Object Detection and Classification based on Hierarchical Semantic Features and Deep Neural Networks

By

Wenchi Ma

Dissertation Submitted to the
Department of Electrical Engineering and Computer Science and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

The Dissertation Committee for Wenchi Ma certifies
that this is the approved version of the following dissertation:

Object Detection and Classification based on Hierarchical Semantic Features and Deep Neural Networks

_____

,

Date approved: July 27, 2021

Acknowledgements

Let me take this opportunity to give my sincere gratitude and appreