an additional input to the network; after training is complete, the network parameters Φ are fixed but u can change.

The problem is then formulated as the minimization of the "task loss" under all utilization settings, and the general objective of training TNNs is

$$\min \mathbb{E}_{u \sim \mathrm{Uniform}(0,1)} \left[\mathcal{L}_{\text{task}}(\mathbf{y}, \hat{\mathbf{y}}; u) \right], \tag{6.1}$$

where $\mathbf{y}$ is the ground-truth label of $\mathbf{x}$, and the utilization u is uniformly drawn from [0, 1]. The loss

$\mathcal{L}_{\text{task}}$ is a task-specific performance measure. We use cross-entropy loss for classification tasks, and smooth L1 loss [39] for bounding box regression in object detection tasks.

6.4.2 Throttleable Neural Networks

6.4.2.1 Throttleable Block

We consider the building block of TNN architectures that we call *throttleable block* (TB). A TB consists of arbitrary number of filters within one convolutional and fully-connected layer, or across several layers with skip-connections. Most CNN architectures can be converted into TNNs by replacing their layers with TBs.

Let $\mathbf{x}$, $\mathbf{y}$ denote the input and output features to the throttleable block t parameterized by a set of transformations $\boldsymbol{\phi} = \{\phi_i \,|\, i = 1, \ldots, D\} \subset \Phi$, where D is the size of the set which is called "cardinality". The transformation ϕ_i could be anything from individual neurons to entire networks, and we focus on an intermediate level of granularity. It can be arbitrary NN modules like convolutional or fully-connected layers, as long as they have the same input space and their outputs can be aggregated appropriately. We define the *gating functions* $\boldsymbol{g} = \{g_i \,|\, i = 1, \ldots, D\}$ to determine whether to execute each transformation. Then the *throttleable block* has the functional form as

$$\begin{aligned} \mathbf{y} &= t_{\boldsymbol{\phi}}(\mathbf{x}, u) \\ &= \texttt{aggr}\left(\boldsymbol{g}(\mathbf{x}, u) \odot \boldsymbol{\phi}(\mathbf{x})\right) \\ &= \texttt{aggr}\left(g_1(\mathbf{x}, u) \cdot \phi_1(\mathbf{x}), \ldots, g_D(\mathbf{x}, u) \cdot \phi_D(\mathbf{x})\right), \end{aligned} \tag{6.2}$$

where $g_i(\mathbf{x}, u) : \mathbf{x} \times (0, 1] \mapsto \{0, 1\}$ is the *gating function* generating a scalar of either 0 or 1, $\odot$ denotes element-wise multiplication, and `aggr` is the *aggregation function* that maps the

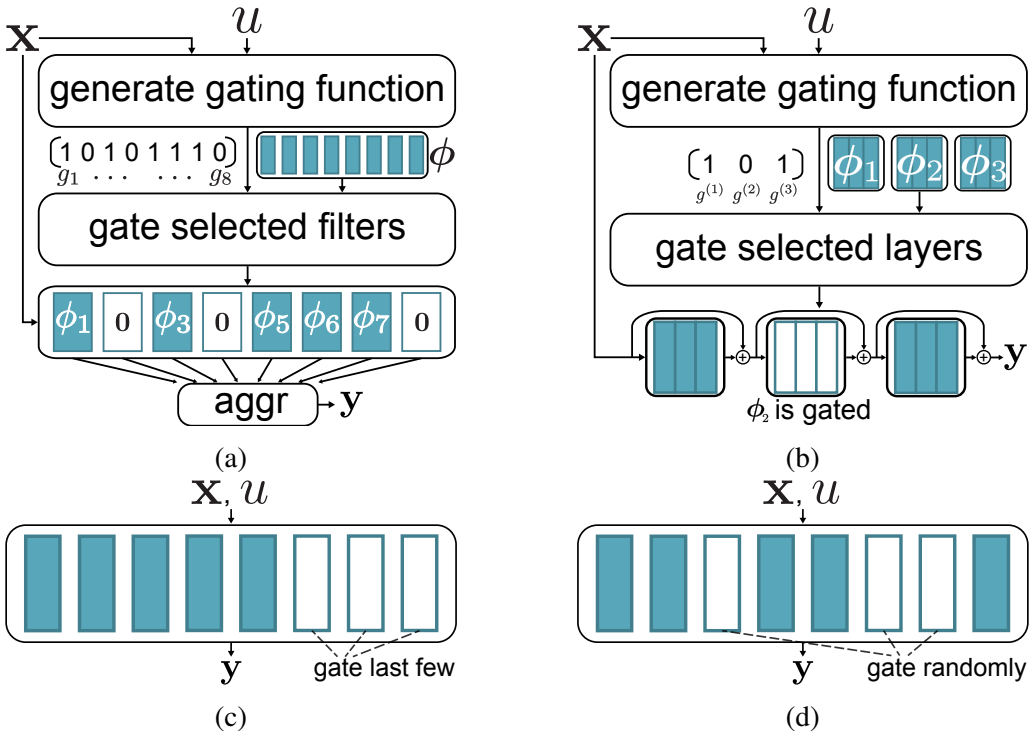


Figure 6.2: Selective gating strategies. The colored blocks are activated groups while white groups are gated. (a) and (b) are gating strategies along different dimensions, while (c) and (d) have different ordering of gating.

set of transformations to the appropriate output space by feature concatenation or summation. The gating function g_i is computed first and determines whether to execute ϕ_i in practice. For example, if $g_i = 0$, the component ϕ_i is effectively disabled and replaced with zero tensors. Our exposition focuses on a single gated module for simplicity, but in practice we compose multiple TBs in a typical TNN. Let $\mathbf{x}^{(l)}$ denote the input feature to the l-th layer, then the output feature $\mathbf{x}^{(l+1)}$ has the functional form as $\mathbf{x}^{(l+1)} = t_{\boldsymbol{\phi}^{(l)}}(\mathbf{x}^{(l)}, u)$, and $\mathbf{x}^{(l+1)}$ is directly fed as input to the next layer.

6.4.2.2 Gating Strategies

For simplicity and comparison, we use convolutional layers for illustrating different gating strategies. Let $\boldsymbol{\phi}$ denote a convolutional layer with C filters. It can be decomposed into D groups of transformations $\{\phi_i \in \mathbb{R}^{C/D \times h \times w} \,|\, i = 1, \ldots, D\}$, each of which consists of at most C/D filters. Each group can be gated selectively. The decomposition and gating is performed along the first dimension (channel) of ϕ_i, and we name this gating strategy as WIDTHWISE gating. Then we apply the non-gated transformations on the input features and perform aggregation following Equation 6.2. An example of WIDTHWISE gating is shown in Figure 6.2a.

Optionally, we can chose to skip the whole layer or block if we have shortcut connections between layers such as in ResNet [2], DenseNet [292], and their variants, then we call this gating strategy as DEPTHWISE gating. Formally, we consider a residual block of the form as $\mathbf{y} = \boldsymbol{\phi}(\mathbf{x}) + \mathbf{x}$ where $\mathbf{x}$ and $\mathbf{y}$ have the same shape, and a DEPTHWISE throttleable block can be represented as

$$\begin{aligned} \mathbf{y} &= t_{\boldsymbol{\phi}}(\mathbf{x}, u) + \mathbf{x} \\ &= \boldsymbol{\phi}(\mathbf{x}) \cdot g(\mathbf{x}, u) + \mathbf{x} \\ &= \begin{cases} \boldsymbol{\phi}(\mathbf{x}) + \mathbf{x} & \text{if } g(\mathbf{x}, u) = 1 \\ \mathbf{x} & \text{otherwise.} \end{cases} \end{aligned} \tag{6.3}$$

In DEPTHWISE gating, $\boldsymbol{\phi}$ can be any block that consists of either a single layer or several layers, and the gating function g is applied to the whole block. An example of DEPTHWISE gating is shown in Figure 6.2b. It is worth noting that WIDTHWISE and DEPTHWISE gating can also be used concurrently within the same network.

6.4.2.3 Mapping Utilization to Gating Functions

For each throttleable block, the mapping $g(\mathbf{x}, u)$ from the utilization to binary gating vectors needs to be defined. We consider INDEPENDENT and NESTED gating strategies to determine which transformations should be gated off, or the order of gating. In previous conditional computation research, the components of each gated module are viewed as independent of one another with few constraints on their pattern of activation. This INDEPENDENT gating strategy (Figure 6.2d) works for each component to model different features, such as in [261, 285]. For our goal of throttling over a range of set points, however, this specialization produces redundancies in the representation. We propose a different method that we call NESTED gating. In the NESTED strategy, the gating function $\boldsymbol{g}$ is constrained such that $g_i = 1 \Rightarrow g_j = 1 \; \forall j < i$ (Figure 6.2c). In our experiments, we employ a training scheme designed to maximize the useful range of u. For each training sample, we draw $u \sim \text{Uniform}[0, 1]$. Then, for each throttleable block, we select d group of transformations to be activated, where $d = \texttt{min}(D, \lfloor u \cdot (D + 1) \rfloor)$ and D is the total number of groups in the block (cardinality). For NESTED gating strategy, we set $g_1, \ldots, g_d$ to 1 and $g_{d+1}, \ldots, g_D$ to 0, while for INDEPENDENT gating strategy we choose d indices at random without replacement. Empirically, we observe that the NESTED strategy gives superior throttling performance given the same architecture.

6.4.2.4 Learnable Gating Function

Equation 6.1 only considers the task performance under all the utilization settings. Optionally, we can add model complexity constraints while maximizing the task performance, and the

overall loss is then defined as

$$\mathcal{L}(\mathbf{x}, u, \mathbf{y}, \hat{\mathbf{y}}) = \mathcal{L}_{\text{task}}(\mathbf{y}, \hat{\mathbf{y}}; u) + \lambda \mathcal{C}(\mathbf{x}, u), \tag{6.4}$$

where $\mathcal{C}$ is a function that measures the resources used (FLOPs, energy, latency, *etc*.) for data $\mathbf{x}$ at utilization u, and λ controls the balance of the two components in the loss. The utilization u is used as the ground-truth for computing the complexity. We enforce the constraint that the actual complexity of the TNN should not exceed the target complexity u by optimizing the combined loss function (Equation 6.4). We experimented with variants of $\mathcal{C}$ of the two functional forms, namely the hinge penalty

$$\mathcal{C}^p_{\text{hinge}}(\mathbf{x}, u) \stackrel{\text{def}}{=} \max(0, c(\Phi; \mathbf{x}, u) - u)^p, \tag{6.5}$$

and the distance penalty

$$\mathcal{C}^p_{\text{dist}}(\mathbf{x}, u) \stackrel{\text{def}}{=} |c(\Phi; \mathbf{x}, u) - u|^p, \tag{6.6}$$

where $p \in \{1, 2\}$, and $c(\Phi; \mathbf{x}, u)$ is the complexity of a TNN. In practice, we found that using the distance penalty with $p = 1$ achieves a higher accuracy for the same resource constraint. The complexity of a TNN is defined as the macro-average of the complexities over all modules:

$$c(\Phi; \mathbf{x}, u) = \frac{1}{L} \sum_l c(\boldsymbol{\phi}^{(l)}; \mathbf{x}^{(l)}, u), \tag{6.7}$$

where $c(\boldsymbol{\phi}^{(l)}; \mathbf{x}^{(l)}, u)$ denotes the complexity of a block, L is the total number of blocks in a TNN, and we set $c(\boldsymbol{\phi}^{(l)}) = 1$ for non-throttleable blocks. A natural measure of the complexity of a TB is its cardinality of active transformations which is defined as

$$c(\boldsymbol{\phi}; \mathbf{x}, u) = \|\boldsymbol{\phi}\|_1^{-1} \sum_i \mathbf{1}\left(g_i(\mathbf{x}, u) = 1\right) \cdot \|\phi_i\|_1, \tag{6.8}$$

where $\|\boldsymbol{\phi}\|_1^{-1}$ is the total number of transformations in a TB, $\mathbf{1}(\cdot)$ is the indicator function, and $\|\phi_i\|_1$ is the number of transformations in a group of the TB.

Learning the gate controller is complicated by the "rich get richer" interaction between $\boldsymbol{g}$ and $\boldsymbol{\phi}$, in which only the subset of $\boldsymbol{\phi}$ selected by $\boldsymbol{g}$ receives training, which improves its performance and reinforces the tendency of $\boldsymbol{g}$ to select it. To address this, we adopt a two-phase training strategy similar to [281]. In the first phase, we train the "data path" with random utilization to optimize only Equation 6.1 for being compatible with throttling. In the second phase, we train the gate to optimize the full objective (Equation 6.4) while keeping the data path fixed.

6.4.2.5 Training the Data Path

The data path is referred to as the TNN for specific tasks without any form of learnable gating functions or controllers. For classification problem, the data path is the network consisting of the feature extractor and multi-class classifier. During phase-1 of training, we train the feature representations of the TNN to be robust to varying amounts of gating. Therefore, the utilization is randomly sampled from a uniform distribution (Equation 6.1). The choice of how u is sampled during training is important for obtaining the desired performance profile. From an empirical risk minimization perspective, we can interpret the training-time distribution of u as a prior distribution on the values of u that we expect at test-time. Naïve training without gating can be viewed as one extreme, where we always set $u = 1$. Either NESTED or INDEPENDENT gating function can be used in phase-1 training.

6.4.2.6 Training the Gate

Besides the fixed gating functions NESTED and INDEPENDENT, we propose two learnable gating functions called CONCRETE and REINFORCE. Our objective is to learn all the gating functions $\mathrm{G} = (\boldsymbol{g}^{(1)}, \cdots, \boldsymbol{g}^{(L)})$ during the phase-2, where we freeze the data path parameters Φ and

optimize the gate function parameterized by Ψ. To make the output of a gating function g_i either 0 or 1, it is modeled as a Bernoulli random variable parameterized by ψ_i:

$$g_i(\mathbf{x}, u; \psi_i) \sim \text{Bernoulli}(\mathrm{P}(\mathbf{x}, u; \psi_i)), \tag{6.9}$$

and our task is to learn the parameters ψ_i for minimizing $\mathcal{C}(\mathbf{x}, u)$ while maintaining the task performance. It is worth noting that the utilization for each TB could vary and the average utilization reaches u approximately. We minimize the difference of the actual utilization and the target utilization u for each TB instead of enforcing an exact utilization value. Since the complexity estimate $\mathcal{C}$ is discontinuous and non-differentiable, we need to employ a gradient estimator for training. We evaluated two existing methods of training networks with stochastic discrete neurons for this purpose.

Score function estimator: The most common approach [261, 279, 293, 294] is to treat $\boldsymbol{g}$ as the output of a stochastic policy and train it with a policy gradient method such as the score function (REINFORCE) estimator,

$$\nabla_\Psi \mathbb{E}[\mathcal{C}] = \mathbb{E}_{\mathbf{x}} \mathbb{E}_u [\mathcal{C} \cdot \nabla_\Psi \log \mathrm{P}(\mathrm{G}(\mathbf{x}, u; \Psi))], \tag{6.10}$$

where $\mathrm{P}(G(\mathbf{x}, u; \Psi))$ is the probability density of random variables G. Since each g_i is an independent Bernoulli random variable (Eqn 6.9), the log probability is given by

$$\begin{aligned} \log \mathrm{P}(\mathrm{G}) &= \sum_l \log \mathrm{P}(\boldsymbol{g}^{(l)}) \\ &= \sum_l \sum_i \log[g_i^{(l)} p_i + (1 - g_i^{(l)})(1 - p_i)], \end{aligned} \tag{6.11}$$

where $p_i = \mathrm{P}(g_i; \mathbf{x}, u, \psi_i)$ is the probability of gating determined by the learnable gating functions.

Continuous relaxations: Relaxation methods soften the discrete gate vector into a continuous vector of "activation strengths". In particular, we use Concrete random variables (CONCRETE)

[295] to stand in for discrete gating during training. Concrete distributions have a temperature parameter τ where the limit $\tau \rightarrow 0$ recovers a corresponding discrete distribution. The Bernoulli distribution is replaced by the binary Concrete distribution,

$$g_i \sim \text{Sigmoid}((Z + \log \frac{p_i}{1 - p_i})^{-\tau}), \tag{6.12}$$

where $Z \sim \text{Logistic}(0, 1)$. We set $\tau = 0.1$ during training to make the network differentiable, and use $\tau = 0$ during testing to recover the desired hard-gated network.

6.4.3 Context-Aware Controller

Trained TNNs take an input and a single control parameter u. However, using a fixed control parameter at run-time may not provide optimal trade-off between performance and utilization. The goal of incorporating the context-aware controller is to adjust the control parameter for TNNs dynamically, and there are many methods to achieve this functionality in a heuristic or learned manner. In this paper, we investigate a trainable input-dependent controller via solving a contextual bandit problem. Such a controller for the hand gesture recognition task is elaborated later, which we believe offers a generalizeable example for a wide variety of vision-based applications.

6.4.3.1 Input-dependent Contextual Bandit

A simplified approach to learn the controller is to frame it as a contextual bandit problem. Hypothetically, for a classification problem, the fact that different inputs or classes require different amount of computational resources is demonstrated in [261, 262]. Based on this assumption, in order to determine what is the best utilization parameter for each input, the prediction of it can be formulated as a sub-problem using contextual bandits. To derive the optimal controller, we

develop the controller network F_{Θ} which is parameterized by weights Θ. The controller outputs the probabilities of choosing a pre-defined set of actions. During training, a reward is given considering the actual FLOPS and confidence of the prediction.

6.4.3.2 State

Given input image or video sequence $\mathbf{x}$, the state representation is simply defined by the input itself as s. The goal of the controller is to receive current input signal and select actions in a way that maximizes rewards for all similar input signals. It will learn appropriate utilization parameters u for different states. The controller network will extract a dense feature representation of input, as well as produce the probabilities for taking each action. The state reflects the contextual information associated with input $\mathbf{x}$.

6.4.3.3 Action

The action is the selection of the utilization parameter u from the action set $\mathcal{U}$. Let $K = |\mathcal{U}|$ denote the size of the action set. The actions resemble the definition of "arms" in multi-armed contextual bandit problems. The learner pulls an arm according to information provided by input $\mathbf{x}$. As a result, the prediction of an action is context-dependent. The size of the action set depends on how u is discretized, namely the step between the adjacent selections of utilization. As long as the number of possible actions is determined, the action set is defined as $\mathcal{U} = \{u_k = \frac{k}{K} \,|\, k = 1, \ldots, K\}$. There can be as many as actions in the set, but K should correlate with the cardinality D of the throttleable blocks. Larger K requires larger D to differentiate TNN sub-models if the differences among actions are small. The context-aware controller is relatively small, it is much more difficult to train with larger K. For the rest of the paper, we simply set $K = 10$ since we have

at most 16 filters in each TB.

6.4.3.4 Reward

Only the throttleable blocks that are not gated off according to u and gating strategy will be evaluated in the forward pass. The actual percentage of computations over the maximum of being full-throttled does not always equal to the utilization. Therefore, the true utilization is measured in the forward pass where the ratio of actual number over the maximum of Multiply–Accumulate Operations (MACs). One can also simply use u as an approximation for simplicity which is shown below. Given the prediction $\hat{\mathbf{y}}$ with confidence c (softmax of the prediction logits) from the TNN, the reward of taking u is defined as:

$$r(u|\mathbf{y}, \hat{\mathbf{y}}, c) = \begin{cases} \exp(1-u) \cdot (1-u) & \text{if } \mathbf{y} = \hat{\mathbf{y}} \\ -(c+\gamma_1) \cdot (u+\gamma_2) & \text{otherwise,} \end{cases} \tag{6.13}$$

where γ_1 and γ_2 are constant parameters for balancing the reward. Larger γ_1 gives more penalty if we use more computations, and larger γ_2 gives more penalty if the network is too confident of a wrong prediction. Their values are empirically selected so that no over-optimization occurs during training, which may make the controller only predict the same utilization value. In experiments, they are empirically set to 0.5 and 1.5 respectively. Such design of reward encourages to achieve higher accuracy while retaining low utilization. And more penalty is given if the prediction is wrong but takes a high utilization parameter.

6.4.3.5 Optimization

Formally, we define the policy of choosing the utilization as $\pi_\Theta(u_k|\mathbf{x}) = [\mathrm{F}_\Theta(\mathbf{x})]_k$, where Θ denotes the learnable weights of the controller network F_Θ, and $[\mathrm{F}_\Theta(\mathbf{x})]_k \in [0, 1]$ is the k-th entry

of the output vector produced by the controller network which represents the probability of choosing the particular utilization u_k. Then the utilization is determined by the optimal action with the highest probability such as

$$u^* = \arg\max_{u_k \in \mathcal{U}} \pi_\Theta(u_k|\mathbf{x}). \tag{6.14}$$

In order to encourage exploration, ϵ-greedy is applied where a random utilization is selected with probability ϵ. The contextual bandit network is trained via a policy gradient approach [296] that maximize the expectation of rewards. The loss function for training the controller is defined as

$$\mathcal{L}_\Theta = -r(u) \log \pi_\Theta(u|\mathbf{x}), \tag{6.15}$$

where the conditions of the reward are omitted for the simplicity of presentation. There are other designs of such a context-aware controller network. For example, we can utilize more complex interactions between the physical environment and the state of our system if we frame the controller as a full Markov Decision Process. In our formulation of the controller, we treat the TNN as a fixed model, but they can also be fine-tuned by end-to-end training.

6.4.3.6 Training and Testing

Training the TNN is a two-phase procedure as discussed in Section 6.4.2.5 and 6.4.2.6. The context-aware controller is considered as a learnable gating function decoupled from the TNN, hence only the phase-1 TNN training (Eqn 6.1) is adopted via curriculum learning [297]. Specifically, the TNN is firstly trained with a low utilization, and then with a higher utilization as training epochs increases. By starting with easier tasks and less learnable parameters, training the TNN is more stable with faster convergence. After training the TNN, we freeze its parameters and train the context-aware controller via Eqn 6.15. The entire procedure for training a DTNN is illustrated in

Algorithm 4.

Algorithm 4 Context-aware TNN training

Input: Training set $\mathbb{X} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$

Output: TNN T_Φ, Controller network F_Θ

Let $u \leftarrow u_0$, step Δu, learning rates α_1, α_2

for iteration $\leftarrow 1, 2, \cdots, \mathcal{N}_{\text{TNN}}$ **do** ▷ TNN training

  $(\mathbf{x}, \mathbf{y}) \leftarrow$ Sample data from $\mathbb{X}$

  $\hat{\mathbf{y}} \leftarrow \mathrm{T}_\Phi(\mathbf{x}, u)$

  $\Phi \leftarrow \Phi - \alpha_1 \nabla \mathcal{L}_{task}(\mathbf{y}, \hat{\mathbf{y}})$

  if reach the iteration of increasing u **then**

    $u \leftarrow u + \Delta u$

  end if

end for

Freeze parameters of T_Φ ▷ Controller training

for iteration $\leftarrow 1, 2, \cdots, \mathcal{N}_{\text{controller}}$ **do**

  $(\mathbf{x}, \mathbf{y}) \leftarrow$ Sample data from $\mathbb{X}$

  $u \leftarrow$ Forward F_Θ via $\epsilon-$greedy

  $\hat{\mathbf{y}}, c \leftarrow$ Forward T_Φ

  $r \leftarrow$ Compute via Equation 6.13

  $\Theta \leftarrow \Theta - \alpha_2 \nabla \mathcal{L}_\Theta$

end for

return T_Φ, F_Θ

One of the advantages of DTNN is that the controller is replaceable after deployment. There are many methodologies to design controllers, such as heuristic rule-based approaches [298, 299], data-driven model-based approaches using reinforcement learning (RL) [259], *etc*. Heuristic rule-based approaches are intuitive, hand-crafted, but not context-aware. Data-driven model-based approaches are capable of making context-aware decisions. For example, the controller is implemented as a much smaller neural network than the data network that takes in the input and predicts a proper control parameter. This is another advantage of a single controllable parameter as it is more feasible to learn.

6.5 Experiments

6.5.1 Experimental Setup

To examine the generality of our TNN concept, we implemented throttleable convolutional and fully-connected layers to directly replace the vanilla layers. Then, we created throttleable variants of several popular CNN architectures. All experiments are implemented with the same default architecture and training hyper-parameters for each dataset unless explicitly mentioned. We use suffix to indicate specific gating strategies such as "-W" for WIDTHWISE, "-D" for DEPTHWISE, "-N" for NESTED and "-WN" for WIDTHWISE NESTED gating, *etc*. All experiments are implemented using PyTorch [111] (versions 0.3.1, 0.4, 1.0 and 1.3), and run on Nvidia GTX 2080 Ti or GPUs. We generate all of the performance curves by evaluating each model on the full test set using fixed values of u. We refer to the vanilla model that do not have gating applied during training as the baseline. Each data point in each chart is the result of a single evaluation run for a single instance of the trained model.

VGG-W: VGG [42] is a typical example of a "single-path" CNN. We apply WIDTHWISE gating with concatenation aggregation to groups of convolutional filters in each layer and combine the outputs by concatenation. Because VGG lacks skip-connections, we enforce that at least one group must be active in each layer, to avoid making the output zero.

ResNet-D: We also experimented with a depthwise-gated version of standard ResNet with summation aggregation, similar to BlockDrop and SkipNet [279, 261]. In this architecture, a throttleable block is converted from an entire ResNet block that can be skipped when gated off.

ResNeXt-W: ResNeXt [43] is a modification of ResNet [2] that structures each ResNet block into groups of filters then aggregates the results by summation. We created a widthwise-gated version of ResNeXt ("ResNeXt-W") by considering each group as a throttleable block.

DenseNet-D/-W: In the DenseNet architecture [292], each dense block contains multiple narrow layers that are combined via concatenation. These narrow layers make natural units for gating. We view this architecture as both widthwise and depthwise gating due to the nature of dense blocks and skip connections.

C3D-W: The throttleable block used in C3D is of the NESTED widthwise gating strategy with concatenation aggregation, which is applied to 3D convolutional and fully-connected layers. We only apply the widthwise gating along the channel dimension, and it is worth noting that it can also be applied along the temporal dimension.

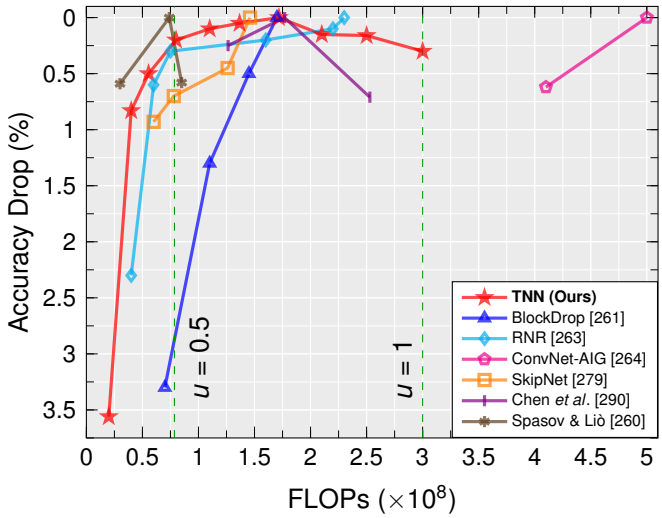


Figure 6.3: Comparisons of relative accuracy drop (%) w.r.t. the peak accuracy on CIFAR-10 for DenseNet-WN with recent dynamic computation methods. Green shaded area denotes the utilization range of [0.5, 1] for the TNN, which has 3×10^8 FLOPs at $u = 1$ and 0.79×10^8 FLOPs at $u = 0.5$.

6.5.2 Image Classification Task

6.5.2.1 CIFAR-10

We evaluate the proposed TNNs on CIFAR-10 [300] and ImageNet [301] datasets following the standard settings of dataset split and pre-processing [2, 42, 300, 301], and report the corresponding top-1 classification accuracy.

The TNNs are based on following architectures: DenseNet-WN, the DenseNet-BC (a compressed DenseNet) with 3 dense blocks with a growth rate $k = 12$, where each dense block is of WIDTHWISE NESTED gating with cardinality $D = 16$; ResNeXt-WN, the ResNeXt-50 architecture as described in [43] with cardinality $D = 16$ in each of the 4 stages; VGG, the VGG-D architecture truncated to the first 3 convolution stages followed by a 4096-dimensional fully-connected layer, where all three convolutional stages and the fully-connected layer are throttleable with cardinality $D = 16$. The learnable gating function (REINFORCE or CONCRETE) is implemented as a Multi-

layer Perceptron (MLP) network (FC $\rightarrow$ ReLU $\rightarrow$ FC) that maps the control input u to gate vectors $\boldsymbol{g}$ for each throttleable block. We show results for the C^2_{dist} complexity penalty (Equation 6.6) where $p = 2$ and $\lambda = 10$.

We compare DenseNet-WN with some state-of-the-art models described in Section 6.2 on CIFAR-10. Due to the diversity of the ideas, implementations, and reported metrics, we only use methods that report computation complexity (*e.g.* FLOPs) and accuracy across multiple operational points for comparison. The FLOPs for other methods are directly obtained from their papers. For every operational point of each method, we compute its FLOPs and accuracy drop between the corresponding peak accuracy, namely the increase of error. It is important to note that utilization controls the ratio of active filters instead of ratio of FLOPs (low-level filters will have more FLOPs). The FLOPs of other methods are obtained from their papers. As shown in Figure 6.3, our throttleable DenseNet achieves the most robust performance across a wide range of utilization (only a few operational points are drawn for clear visualization) . We also have the largest number of operational points that can be adjusted under different computational constraints in a single model without fine-tuning. For all dynamic inference models [260, 261, 262, 263, 264, 279, 290], re-training or fine-tuning is required to obtain each individual model shown in Figure 6.3.

6.5.2.2 ImageNet

Our second set of experiments examines image classification on the 1000-class ImageNet dataset [301] based on DenseNet-169, ResNeXt-50, and ResNet-50 architectures. For ImageNet, we use pre-trained weights to initialize the data path, then fine-tuned the weights with gating. In these

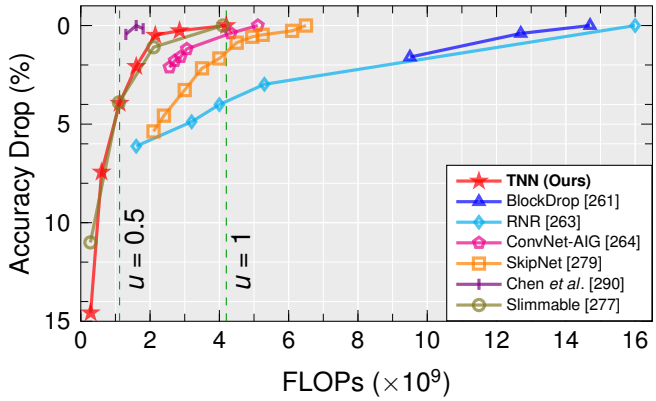


Figure 6.4: Comparisons of relative accuracy drop (%) w.r.t. the peak accuracy on ImageNet for ResNeXt-WN with recent dynamic computation methods. Green shaded area denotes the utilization range of [0.5, 1] for the TNN, which has 4.2 GFLOPs at $u = 1$ and 1.1 GFLOPs at $u = 0.5$.

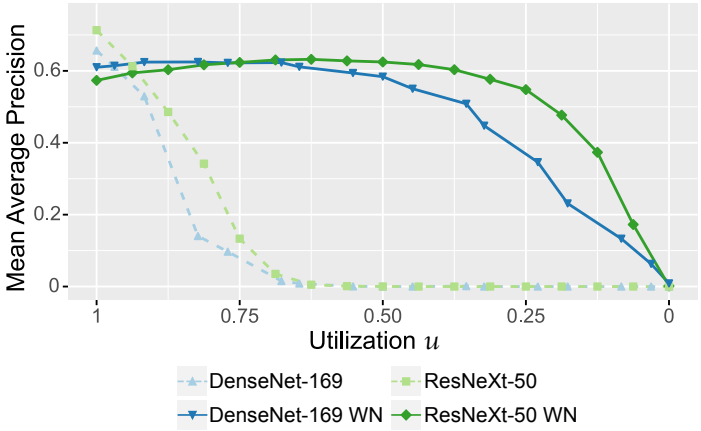


Figure 6.5: TNNs are robust to test-time dropout for object detection on VOC2007 using Faster R-CNN with throttleable "backbones".

experiments, we consider WIDTHWISE and DEPTHWISE NESTED gating. For the DEPTHWISE NESTED strategy, we repeatedly iterate through the stages of the network from output to input and turn on one additional block in each stage, unless the proportion of active residual blocks in that stage exceeds u, and stop when the total utilization exceeds u.

We compare ResNeXt-WN with methods described in Section 6.2 on ImageNet, and perform the same comparison as described in Section 6.5.2.1. As shown in Figure 6.4, our TNN model still achieves the best trad-off between accuracy and efficiency across a wide range of utilization.

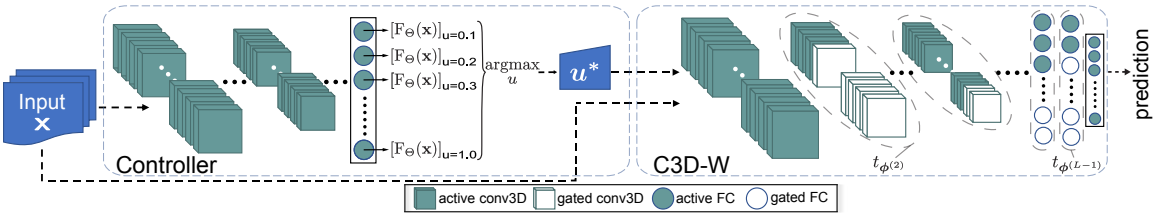


Figure 6.6: A DTNN framework consisting of a light-weight context-aware controller and WIDTH-WISE throttleable 3D convolutional neural network (C3D-W) for video-based hand gesture recognition. The first layer $t_{\phi^{(1)}}$ of C3D-W is non-throttleble.

ResNeXt-WN achieves its peak accuracy of 75.66% at utilization $u = 1$ (4.2G FLOPs), and 71.72% at $u = 0.5$ (1.13G FLOPs). As comparisons, Slimmable [277] achieves similar peak accuracy of 76%, but degenerates more with only four pre-defined operational points; the best pruned model from Chen *et al*. [290] achieves 68.62% accuracy with 1.6G FLOPs, where our model outperforms it by 3.1% while having 0.47G less FLOPs.

6.5.3 Object Detection

We experiment TNNs for object detection task on the PASCAL-VOC2007 dataset [302]. To create a throttleable object detector, Faster R-CNN [19] is adopted without changing its hyper-parameters and the backbone network is converted into a TNN. We use WIDTHWISE NESTED DenseNet-169 and ResNeXt-50 in this experiment. Following the approach of [2] for combining ResNet with Faster R-CNN, we use the TNN as the feature extractor, followed by a region proposal network (RPN) and a detection "head" for object proposal classification and regression. The vanilla models are trained on ImageNet and then fine-tuned on VOC2007. The throttleable models are pre-trained TNNs on ImageNet and fine-tuned on VOC2007 with uniform sampling of u.

We evaluate the above-mentioned models and report the mean average precision (mAP [.5, .95]) in Figure 6.5. Similar to the results on image classification, we observe that the baseline

methods degenerate quickly when any gating is applied and reduce to zero at $u = 0.55$ approximately. For throttleable DenseNet and ResNeXt, the detection performance is well maintained. DenseNet-169-WN achieves 0.61 mAP at $u = 1$ and 0.58 mAP at $u = 0.5$, saving approximately 60% FLOPs with only 0.03 mAP drop. ResNeXt-50-WN achieves even higher mAP in a broader range of utilization (0.6 mAP at $u = 0.375$). Though the TNN have little lower peak MAP, it degrades more gracefully and achieves exceptional trade-offs between performance and computations. TNNs achieve higher tolerance to test-time dropout because of learning robust features under different utilization levels.

6.5.4 Video-based Hand Gesture Recognition

To showcase the TNN architecture for more complex vision applications, we implemented a TNN architecture based on C3D [303] to perform hand gesture classification referred as C3D-W. It takes a fixed number of frames and a control parameter as input and outputs the gesture classification results. The controller is a deep contextual bandit network that follows the design discussed above. The proposed DTNN framework is shown in Figure 6.6.

C3D-WN: We implement a C3D-W architecture with WIDTHWISE NESTED gating strategy. Experiments are performed for the hand gesture recognition task on the 20BN-JESTER dataset [304]. There are a total of 148,092 videos of which 118,562, 14,787 and 14,743 are in training, validation and test set, respectively. Since the ground-truth of test dataset is not publicly available, we use the validation dataset for testing and report the results. The class distribution of the dataset is well balanced except for the class `Doingotherthings` which represents activities other than hand gestures. We resize the input video spatial size (height $\times$ width) to 100×160, and sample 16 continuous frames with random starting index from each video. For C3D-W training, we use four

Nvidia RTX 2080 Ti graphic cards with the batch size of 20 for each GPU. Adam optimizer [305] is applied with an initial learning rate of 5×10^{-5}, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is reduced by 10 with the patience of 3 epochs if the loss is not decreasing. The network is trained for 20 epochs with the curriculum learning schedule where u is set to 0.1, and then increased by 0.1 every two epochs. No further fine-tuning is carried out for training the TNN.

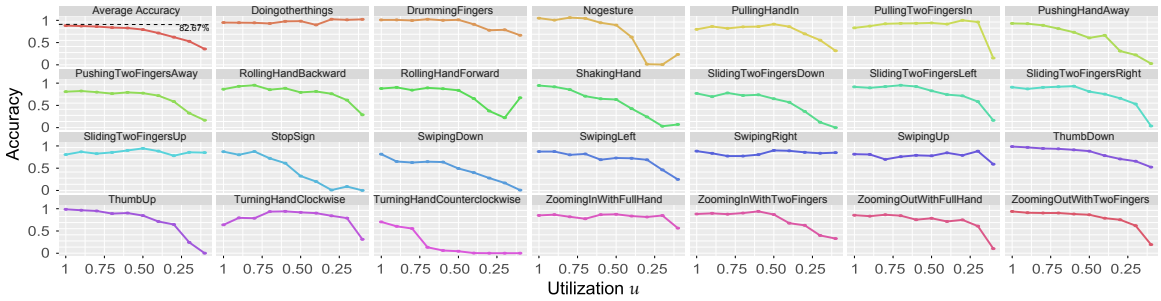


Figure 6.7: Classification accuracy on validation set over utilization parameter u for each gesture class. The top-left facet shows the average accuracy of 81.10% across all classes ,while a vanilla C3D achieves an accuracy of 82.67%.

6.5.4.1 Context-Aware Controller

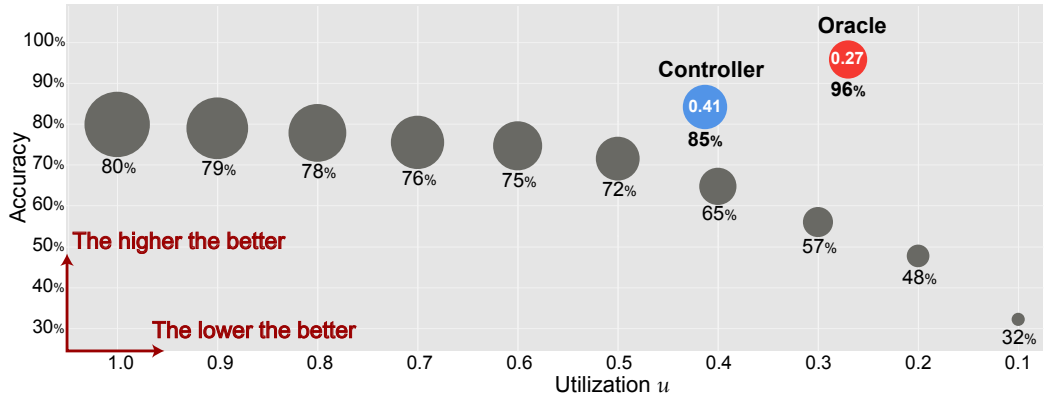
We experiment a 3D version of ShuffleNet architecture [234, 306] as the controller network. The specifications of the DTNN are presented in the Appendix Table 6.4 and 6.5. The design of the controller network is computation-efficient such that the total number of parameters is only 0.955M with 0.255G MACs. Note that the controller network is not a TNN architecture. For controller training, we use a smaller batch size of 16 for handling both the data path and controller network at the same time. We use RMSprop optimizer [307] with a learning rate of 1×10^{-7}. The controller is trained for 10 epochs while freezing the parameters of the data path.

Figure 6.7 shows the various utilization levels for each gesture class. We verify our

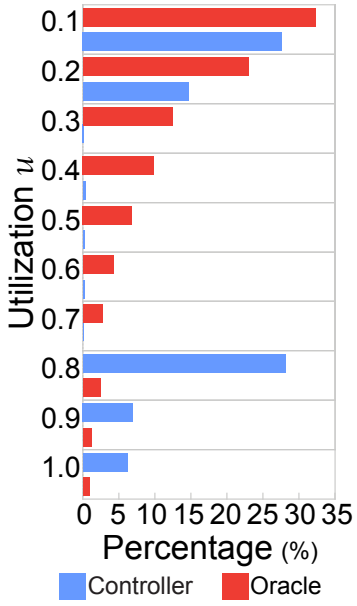
earlier hypothesis (Section 6.4.3.1) that some classes require more resources than others. For instance, the classification performance remains almost constant for each tested utilization parameter u for `Doingotherthings`, while it is linear with u for `Shakinghand`. This demonstrates that we can use less resources (*e.g.*, set $u = 0.1$) and still achieve high classification accuracy for `Doingotherthings`. Practically, DTNN can run with low utilization ("idle") when detecting non-gesture activities, and with higher utilization when detecting gestures ("throttle"). Given this trained TNN, the classification results of each test video for every u parameter are collected. Then a performance oracle is derived by collecting the lowest u where the TNN makes the correct prediction for each testing data. The oracle has an accuracy of 96.14% and an average utilization of 0.27. It is important to note that the performance oracle is just an ideal case for a particular TNN which is impossible to achieve.

The effectiveness of this context-aware controller is confirmed with Figure 6.8a. The learned controller achieves an accuracy of 85.24% with an average utilization of 0.41, which outperforms the vanilla C3D by 2.57%. It manifests that the controller learns the input-specific utilization for each input. In Figure 6.8b, by comparing utilization distributions with the oracle, the controller learns a sparse selection of u where $u = 0.1$, $u = 0.2$ and $u = 0.8$ are chosen more often. Although the controller has 0.14 more utilization on average and 10.90% accuracy drop compared to the oracle, it outperforms all the fixed utilization configurations as well as the vanilla C3D. It further demonstrates how the system can operate within different utilization levels with a learned control policies. Regarding the time of processing a 16-frame clip, we compute the number of parameters, MACs and inference time (clip per millisecond) as shown in Table 6.6. The controller adds a very little computation cost (0.52% more MACs), but reduces much more overall computations by

dynamically predicting per-input utilization for the TNN.



(a) Utilization s accuracy for the learned controller, performance oracle, and fixed u configurations. The area of a circle denotes the average utilization, and the accuracy for each configuration is shown below the circle.



(b) A comparison of utilization distributions.

Figure 6.8: Hand gesture recognition results on 20BN-Jester validation set (as test set). With the context-aware controller, DTNN achieves the best accuracy-computation trade-off comparing to TNN with fixed utilization.

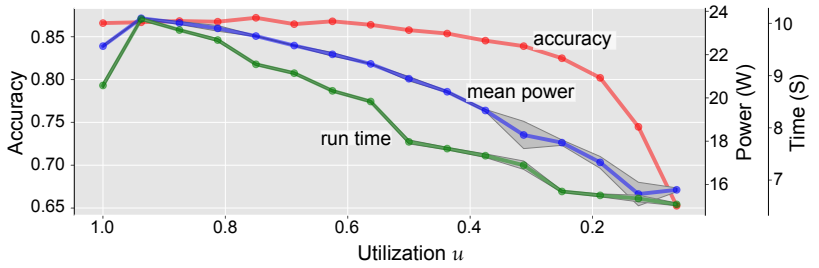


Figure 6.9: Measured throttling performance on NVIDIA Jetson AGX Xavier.

Algorithm 5 WIDTHWISE NESTED Conv2D

Function: Conv2D-WN($\mathbf{x}$, u, $\mathbf{W}$, $\mathbf{b}$)

Let $C \leftarrow$ OUT_CHANNELS($\mathbf{W}$)

Let $n \leftarrow \lceil u \cdot C \rceil$

$\hat{\mathbf{W}}, \hat{\mathbf{b}} \leftarrow \mathbf{W}[0:n], \mathbf{b}[0:n]$ ▷ Slice parameters

$\hat{\mathbf{y}} \leftarrow$ CONV2D($\mathbf{x}$, $\hat{\mathbf{W}}$, $\hat{\mathbf{b}}$)

$\mathbf{z} \leftarrow$ ZEROS($n - C$, size($\hat{\mathbf{y}}$))

$\mathbf{y} \leftarrow$ CONCATENATE($\hat{\mathbf{y}}$, $\mathbf{z}$) ▷ Pad with zeros

return $\mathbf{y}$

6.5.5 Hardware Implementation

We examine the performance of TNN on embedded systems. We implement and measure the computational benefits of TNNs on a NVIDIA Jetson AGX Xavier processor [308], using a throttleable VGG-WN model trained on the CIFAR-10 dataset. Relative savings in run-time and power rather than actual peak values are shown because the results highly depend on the provided hardware mechanisms. We replace the convolution operations in VGG with throttleable 2D convolution by tensor slicing described in Algorithm 5 where $\mathbf{W}$ and $\mathbf{b}$ are the weights and bias for the convolution kernel. To perform a gated convolution, we form truncated weights and bias tensors by

removing channels corresponding to nonactive groups, then perform an ordinary convolution with the truncated weights, and finally restore the output to its original shape by padding with zeros. In the case of NESTED gating, the implementation is highly efficient, requiring only two tensor slice operations and one concatenation.

Figure 6.9 shows accuracy, mean power draw, and run time for classifying the entire CIFAR-10 test set on the Xavier GPU in "MAXN" power mode. The shaded regions show standard deviation of 3 experiments. Salient aspects of this result is the linear ramp-down of the power and run-time while accuracy remains high at lower utilization settings. Compared to no throttling where $u = 1$, the TNN at $u = 0.1$ uses 70% of the power, 74% of the run time, and 52% of the total energy (103J $\boldsymbol{s}$ 197J). We anticipate similar gains in TNN inference efficiencies for other TNN models described in this paper. The key is having an effective control-policy to guide the utilization settings to match application needs.

6.5.6 Analysis

TNNs enable a general framework for conditional computation, whereby the overall computational load and model accuracy can be determined dynamically at inference time. In this section, we offer in depth analysis of our results to best evaluate the proposed architecture and methodology.

6.5.6.1 Ablation Study on Gating Strategies

The experimental results on CIFAR-10 are shown in Figure 6.10. The most noticeable result is that all TNNs are much more robust to various utilization configurations, while the naïve models degrade dramatically (less than 50% accuracy when applying 25% dropout). By comparing the three architectures, DenseNet-DW achieves the best peak accuracy of 91.19% at $u = 0.8125$,

and maintains the accuracy over 91% in the utilization range of [0.5, 1]. The other two TNN architectures also demonstrate strong and consistent performance in the same utilization range.

Among all gating strategies, NESTED gating substantially outperforms all variations over INDEPENDENT for all 3 architectures. The difference is especially pronounced for VGG, and we attribute this to that VGG or similar architectures learn more "entangled" representations than architectures with skip connections, which could make it more sensitive to exactly which transformations are gated off. For depth-wise gating, the performance difference between applying NESTED and INDEPENDENT is smaller than it is for width-wise. This observation indicates that there are more dependencies among features when using width-wise gating. Depth-wise gating is more tolerant of losing features when applying INDEPENDENT due to the short connection between adjacent TBs.

Among models with INDEPENDENT gating strategy, learnable gating models (REINFORCE [309] and CONCRETE [295]) are consistently better than random gating. By training with INDEPENDENT gating, the TNN is robust to different levels of utilization. With learnable gating functions, the learned gating pattern achieves better performance by allocating computation non-uniformly across different stages of the network (shown in Figure 6.11). Blocks in the later stages (higher-numbered) are used preferentially over components in earlier stages. Note that the learned gating functions do not cover the entire range of possible utilization $[0, 1]$. The useful range of u is larger for higher λ and for complexity penalties with $p = 1$. We observe that learnable gating functions do not outperform fixed gating since it is hard to train with very few learnable parameters using MLPs. Thus, we derive the context-aware controller for improving the results by decoupling the control from TNNs.

The ImageNet classification results are shown in Figure 6.12. All TNNs are smoothly

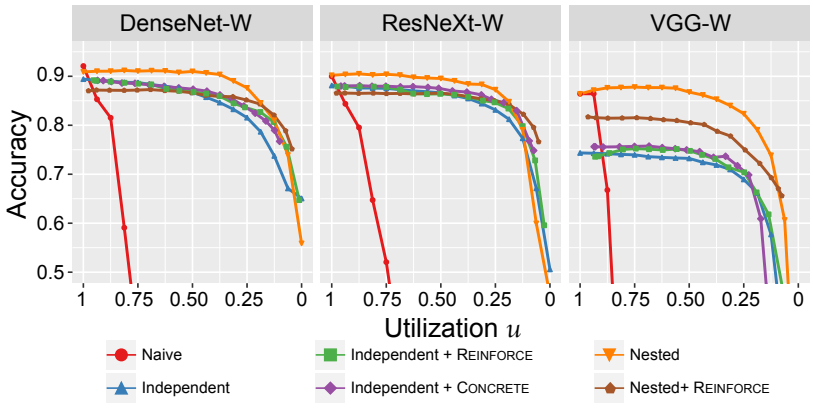


Figure 6.10: Comparison of classification accuracy with different gate control methods for three standard CNN architectures on the CIFAR-10 dataset.

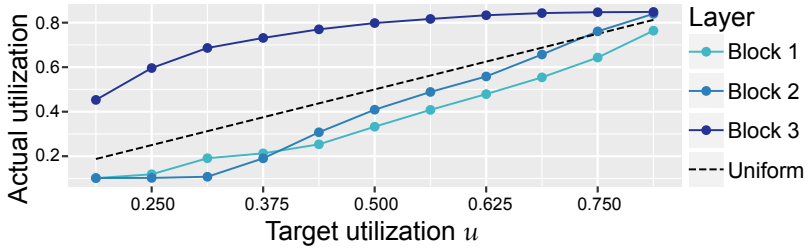


Figure 6.11: The learned gating pattern for selected blocks of DenseNet-DW on CIFAR-10 with the REINFORCE training. The dotted line shows uniform utilization.

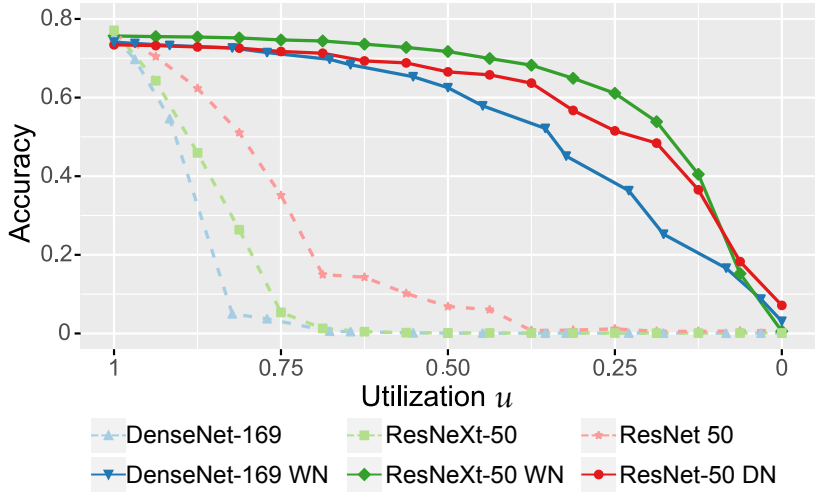


Figure 6.12: Comparisons of results between throttleable and vanilla architectures for image classification on ImageNet-1K.

throttleable through the full range of utilization whereas the pre-trained models degrade rapidly with increased throttling. The ResNeXt-50 model is the best in terms of both peak accuracy (75.66% at $u = 1$) and area-under-curve. It maintains at least 71% accuracy in the utilization range of [0.5, 1].

6.5.6.2 Full-throttle TNN vs. Vanilla Architecture

Table 6.2: Comparisons of accuracy (%) on CIFAR-10 between full-throttle TNNs and vanilla architectures.

	Vanilla	Vanilla u=0.5	TNN u=0.5	TNN u=1.0	Peak	u
DenseNet-W	92.09	0.10	90.99	90.89	91.19	0.81
	(0.00)	(91.99)	(1.10)	(1.20)	(0.90)	
ResNeXt-W	89.99	0.10	89.55	90.20	90.50	0.88
	(0.00)	(89.89)	(0.44)	(0.21)	(0.51)	
VGG-W	86.43	0.09	86.77	86.39	87.79	0.75
	(0.00)	(86.34)	(0.34)	(0.04)	(1.37)	

For each architecture, the second row shows the relative change (increase or decrease) compared to the vanilla baseline. The peak performance of a TNN could be achieved with a lower utilization.

To evaluate and emphasize the effectiveness of TNNs, we summarize and show the results comparisons on CIFAR-10 between the TNNs and the corresponding vanilla architectures in Table 6.2. We consider the vanilla architectures as the baselines, and all the throttleable variants are applied with NESTED gating strategy without learnable gates. Applying a 50% dropout on the

vanilla architecture (Vanilla $u = 0.5$) will result in catastrophic accuracy drop. Instead, TNNs at $u = 0.5$ only have a relative decrease within 1.1%. As for the full-throttle models, we observe an increase of 0.21% for ResNeXt-W. Remarkably, TNNs can achieve the peak performance at a lower utilization instead of full-throttling. The peak accuracy of TNNs is competitive or even superior to the baselines'. For example, the accuracy on CIFAR-10 for VGG-W at $u = 0.75$ has 1.36% accuracy improvement, and for ResNeXt-W at $u = 0.88$ improves by 0.51%. The trivial performance difference between the full-throttle TNN and baseline reveals that most CNN architectures can be converted into throttleble ones while maintaining the peak performance.

It can be observed that generic TNN architectures achieve better performance. VGG consists of several basic convolutional layers; ResNeXt consists of groups of convolutional layers; and DenseNet consists of convolutional layers that are connected with previous layers. Without more complex designs of gating strategies, we expect that VGG and ResNeXt perform better than DenseNet as shown in Table 6.2. It is worth noting that no fine-tuning or data augmentation is applied, and we can always fine-tune a trained TNN at any level of utilization. More importantly, the task performance can be retained or even improved with lower utilization as demonstrated in Table 6.2.

6.5.6.3 Controller Efficacy

For experiments discussed in Section 6.5.4. The controller network only has 3% parameters and 0.05% MACs compared with the TNN. The detailed architectures and their computational costs are shown in the Appendix 6.6.2.

6.5.6.4 Advantages and Potential Applications

One important benefit of the proposed DTNN is to leverage different groups of features in a single architecture for robust prediction, which is controlled dynamically by a single utilization parameter. Decoupling DTNN into two modules alleviates the training difficulty, and enables more flexible architectures and applications. In a TNN, how much to throttle and how to throttle are also disentangled while existing approaches [259, 261, 262, 263, 279] focus on more complex methods that are not practical in real-world deployment.

Having a single control signal u also allows us to enable dynamic throttling to additional constraints beyond those are presented during training. For example, a deployed application may run differently based on environmental conditions (such as battery charge, illumination levels, temperature, *etc*.), as a result, may require alternative operating conditions for the TNN. Because of decoupled design of DTNN, we can still throttle TNN based on application inputs. For example, a system with low battery charge may impose a lower utilization u, and thereby dynamically adjust the quality of services based on system capability. Moreover, the controller behaviour can be changed to any other handcrafted or learnable policies at any time. This modularized design of DTNN offers a user-friendly and domain-agnostic learning system for a wide range of real-world applications such as the presented video-based hand gesture recognition, object detection and tracking, video analytics and monitoring.

6.6 Additional Analysis

6.6.1 FLOPs Calculation

We use the the codes from [310] for computing the FLOPs. The table below summarizes how to compute the FLOPs for some common layers: For computing FLOPs of a convolutional layer in the TNN, we count how many activated kernels within each layer, and the FLOPs are calculated as $2 \times$ No. activated kernels $\times$ kernel shape $\times$ output shape. For fully-connected layers, the input size will change, and the FLOPs are calculated as $2 \times$ activated input size $\times$ output size.

Table 6.3: FLOPs calculation for common layers.

Layer	FLOPs
Convolutional layer	$2 \times$ No. kernels $\times$ kernel shape $\times$ output shape
Fully-connected layer	$2 \times$ input size $\times$ output size
2D Pooling layer	$2 \times$ No. output channels $\times$ output size

6.6.2 Detailed Architectures

Table 6.4 presents the exact specification of the C3D-W used for hand gesture recognition. The conv-block consists of a 3D convolutional layer with ReLU activation. Stride of all convolutional layers is $1 \times 1 \times 1$. Stride of max-pool-1 is $1 \times 2 \times 2$, and other max-pool layers are $2 \times 2 \times 2$. Padding of all conv-blocks is one, and padding of all max-pool layers is zero. In total, C3D-WN has 2.654G FLOPs of non-gated operations, and 94.752G FLOPs of throttleable operations. The detailed architecture of the controller is illustrated in Table 6.5. The network begins with a convolutional layer followed by 16 ShuffleNet units grouped into 3 stages (conv2_x to conv4_x). Each ShuffleNet unit is a residual block where the residual branch consists of one $1 \times 1 \times 1$ group

convolution, channel shuffle operation, $3 \times 3 \times 3$ depthwise convolution [232], and $1 \times 1 \times 1$ group convolution. A cost comparison of the TNN and controller is shown in Table 6.6, indicating that the controller is much more computationally efficient.

6.6.3 Class distribution of 20BN-JESTER

The class distribution of the 20BN-JESTER training set is shown in Figure 6.13.

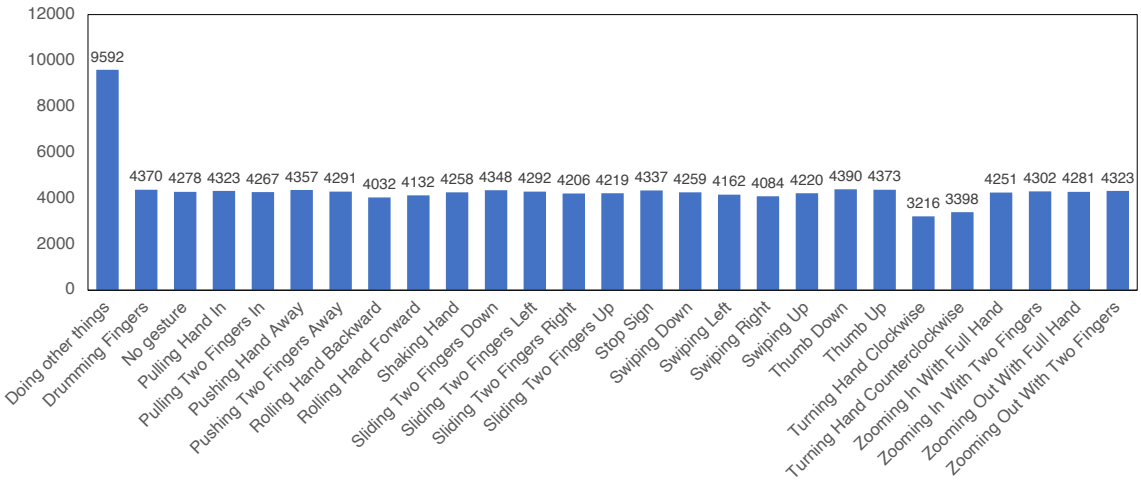


Figure 6.13: Class distribution of 20BN-JESTER training set.

Table 6.4: Detailed architectures of the DTNN for video-based hand gesture recognition on the Jester dataset.

layer	kernel size	C	D	output size
conv1	$3 \times 3 \times 3$	64	1	$16 \times 100 \times 160$
MaxPool	$1 \times 2 \times 2$	64	1	$16 \times 50 \times 80$
conv2	$3 \times 3 \times 3$	128	16	$16 \times 50 \times 80$
MaxPool	$2 \times 2 \times 2$	128	1	$8 \times 25 \times 40$
conv3	$3 \times 3 \times 3$	256	32	$8 \times 25 \times 40$
conv4	$3 \times 3 \times 3$	256	32	$8 \times 25 \times 40$
MaxPool	$2 \times 2 \times 2$	256	1	$4 \times 12 \times 20$
conv5	$3 \times 3 \times 3$	512	32	$4 \times 12 \times 20$
conv6	$3 \times 3 \times 3$	512	32	$4 \times 12 \times 20$
MaxPool	$2 \times 2 \times 2$	512	1	$2 \times 6 \times 10$
conv7	$3 \times 3 \times 3$	512	32	$2 \times 6 \times 10$
conv8	$3 \times 3 \times 3$	512	32	$2 \times 6 \times 10$
MaxPool	$2 \times 2 \times 2$	512	1	$1 \times 3 \times 5$
fc1	-	512	16	$1 \times 1 \times 1$
fc2	-	512	16	$1 \times 1 \times 1$
fc_class	-	27	1	$1 \times 1 \times 1$

Sizes are expressed as *depth* $\times$ *height* $\times$ *width*.

C denotes the output number of channels.

D denotes the cardinality of a TB, and D=1 suggests a non-throttleable layer.

Table 6.5: The contextual controller architecture based on 3D-ShuffleNet.

Stage*	kernel size	stride	repeat	output size
conv1	$3 \times 3 \times 3$	$1 \times 2 \times 2$	1	$24 \times 16 \times 50 \times 80$
MaxPool	$3 \times 3 \times 3$	$2 \times 2 \times 2$	1	$24 \times 8 \times 25 \times 40$
conv2_x	-	$2 \times 2 \times 2$	1	$240 \times 4 \times 13 \times 20$
	-	$1 \times 1 \times 1$	3	
conv3_x	-	$2 \times 2 \times 2$	1	$480 \times 2 \times 7 \times 10$
	-	$1 \times 1 \times 1$	7	
conv4_x	-	$2 \times 2 \times 2$	1	$960 \times 1 \times 4 \times 5$
	-	$1 \times 1 \times 1$	3	
AvgPool	$1 \times 4 \times 5$	$1 \times 1 \times 1$	1	$960 \times 1 \times 1 \times 1$
fc_action	-	-	1	$10 \times 1 \times 1 \times 1$

* each stage from conv2_x to conv4_x consists of one or several ShuffleNet units (the number of repetition is shown in the 4th column).

Table 6.6: Cost comparison between the data path network (C3D-WN) and controller (3D-ShuffleNet).

	# params. (M)	FLOPs (G)	Speed (cpms)
C3D-W	31.865	97.406	137.55
3D-ShuffleNet	0.955	0.510	17.03

Chapter 7

Conclusions

In this work, a pose-guided R-CNN multi-task framework is proposed as an all-in-one solution for person detection, body keypoints prediction and jersey number recognition. It produces the best digit accuracy of 94.09% comparing with related literature. Three insights are used to achieve this performance: 1. re-designed three-class RPN for anchor association; 2. implementation of pose-guided localization network that can impose proposal refinement for jersey number location through human pose; 3. the generality of region-based CNN model. By combining the three components, the proposed approach is end-to-end trainable and can be easily extended to other sports. In this work, a universal jersey number detector (JEDE) was proposed as an end-to-end solution for automated sports analysis that performs player detection, human pose estimation, jersey digit detection, and jersey number detection simultaneously. A dataset was collected that consists of 4477 images from soccer and basketball matches with annotations of 6054 player bounding boxes, 5406 poses, and 9075 digit bounding boxes. Exhaustive evaluations and comparisons were performed on this dataset. By conditioning digit detection on player's features and pose information, JEDE out-

performed the state-of-the-art methods by a large margin. Moreover, to overcome the problem of insufficient data, data augmentation techniques `CopyPasteMix` and `SwapDigit` were proposed that significantly improved the results without extra inference cost. Extensive ablation studies were performed that showed how individual modules, hyper-parameters, and augmentations affect the performance of jersey number detection. Finally, the strong generalization capability of the proposed framework was demonstrated by showing the superior qualitative results across many sport domains.

Scene graph generation is a critical pillar for building machines to visually understand scenes and perform high-level vision and language tasks. In this paper, we introduce a fully convolutional scene graph generation framework that is simple yet effective with fast inference speed. The proposed relation affinity fields serve as a novel representation for visual relationship and produce strong generalizability for unseen relationships. By only using visual features, our exploratory method achieves competitive results over object detection and SGG metrics on the VG dataset. We expect that FCSGG can serve as a general and strong baseline for SGG task, as well as a vital building block extending to down-stream tasks. We have explored novel representations of entities and relationships for scene graph generation, and introduced a performance-guided logit adjustment strategy for long-tailed learning. The proposed RepSGG architecture models entities as subject queries and object keys, and relationships as the attention weights between subjects and objects. The proposed PGLA significantly mitigates the long-tailed problem in SGG. Our experiments demonstrate that RepSGG trained with PGLA compares favourably against box-based, query-based, and point-based SGG models with considerably less design complexity. Our methods also achieve the state-of-the-art performance with fast inference speed. Due to its effectiveness and efficiency, we

envision RepSGG to serve as a strong and simple alternative to current mainstream SGG methods.

In this paper, we presented a novel run-time dynamically *throttleable* neural network (DTNN), as an adaptive model with flexible topology whose performance can be varied dynamically to produce a range of trade-offs between task performance and resource consumption. A DTNN is composed of a throttleable neural network and a contextual controller for dynamically adjusting TNN's inference path. Comprehensive results on image classification and object detection show that TNNs can be effectively throttled across a range of operational points, while having peak accuracy comparable to their vanilla architectures. The experimental results on hand gesture recognition task demonstrate that the proposed DTNN achieves dynamic execution of TNNs with a context-aware controller, outperforming the vanilla architecture and all fixed configurations of utilization.

Bibliography

[1] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *IJCV*, vol. 123, no. 1, pp. 32–73, 2017.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, pp. 770–778, 2016.

[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, pp. 2117–2125, 2017.

[4] S. Gerke, K. Muller, and R. Schafer, "Soccer Jersey Number Recognition using Convolutional Neural Networks," in *ICCVW*, pp. 17–24, 2015.

[5] H. Zhang, Z. Kyaw, S.-F. Chang, and T.-S. Chua, "Visual translation embedding network for visual relation detection," in *CVPR*, pp. 5532–5540, 2017.

[6] Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang, "Scene graph generation from objects, phrases and region captions," in *ICCV*, pp. 1261–1270, 2017.

[7] H. Liu, N. Yan, M. Mortazavi, and B. Bhanu, "Fully convolutional scene graph generation," in *CVPR*, pp. 11546–11556, 2021.

[8] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE TPAMI*, 2020.

[9] G. Adaimi, D. Mizrahi, and A. Alahi, "Composite relationship fields with transformers for scene graph generation," in *WACV*, pp. 52–64, January 2023.

[10] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV*, pp. 10012–10022, 2021.

[11] N. Gkanatsios, V. Pitsikalis, P. Koutras, and P. Maragos, "Attention-translation-relation network for scalable scene graph generation," in *ICCVW*, pp. 0–0, 2019.

[12] H. Liu and B. Bhanu, "Pose-guided R-CNN for Jersey Number Recognition in Sports," in *CVPRW*, pp. 2457–2466, 2019.

[13] H. Liu and B. Bhanu, "Jede: Universal jersey number detector for sports," *IEEE TCSVT*, vol. 32, no. 11, pp. 7894–7909, 2022.

[14] H. Liu and B. Bhanu, "Repsgg: Novel representations of entities and relationships for scene graph generation," *IEEE TPAMI*, 2024.

[15] H. Liu, S. Parajuli, J. Hostetler, S. Chai, and B. Bhanu, "Dynamically throttleable neural networks," *Machine Vision and Applications*, vol. 33, no. 4, p. 59, 2022.

[16] T. Gupta, H. Liu, and B. Bhanu, "Early wildfire smoke detection in videos," in *ICPR*, pp. 8523–8530, IEEE, 2021.

[17] B. X. Guan, B. Bhanu, R. Theagarajan, H. Liu, P. Talbot, and N. Weng, "Human embryonic stem cell classification: random network with autoencoded feature extractor," *Journal of biomedical optics*, vol. 26, no. 5, pp. 052913–052913, 2021.

[18] H. Liu, K. Min, H. A. Valdez, and S. Tripathi, "Contrastive language video time pre-training," *arXiv preprint arXiv:2406.02631*, 2024.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015.

[20] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, pp. 2961–2969, 2017.

[21] M. Šari, H. Dujmi, V. Papi, and N. Roži, "Player number localization and recognition in soccer video using hsv color space and internal contours," in *ICSIP*, 2008.

[22] Q. Ye, Q. Huang, S. Jiang, Y. Liu, and W. Gao, "Jersey number detection in sports video for athlete identification," in *Visual Communications and Image Processing*, vol. 5960, pp. 1599–1606, 2005.

[23] C.-W. Lu, C.-Y. Lin, C.-Y. Hsu, M.-F. Weng, L.-W. Kang, and H.-Y. M. Liao, "Identification and tracking of players in sport videos," in *Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service*, pp. 113–116, ACM, 2013.

[24] G. Li, S. Xu, X. Liu, L. Li, and C. Wang, "Jersey Number Recognition with Semi-Supervised Spatial Transformer Network," in *CVPRW*, pp. 1864–18647, 2018.

[25] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *NeurIPS*, pp. 2017–2025, 2015.

[26] D. Delannay, N. Danhier, and C. D. Vleeschouwer, "Detection and recognition of sports(wo)men from multiple views," in *ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pp. 1–7, Aug 2009.

[27] S. Gerke, A. Linnemann, and K. Müller, "Soccer player recognition using spatial constellation features and jersey number recognition," *Computer Vision and Image Understanding*, vol. 159, pp. 105–115, 2017.

[28] W.-L. Lu, J.-A. Ting, K. P. Murphy, and J. J. Little, "Identifying Players in Broadcast Sports videos using Conditional Random Fields," in *CVPR*, pp. 3249–3256, IEEE, 2011.

[29] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *ICML*, p. 78, ACM, 2004.

[30] W.-L. Lu, J.-A. Ting, J. J. Little, and K. P. Murphy, "Learning to track and identify players from broadcast sports videos," *IEEE TPAMI*, vol. 35, no. 7, pp. 1704–1716, 2013.

[31] A. Senocak, T.-H. O. J. Kim, and I. S. Kweon, "Part-based player identification using deep convolutional representation and multi-scale pooling," in *CVPRW*, pp. 1732–1739, 2018.

[32] J. Poignant, L. Besacier, G. Quenot, and F. Thollard, "From text detection in videos to person identification," in *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo*, pp. 854–859, IEEE Computer Society, 2012.

[33] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," in *ICLR*, 2014.

[34] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *IJCV*, vol. 116, no. 1, pp. 1–20, 2016.

[35] C. Bartz, H. Yang, and C. Meinel, "See: towards semi-supervised end-to-end scene text recognition," in *AAAI*, 2018.

[36] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai, "Robust scene text recognition with automatic rectification," in *CVPR*, pp. 4168–4176, 2016.

[37] M. Busta, L. Neumann, and J. Matas, "Deep textspotter: An end-to-end trainable scene text localization and recognition framework," in *ICCV*, pp. 2204–2212, 2017.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE TPAMI*, vol. 38, no. 1, pp. 142–158, 2016.

[39] R. Girshick, "Fast r-cnn," in *ICCV*, pp. 1440–1448, 2015.

[40] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Transactions on Multimedia*, vol. 20, no. 11, pp. 3111–3122, 2018.

[41] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *CVPR*, pp. 6154–6162, 2018.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[43] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *CVPR*, pp. 1492–1500, 2017.

[44] D. B. West *et al.*, *Introduction to graph theory*, vol. 2. Prentice hall Upper Saddle River, NJ, 1996.

[45] R. Alp Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *CVPR*, pp. 7297–7306, 2018.

[46] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, pp. 7291–7299, 2017.

[47] A. Dutta, A. Gupta, and A. Zissermann, "VGG image annotator (VIA)." `http://www.robots.ox.ac.uk/~vgg/software/via/`, 2016. Version: 2.0.0, Accessed: 7.1.2018.

[48] 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 *European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[49] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow." `https://github.com/matterport/Mask_RCNN`, 2017.

[50] Sportlogiq.

[51] S. Spectrum.

[52] F. Chen and C. De Vleeschouwer, "Formulating Team-Sport Video Summarization as a Resource Allocation Problem," *IEEE TCSVT*, vol. 21, no. 2, pp. 193–205, 2011.

[53] M. Tavassolipour, M. Karimian, and S. Kasaei, "Event Detection and Summarization in Soccer Videos using Bayesian Network and Copula," *IEEE TCSVT*, vol. 24, no. 2, pp. 291–304, 2013.

[54] Z. Wang, J. Yu, and Y. He, "Soccer Video Event Annotation by Synchronization of Attack–Defense Clips and Match Reports with Coarse-Grained Time Information," *IEEE TCSVT*, vol. 27, no. 5, pp. 1104–1117, 2016.

[55] R. Li and B. Bhanu, "Fine-Grained Visual Dribbling Style Analysis for Soccer Videos with Augmented Dribble Energy Image," in *CVPRW*, pp. 2439–2447, 2019.

[56] J. Wang, S. Tan, X. Zhen, S. Xu, F. Zheng, Z. He, and L. Shao, "Deep 3D Human Pose Estimation: A Review," *Comput. Vis. and Image Underst.*, vol. 210, p. 103225, 2021.

[57] Y. Yuan, S.-E. Wei, T. Simon, K. Kitani, and J. Saragih, "SimPoE: Simulated Character Control for 3D Human Pose Estimation," in *CVPR*, pp. 7159–7169, 2021.

[58] L. Jin, C. Xu, X. Wang, Y. Xiao, Y. Guo, X. Nie, and J. Zhao, "Single-Stage Is Enough: Multi-Person Absolute 3D Pose Estimation," in *CVPR*, pp. 13086–13095, 2022.

[59] X. Shu, L. Zhang, G.-J. Qi, W. Liu, and J. Tang, "Spatiotemporal Co-attention Recurrent Neural Networks for Human-skeleton Motion Prediction," *IEEE TPAMI*, 2021.

[60] R. Zhang, X. Shu, R. Yan, J. Zhang, and Y. Song, "Skip-attention Encoder-decoder Framework for Human Motion Prediction," *Multimedia Syst.*, vol. 28, no. 2, pp. 413–422, 2022.

[61] P. Shukla, H. Sadana, A. Bansal, D. Verma, C. Elmadjian, B. Raman, and M. Turk, "Automatic Cricket Highlight Generation using Event-Driven and Excitement-Based Features," in *CVPRW*, pp. 1881–1889, 2018.

[62] R. Theagarajan, F. Pala, X. Zhang, and B. Bhanu, "Soccer: Who Has the Ball? Generating Visual Analytics and Player Statistics," in *CVPRW*, pp. 1830–1838, 2018.

[63] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, *et al.*, "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[64] K. Tuyls, S. Omidshafiei, P. Muller, Z. Wang, J. Connor, D. Hennes, I. Graham, W. Spearman, T. Waskett, D. Steel, *et al.*, "Game Plan: What AI can do for Football, and What Football can do for AI," *Journal of Artif. Intell. Res.*, vol. 71, pp. 41–88, 2021.

[65] H.-C. Shih, "A Survey of Content-aware Video Analysis for Sports," *IEEE TCSVT*, vol. 28, no. 5, pp. 1212–1231, 2017.

[66] M. Beetz, S. Gedikli, J. Bandouch, B. Kirchlechner, N. v. Hoyningen-Huene, and A. Perzylo, "Visually Tracking Football Games Based on TV Broadcasts," in *Proc. of the Int. Joint Conf. on Artif. Intell.*, pp. 2066–2071, 2007.

[67] M. Bertini, A. Del Bimbo, and W. Nunziati, "Player Identification in Soccer Videos," in *Proc. of the ACM SIGMM Int. Workshop on Multimedia Inf. Retr.*, pp. 25–32, 2005.

[68] M. Bertini, A. Del Bimbo, and W. Nunziati, "Matching Faces with Textual Cues in Soccer Videos," in *IEEE Int. Conf. on Multimedia and Expo*, pp. 537–540, IEEE, 2006.

[69] M. Bertini, A. Del Bimbo, and W. Nunziati, "Automatic Detection of Player's Identity in Soccer Videos using Faces and Text Cues," in *Proc. of the ACM Int. Conf. on Multimedia*, pp. 663–666, 2006.

[70] L. Ballan, M. Bertini, A. D. Bimbo, and W. Nunziati, "Soccer players identification based on visual local features," in *Proceedings of the 6th ACM international conference on Image and video retrieval*, pp. 258–265, ACM, 2007.

[71] Z. Mahmood, T. Ali, S. Khattak, L. Hasan, and S. U. Khan, "Automatic Player Detection and Identification for Sports Entertainment Applications," *IEEE TPAMI*, vol. 18, no. 4, pp. 971–982, 2015.

[72] S. Messelodi and C. M. Modena, "Scene Text Recognition and Tracking to Identify Athletes in Sport Videos," *Multimedia Tools and Appl.*, vol. 63, no. 2, pp. 521–545, 2013.

[73] K. Akila, S. Chitrakala, and S. Vaishnavi, "Survey on Illumination Condition of Video/Image under Heterogeneous Environments for Enhancement," in *Int. Conf. on Adv. Comput. and Commun. Syst.*, vol. 1, pp. 1–7, IEEE, 2016.

[74] K. Okuma, D. G. Lowe, and J. J. Little, "Self-Learning for Player Localization in Sports Video," *arXiv preprint arXiv:1307.7198*, 2013.

[75] A. Lehuger, S. Duffner, and C. Garcia, "A Robust Method for Automatic Player Detection in Sport Videos," *Orange Labs*, vol. 4, 2007.

[76] D. Acuna, "Towards Real-Time Detection and Tracking of Basketball Players using Deep Neural Networks," in *NeurIPS*, pp. 4–9, 2017.

[77] J. L. Keyu Lu, Jianhui Chen and H. He, "Light Cascaded Convolutional Neural Networks for Accurate Player Detection," in *BMVC*, pp. 173.1–173.13, BMVA Press, September 2017.

[78] A. Deliege, A. Cioppa, S. Giancola, M. J. Seikavandi, J. V. Dueholm, K. Nasrollahi, B. Ghanem, T. B. Moeslund, and M. Van Droogenbroeck, "SoccerNet-v2: A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos," in *CVPR*, pp. 4508–4519, 2021.

[79] A. Nady and E. E. Hemayed, "Player Identification in Different Sports," in *VISIGRAPP (5: VISAPP)*, pp. 653–660, 2021.

[80] Q. Liang, W. Wu, Y. Yang, R. Zhang, Y. Peng, and M. Xu, "Multi-player Tracking for Multi-view Sports Videos with Improved K-shortest Path Algorithm," *Appl. Sci.*, vol. 10, no. 3, p. 864, 2020.

[81] M. Liao, G. Pang, J. Huang, T. Hassner, and X. Bai, "Mask TextSpotter V3: Segmentation Proposal Network for Robust Scene Text Spotting," in *ECCV*, pp. 706–722, Springer, 2020.

[82] Z. Raisi, M. A. Naiel, G. Younes, S. Wardell, and J. S. Zelek, "Transformer-Based Text Detection in the Wild," in *CVPRW*, pp. 3162–3171, June 2021.

[83] M. Huang, Y. Liu, Z. Peng, C. Liu, D. Lin, S. Zhu, N. Yuan, K. Ding, and L. Jin, "SwinTextSpotter: Scene Text Spotting via Better Synergy between Text Detection and Text Recognition," in *CVPR*, pp. 4593–4603, 2022.

[84] S. Gerke, S. Singh, A. Linnemann, and P. Ndjiki-Nya, "Unsupervised Color Classifier Training for Soccer Player Detection," in *Vis. Commun. and Image Process.*, pp. 1–5, IEEE, 2013.

[85] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Comput. Soc. Conf. on Comput. Vis. and Pattern Recog.*, vol. 1, pp. 886–893, IEEE, 2005.

[86] L. Zhang, Y. Lu, G. Song, and H. Zheng, "RC-CNN: Reverse Connected Convolutional Neural Network for Accurate Player Detection," in *Pacific Rim Int. Conf. on Artif. Intell.*, pp. 438–446, Springer, 2018.

[87] T. Guo, K. Tao, Q. Hu, and Y. Shen, "Detection of Ice Hockey Players and Teams via a Two-Phase Cascaded CNN Model," *IEEE Access*, vol. 8, pp. 195062–195073, 2020.

[88] M. Pobar and M. Ivasic-Kos, "Active Player Detection in Handball Scenes Based on Activity Measures," *Sensors*, vol. 20, no. 5, p. 1475, 2020.

[89] M. Şah and C. Direkoğlu, "Review and Evaluation of Player Detection Methods in Field Sports," *Multimedia Tools and Appl.*, pp. 1–25, 2021.

[90] M. Manafifard, H. Ebadi, and H. A. Moghaddam, "A Survey on Player Tracking in Soccer Videos," *Comput. Vis. and Image Underst.*, vol. 159, pp. 19–46, 2017.

[91] R. Zhang, L. Wu, Y. Yang, W. Wu, Y. Chen, and M. Xu, "Multi-camera Multi-player Tracking with Deep Player Identification in Sports Video," *Pattern Recog.*, vol. 102, p. 107260, 2020.

[92] R. Theagarajan and B. Bhanu, "An Automated System for Generating Tactical Performance Statistics for Individual Soccer Players from Videos," *IEEE TCSVT*, vol. 31, no. 2, pp. 632–646, 2020.

[93] K. Vats, P. Walters, M. Fani, D. A. Clausi, and J. Zelek, "Player Tracking and Identification in Ice Hockey," *arXiv preprint arXiv:2110.03090*, 2021.

[94] S. Baysal and P. Duygulu, "Sentioscope: A Soccer Player Tracking System using Model Field Particles," *IEEE TCSVT*, vol. 26, no. 7, pp. 1350–1362, 2015.

[95] T. Feng, K. Ji, A. Bian, C. Liu, and J. Zhang, "Identifying Players in Broadcast Videos using Graph Convolutional Network," *Pattern Recog.*, p. 108503, 2021.

[96] A. Chan, M. D. Levine, and M. Javan, "Player Identification in Hockey Broadcast Videos," *Expert Syst. with Appl.*, vol. 165, p. 113891, 2021.

[97] D. Delannay, N. Danhier, and C. De Vleeschouwer, "Detection and Recognition of Sports(wo)men from Multiple Views," in *ACM/IEEE Int. Conf. on Distrib. Smart Cameras*, pp. 1–7, IEEE, 2009.

[98] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *CVPR*, pp. 580–587, 2014.

[99] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic Feature Pyramid Networks," in *CVPR*, pp. 6399–6408, 2019.

[100] Wikipedia contributors, "Number (sports)."

[101] Z. Wang and J.-C. Liu, "Translating Math Formula Images to LaTeX Sequences using Deep Neural Networks with Sequence-level Training," *Int. Journal on Doc. Anal. and Recog.*, vol. 24, no. 1, pp. 63–75, 2021.

[102] H. Law and J. Deng, "CornerNet: Detecting Objects as Paired Keypoints," in *ECCV*, pp. 734–750, 2018.

[103] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as Points," *arXiv preprint arXiv:1904.07850*, 2019.

[104] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *ICCV*, pp. 2980–2988, 2017.

[105] D. Dwibedi, I. Misra, and M. Hebert, "Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection," in *ICCV*, pp. 1301–1310, 2017.

[106] N. Dvornik, J. Mairal, and C. Schmid, "Modeling Visual Context is Key to Augmenting Object Detection Datasets," in *ECCV*, pp. 364–380, 2018.

[107] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features," in *ICCV*, pp. 6023–6032, 2019.

[108] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[109] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation," in *CVPR*, pp. 2918–2928, 2021.

[110] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.

[111] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *NeurIPS*, vol. 32, pp. 8026–8037, 2019.

[112] Z. Cai and N. Vasconcelos, "Cascade R-CNN: High Quality Object Detection and Instance Segmentation," *IEEE TPAMI*, 2019.

[113] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-Aware Trident Networks for Object Detection," in *ICCV*, pp. 6054–6063, 2019.

[114] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[115] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[116] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE TPAMI*, vol. 39, no. 11, pp. 2298–2304, 2017.

[117] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What Is Wrong with Scene Text Recognition Model Comparisons? Dataset and Model Analysis," in *ICCV*, pp. 4715–4723, 2019.

[118] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *ICML*, pp. 369–376, 2006.

[119] J. Quiñonero-Candela, M. Sugiyama, N. D. Lawrence, and A. Schwaighofer, *Dataset Shift in Machine Learning*. MIT Press, 2009.

[120] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," *Proc. of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[121] M. Wang and W. Deng, "Deep Visual Domain Adaptation: A Survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[122] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Int. Conf. on Mach. Learn.*, pp. 448–456, PMLR, 2015.

[123] Y. Wu and K. He, "Group Normalization," in *ECCV*, pp. 3–19, 2018.

[124] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[125] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: the Sequential Learning Problem," in *Psychol. of Learn. and Motiv.*, vol. 24, pp. 109–165, Elsevier, 1989.

[126] W. Benjamin and E. Leslie, *On Photography*. Reaktion Books, 2015.

[127] N. Chomsky, *Language and Mind*. Cambridge University Press, 3 ed., 2006.

[128] C. Lewis, *The Poetic Image (Clark Lectures)*. Cambridge, 1946.

[129] G. Ryle, *The Concept of Mind*. U of Chicago Press, 1949.

[130] J. Berger, *Ways of Seeing*. Penguin Modern Classics, Penguin Books Limited, 2008.

[131] S. Sontag, *On Photography*. Kushiel's Legacy, Picador, 2001.

[132] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, "Devise: A deep visual-semantic embedding model," in *NeurIPS*, pp. 2121–2129, 2013.

[133] D. Harwath, W.-N. Hsu, and J. Glass, "Learning hierarchical discrete linguistic units from visually-grounded speech," in *ICLR*, 2020.

[134] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *CVPR*, pp. 3128–3137, 2015.

[135] M. Mortazavi, "Speech-image semantic alignment does not depend on any prior classification tasks," in *Proceedings of InterSpeech*, 2020.

[136] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," in *European conference on computer vision*, pp. 852–869, Springer, 2016.

[137] H. Wu, J. Mao, Y. Zhang, Y. Jiang, L. Li, W. Sun, and W.-Y. Ma, "Unified visual-semantic embeddings: Bridging vision and language with structured meaning representations," in *CVPR*, pp. 6609–6618, 2019.

[138] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *CVPR*, pp. 5410–5419, 2017.

[139] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *CVPR*, pp. 3668–3678, 2015.

[140] X. Yang, K. Tang, H. Zhang, and J. Cai, "Auto-encoding scene graphs for image captioning," in *CVPR*, pp. 10685–10694, 2019.

[141] T. Yao, Y. Pan, Y. Li, and T. Mei, "Exploring visual relationship for image captioning," in *ECCV*, pp. 684–699, 2018.

[142] J. Johnson, A. Gupta, and L. Fei-Fei, "Image generation from scene graphs," in *CVPR*, pp. 1219–1228, 2018.

[143] D. A. Hudson and C. D. Manning, "Gqa: A new dataset for real-world visual reasoning and compositional question answering," in *CVPR*, pp. 6700–6709, 2019.

[144] D. Teney, L. Liu, and A. van Den Hengel, "Graph-structured representations for visual question answering," in *CVPR*, pp. 1–9, 2017.

[145] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *CVPR*, pp. 1316–1324, 2018.

[146] T. Chen, W. Yu, R. Chen, and L. Lin, "Knowledge-embedded routing network for scene graph generation," in *CVPR*, pp. 6163–6171, 2019.

[147] K. Tang, Y. Niu, J. Huang, J. Shi, and H. Zhang, "Unbiased scene graph generation from biased training," in *CVPR*, pp. 3716–3725, 2020.

[148] K. Tang, H. Zhang, B. Wu, W. Luo, and W. Liu, "Learning to compose dynamic tree structures for visual contexts," in *CVPR*, pp. 6619–6628, 2019.

[149] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context," in *CVPR*, pp. 5831–5840, 2018.

[150] R. Yu, A. Li, V. I. Morariu, and L. S. Davis, "Visual relationship detection with internal and external linguistic knowledge distillation," in *ICCV*, pp. 1974–1982, 2017.

[151] A. Zareian, S. Karaman, and S.-F. Chang, "Bridging knowledge graphs to generate scene graphs," *arXiv preprint arXiv:2001.02314*, 2020.

[152] A. Zareian, H. You, Z. Wang, and S.-F. Chang, "Learning visual commonsense for robust scene graph generation," *arXiv preprint arXiv:2006.09623*, 2020.

[153] V. S. Chen, P. Varma, R. Krishna, M. Bernstein, C. Re, and L. Fei-Fei, "Scene graph prediction with limited labels," in *ICCV*, pp. 2580–2590, 2019.

[154] A. Dornadula, A. Narcomey, R. Krishna, M. Bernstein, and F.-F. Li, "Visual relationships as functions: Enabling few-shot scene graph prediction," in *ICCVW*, pp. 0–0, 2019.

[155] N. Gkanatsios, V. Pitsikalis, and P. Maragos, "From saturation to zero-shot visual relationship detection using local context," in *BMVC*, 2020.

[156] Z.-S. Hung, A. Mallya, and S. Lazebnik, "Contextual translation embedding for visual relationship detection and scene graph generation," *IEEE TPAMI*, 2020.

[157] B. Knyazev, H. de Vries, C. Cangea, G. W. Taylor, A. Courville, and E. Belilovsky, "Graph density-aware losses for novel compositions in scene graph generation," *arXiv preprint arXiv:2005.08230*, 2020.

[158] Y. Li, W. Ouyang, B. Zhou, J. Shi, C. Zhang, and X. Wang, "Factorizable net: an efficient subgraph-based framework for scene graph generation," in *ECCV*, pp. 335–351, 2018.

[159] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *ECCV*, pp. 670–685, 2018.

[160] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *ICCV*, pp. 9627–9636, 2019.

[161] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "Reppoints: Point set representation for object detection," in *ICCV*, pp. 9657–9666, 2019.

[162] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," *Syntax, Semantics and Structure in Statistical Translation*, p. 103, 2014.

[163] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[164] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, 2015.

[165] Y. Li, W. Ouyang, X. Wang, and X. Tang, "Vip-cnn: Visual phrase guided convolutional neural network," in *CVPR*, pp. 1347–1356, 2017.

[166] B. Dai, Y. Zhang, and D. Lin, "Detecting visual relationships with deep relational networks," in *CVPR*, pp. 3076–3086, 2017.

[167] S. Woo, D. Kim, D. Cho, and I. S. Kweon, "Linknet: Relational embedding for scene graph," in *NeurIPS*, pp. 560–570, 2018.

[168] W. Wang, R. Wang, S. Shan, and X. Chen, "Exploring context and visual pattern of relationship for scene graph generation," in *CVPR*, pp. 8188–8197, 2019.

[169] N. Gkanatsios, V. Pitsikalis, P. Koutras, and P. Maragos, "Attention-translation-relation network for scalable scene graph generation," in *ICCVW*, Oct 2019.

[170] B. A. Plummer, A. Mallya, C. M. Cervantes, J. Hockenmaier, and S. Lazebnik, "Phrase localization and visual relationship detection with comprehensive image-language cues," in *ICCV*, pp. 1928–1937, 2017.

[171] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[172] J. Gu, H. Zhao, Z. Lin, S. Li, J. Cai, and M. Ling, "Scene graph generation with external knowledge and image reconstruction," in *CVPR*, pp. 1969–1978, 2019.

[173] X. Lin, C. Ding, J. Zeng, and D. Tao, "Gps-net: Graph property sensing network for scene graph generation," in *CVPR*, pp. 3746–3753, 2020.

[174] A. Newell and J. Deng, "Pixels to graphs by associative embedding," in *NeurIPS*, pp. 2171–2180, 2017.

[175] Z. Cao, G. H. Martinez, T. Simon, S.-E. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE TPAMI*, 2019.

[176] L. Chen, H. Zhang, J. Xiao, X. He, S. Pu, and S.-F. Chang, "Counterfactual critic multi-agent training for scene graph generation," in *ICCV*, pp. 4613–4623, 2019.

[177] X. Yang, H. Zhang, and J. Cai, "Shuffle-then-assemble: Learning object-agnostic visual relationship features," in *ECCV*, pp. 36–52, 2018.

[178] Q. Li, F. Xiao, B. Bhanu, B. Sheng, and R. Hong, "Inner knowledge-based img2doc scheme for visual question answering," *ACM TOMM*, vol. 18, no. 3, pp. 1–21, 2022.

[179] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, pp. 483–499, Springer, 2016.

[180] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: A simple and strong anchor-free object detector," *IEEE TPAMI*, vol. 44, no. 4, pp. 1922–1933, 2020.

[181] R. Li, S. Zhang, and X. He, "SGTR: End-to-end scene graph generation with transformer," in *CVPR*, pp. 19486–19496, 2022.

[182] A. Desai, T.-Y. Wu, S. Tripathi, and N. Vasconcelos, "Single-stage visual relationship learning using conditional queries," *NeurIPS*, vol. 35, pp. 13064–13077, 2022.

[183] Y. Cong, M. Y. Yang, and B. Rosenhahn, "RelTR: Relation transformer for scene graph generation," *IEEE TPAMI*, pp. 1–16, 2023.

[184] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *ECCV*, pp. 213–229, Springer, 2020.

[185] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[186] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," in *ICLR*, 2021.

[187] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar, "Long-tail learning via logit adjustment," in *ICLR*, 2021.

[188] Y. Lu, H. Rai, J. Chang, B. Knyazev, G. Yu, S. Shekhar, G. W. Taylor, and M. Volkovs, "Context-aware scene graph generation with seq2seq transformers," in *ICCV*, pp. 15931–15941, 2021.

[189] N. Dhingra, F. Ritter, and A. Kunz, "BGT-Net: Bidirectional GRU transformer network for scene graph generation," in *CVPR*, pp. 2150–2159, 2021.

[190] Q. Dong, Z. Tu, H. Liao, Y. Zhang, V. Mahadevan, and S. Soatto, "Visual relationship detection using part-and-sum transformers with composite queries," in *ICCV*, pp. 3550–3559, 2021.

[191] Y. Teng and L. Wang, "Structured sparse R-CNN for direct scene graph generation," in *CVPR*, pp. 19437–19446, 2022.

[192] J. Chen, A. Agarwal, S. Abdelkarim, D. Zhu, and M. Elhoseiny, "RelTransformer: A transformer-based long-tail visual relationship recognition," in *CVPR*, pp. 19507–19517, 2022.

[193] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *ICCV*, pp. 764–773, 2017.

[194] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng, "Deep long-tailed learning: A survey," *IEEE TPAMI*, 2023.

[195] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "A comprehensive survey of scene graphs: Generation and application," *IEEE TPAMI*, vol. 45, no. 1, pp. 1–26, 2021.

[196] R. Li, S. Zhang, B. Wan, and X. He, "Bipartite graph network with adaptive message passing for unbiased scene graph generation," in *CVPR*, pp. 11109–11119, 2021.

[197] A. Desai, T.-Y. Wu, S. Tripathi, and N. Vasconcelos, "Learning of visual relations: The devil is in the tails," in *ICCV*, pp. 15404–15413, 2021.

[198] L. Li, L. Chen, Y. Huang, Z. Zhang, S. Zhang, and J. Xiao, "The devil is in the labels: Noisy label correction for robust scene graph generation," in *CVPR*, pp. 18869–18878, 2022.

[199] A. Zhang, Y. Yao, Q. Chen, W. Ji, Z. Liu, M. Sun, and T.-S. Chua, "Fine-grained scene graph generation with data transfer," in *ECCV*, pp. 409–424, Springer, 2022.

[200] T.-J. J. Wang, S. Pehlivan, and J. Laaksonen, "Tackling the unannotated: Scene graph generation with bias-reduced models," in *BMVC*, BMVA, 2020.

[201] M.-J. Chiou, H. Ding, H. Yan, C. Wang, R. Zimmermann, and J. Feng, "Recovering the unbiased scene graphs from the biased ones," in *ACM MM*, pp. 1581–1590, 2021.

[202] Y. Guo, L. Gao, X. Wang, Y. Hu, X. Xu, X. Lu, H. T. Shen, and J. Song, "From general to specific: Informative scene graph generation via balance adjustment," in *ICCV*, pp. 16383–16392, 2021.

[203] T. He, L. Gao, J. Song, J. Cai, and Y.-F. Li, "Learning from the scene and borrowing from the rich: tackling the long tail in scene graph generation," in *IJCAI*, pp. 587–593, 2021.

[204] N. Gkanatsios, V. Pitsikalis, and P. Maragos, "From saturation to zero-shot visual relationship detection using local context.," in *BMVC*, 2020.

[205] B. Knyazev, H. de Vries, C. Cangea, G. W. Taylor, A. Courville, and E. Belilovsky, "Graph density-aware losses for novel compositions in scene graph generation," in *BMVC*, 2020.

[206] S. Yan, C. Shen, Z. Jin, J. Huang, R. Jiang, Y. Chen, and X.-S. Hua, "PCPL: Predicate-correlation perception learning for unbiased scene graph generation," in *ACM MM*, pp. 265–273, 2020.

[207] M. Suhail, A. Mittal, B. Siddiquie, C. Broaddus, J. Eledath, G. Medioni, and L. Sigal, "Energy-based learning for scene graph generation," in *CVPR*, pp. 13936–13945, 2021.

[208] W. Li, H. Zhang, Q. Bai, G. Zhao, N. Jiang, and X. Yuan, "PPDL: Predicate probability distribution based loss for unbiased scene graph generation," in *CVPR*, pp. 19447–19456, 2022.

[209] X. Lyu, L. Gao, Y. Guo, Z. Zhao, H. Huang, H. T. Shen, and J. Song, "Fine-grained predicates learning for scene graph generation," in *CVPR*, pp. 19467–19475, 2022.

[210] S. Zhang, Z. Li, S. Yan, X. He, and J. Sun, "Distribution alignment: A unified framework for long-tail visual recognition," in *CVPR*, pp. 2361–2370, June 2021.

[211] H. Wei, R. Xie, H. Cheng, L. Feng, B. An, and Y. Li, "Mitigating neural network overconfidence with logit normalization," in *ICML*, pp. 23631–23644, PMLR, 2022.

[212] C. Chen, Y. Zhan, B. Yu, L. Liu, Y. Luo, and B. Du, "Resistance training using prior bias: toward unbiased scene graph generation," in *AAAI*, vol. 36, pp. 212–220, 2022.

[213] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[214] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[215] D. Abou Chacra and J. Zelek, "The topology and language of relationships in the visual genome dataset," in *CVPRW*, pp. 4859–4867, IEEE, 2022.

[216] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *ICLR*, OpenReview.net, 2017.

[217] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *IJCV*, vol. 128, no. 7, pp. 1956–1981, 2020.

[218] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, pp. 807–814, 2010.

[219] D. Liu, M. Bober, and J. Kittler, "Constrained structure learning for scene graph generation," *IEEE TPAMI*, 2023.

[220] J. Zhang, K. J. Shih, A. Elgammal, A. Tao, and B. Catanzaro, "Graphical contrastive losses for scene graph parsing," in *CVPR*, pp. 11535–11543, 2019.

[221] Z. Tian, H. Chen, X. Wang, Y. Liu, and C. Shen, "AdelaiDet: A toolbox for instance-level recognition tasks." `https://git.io/adelaidet`, 2019.

[222] A. Gupta, P. Dollar, and R. Girshick, "LVIS: A dataset for large vocabulary instance segmentation," in *CVPR*, pp. 5356–5364, 2019.

[223] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2018.

[224] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *JMLR*, vol. 9, no. 11, 2008.

[225] S. Sharifzadeh, S. M. Baharlou, M. Berrendorf, R. Koner, and V. Tresp, "Improving visual relation detection using depth maps," in *ICPR*, pp. 3597–3604, IEEE, 2021.

[226] J. Ji, R. Krishna, L. Fei-Fei, and J. C. Niebles, "Action genome: Actions as compositions of spatio-temporal scene graphs," in *CVPR*, pp. 10236–10247, 2020.

[227] Y.-W. Chao, Y. Liu, X. Liu, H. Zeng, and J. Deng, "Learning to detect human-object interactions," in *WACV*, pp. 381–389, IEEE, 2018.

[228] M.-J. Chiou, C.-Y. Liao, L.-W. Wang, R. Zimmermann, and J. Feng, "ST-HOI: A spatial-temporal baseline for human-object interaction detection in videos," in *ICDARW*, pp. 9–17, 2021.

[229] X. Shang, T. Ren, J. Guo, H. Zhang, and T.-S. Chua, "Video visual relation detection," in *ACM MM*, pp. 1300–1308, 2017.

[230] J. Yang, Y. Z. Ang, Z. Guo, K. Zhou, W. Zhang, and Z. Liu, "Panoptic scene graph generation," in *ECCV*, pp. 178–196, Springer, 2022.

[231] J. Yang, W. Peng, X. Li, Z. Guo, L. Chen, B. Li, Z. Ma, K. Zhou, W. Zhang, C. C. Loy, *et al.*, "Panoptic video scene graph generation," in *CVPR*, pp. 18675–18685, 2023.

[232] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, pp. 1251–1258, 2017.

[233] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, pp. 4510–4520, 2018.

[234] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *CVPR*, pp. 6848–6856, 2018.

[235] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *ECCV*, pp. 116–131, 2018.

[236] L. Yang, Z. Qi, Z. Liu, H. Liu, M. Ling, L. Shi, and X. Liu, "An embedded implementation of cnn-based hand detection and orientation estimation algorithm," *Machine Vision and Applications*, vol. 30, no. 6, pp. 1071–1082, 2019.

[237] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," in *CVPR*, pp. 1580–1589, 2020.

[238] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, pp. 19–34, 2018.

[239] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *ICLR*, 2019.

[240] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *CVPR*, pp. 2820–2828, 2019.

[241] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *ICLR*, 2019.

[242] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *ICLR*, 2020.

[243] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *CVPR*, pp. 9194–9203, 2018.

[244] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *CVPR*, pp. 11264–11272, 2019.

[245] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *CVPR*, pp. 4340–4349, 2019.

[246] H. Wu, Y. Tang, and X. Zhang, "A pruning method based on the measurement of feature extraction ability," *Machine Vision and Applications*, vol. 32, no. 1, pp. 1–11, 2021.

[247] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *ECCV*, pp. 784–800, 2018.

[248] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *ICCV*, pp. 1365–1374, 2019.

[249] T. Li, B. Wu, Y. Yang, Y. Fan, Y. Zhang, and W. Liu, "Compressing convolutional neural networks via factorized convolutional filters," in *CVPR*, pp. 3977–3986, 2019.

[250] Y. Li, S. Gu, C. Mayer, L. V. Gool, and R. Timofte, "Group sparsity: The hinge between filter pruning and decomposition for network compression," in *CVPR*, pp. 8018–8027, 2020.

[251] S. Hashemi, N. Anthony, H. Tann, R. I. Bahar, and S. Reda, "Understanding the impact of precision quantization on the accuracy and energy of neural networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 1474–1479, IEEE, 2017.

[252] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *ICLR*, OpenReview.net, 2017.

[253] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *CVPR*, pp. 2704–2713, 2018.

[254] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *ICCV*, pp. 4852–4861, 2019.

[255] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Hawq: Hessian aware quantization of neural networks with mixed-precision," in *ICCV*, pp. 293–302, 2019.

[256] B. F. Jimmy Ba, "Adaptive dropout for training deep neural networks," in *NeurIPS*, pp. 3084–3092, 2013.

[257] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 1929–1958, 2014.

[258] C. Riquelme, G. Tucker, and J. Snoek, "Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling," in *ICLR*, 2018.

[259] L. Liu and J. Deng, "Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution," in *AAAI*, 2018.

[260] P. S.E.Spasov, "Dynamic neural network channel execution for efficient training," *BMVC*, 2019.

[261] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *CVPR*, pp. 8817–8826, 2018.

[262] Z. Chen, Y. Li, S. Bengio, and S. Si, "You look twice: Gaternet for dynamic filter selection in cnns," in *CVPR*, pp. 9172–9180, 2019.

[263] Y. Rao, J. Lu, J. Lin, and J. Zhou, "Runtime network routing for efficient image classification," *IEEE TPAMI*, vol. 41, no. 10, pp. 2291–2304, 2018.

[264] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *ECCV*, pp. 3–18, 2018.

[265] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE TPAMI*, 2021.

[266] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, pp. 525–542, Springer, 2016.

[267] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, pp. 6105–6114, 2019.

[268] H. Chen, Y. Wang, C. Xu, B. Shi, C. Xu, Q. Tian, and C. Xu, "Addernet: Do we really need multiplications in deep learning?," in *CVPR*, pp. 1468–1477, 2020.

[269] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.

[270] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *CVPR*, pp. 10734–10742, 2019.

[271] P. Nayak, D. Zhang, and S. Chai, "Bit efficient quantization for deep neural networks," *arXiv preprint arXiv:1910.04877*, 2019.

[272] T. Dinh, A. Melnikov, V. Daskalopoulos, and S. Chai, "Subtensor quantization for mobilenets," in *European conference on computer vision Workshops (ECCVW)* (A. Bartoli and A. Fusiello, eds.), vol. 12539 of *Lecture Notes in Computer Science*, pp. 126–130, Springer, 2020.

[273] S. Wiedemann, K.-R. Müller, and W. Samek, "Compact and computationally efficient representation of deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 31, no. 3, pp. 772–785, 2020.

[274] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 29, no. 11, pp. 5784–5789, 2018.

[275] S. Ghamari, K. Ozcan, T. Dinh, A. Melnikov, J. Carvajal, J. Ernst, and S. Chai, "Quantization-guided training for compact tinyml models," *CoRR*, vol. abs/2103.06231, 2021.

[276] C. Ahn, E. Kim, and S. Oh, "Deep elastic networks with model selection for multi-task learning," in *ICCV*, pp. 6529–6538, 2019.

[277] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," in *ICLR*, 2019.

[278] E. Kim, C. Ahn, and S. Oh, "Nestednet: Learning nested sparse structures in deep neural networks," in *CVPR*, pp. 8669–8678, 2018.

[279] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *ECCV*, pp. 409–424, 2018.

[280] Y. Bengio, "Deep learning of representations: Looking forward," in *International Conference on Statistical Language and Speech Processing*, pp. 1–37, Springer, 2013.

[281] M. Figurnov, M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov, "Spatially adaptive computation time for residual networks," in *CVPR*, pp. 1790–1799, IEEE, 2017.

[282] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, IEEE, 2016.

[283] Z. Li, Y. Yang, X. Liu, F. Zhou, S. Wen, and W. Xu, "Dynamic computational time for visual attention," in *ICCVW*, pp. 1199–1209, IEEE, 2017.

[284] A. Ruiz and J. Verbeek, "Adaptative inference cost with convolutional neural mixture models," in *ICCV*, October 2019.

[285] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *ICLR*, 2017.

[286] R. Teja Mullapudi, W. R. Mark, N. Shazeer, and K. Fatahalian, "Hydranets: Specialized dynamic architectures for efficient inference," in *CVPR*, pp. 8080–8089, 2018.

[287] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *ECCV*, 2016.

[288] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *ICML*, pp. 4092–4101, 2018.

[289] X. Gao, Y. Zhao, Łukasz Dudziak, R. Mullins, and C. zhong Xu, "Dynamic channel pruning: Feature boosting and suppression," in *ICLR*, 2019.

[290] Z. Chen, T.-B. Xu, C. Du, C.-L. Liu, and H. He, "Dynamical channel pruning by conditional accuracy change for deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 32, no. 2, pp. 799–813, 2021.

[291] A. Odena, D. Lawson, and C. Olah, "Changing model behavior at test-time using reinforcement learning," in *International Conference on Learning Representations Workshops (ICLRW)*, 2017.

[292] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, pp. 2261–2269, 2017.

[293] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[294] J. Peng and B. Bhanu, "Closed-loop object recognition using reinforcement learning," *IEEE TPAMI*, vol. 20, no. 2, pp. 139–154, 1998.

[295] C. J. Maddison, A. Mnih, and Y. W. Teh, "The Concrete distribution: A continuous relaxation of discrete random variables," in *ICLR*, 2017.

[296] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *NeurIPS*, pp. 1057–1063, 2000.

[297] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, pp. 41–48, 2009.

[298] H. Tann, S. Hashemi, R. Bahar, and S. Reda, "Runtime configurable deep neural networks for energy-accuracy trade-off," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, p. 34, ACM, 2016.

[299] S. Ganapathy, S. Venkataramani, G. Sriraman, B. Ravindran, and A. Raghunathan, "Dyvedeep: Dynamic variable effort deep neural networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 19, no. 3, pp. 1–24, 2020.

[300] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, Department of Computer Science, 2009.

[301] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[302] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results." `http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html`.

[303] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, pp. 4489–4497, 2015.

[304] twentybn, "The 20bn-jester dataset v1." `https://20bn.com/datasets/jester`, 2019. Version: 1.0, Accessed: 8.1.2019.

[305] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[306] O. Kopuklu, N. Kose, A. Gunduz, and G. Rigoll, "Resource efficient 3d convolutional neural networks," in *ICCVW*, pp. 0–0, 2019.

[307] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, 2012.

[308] NVIDIA, "Nvidia jetson agx xavier module," 2019.

[309] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[310] Facebook, "fvcore." `https://github.com/facebookresearch/fvcore/blob/main/fvcore/nn/flop_count.py`, 2019.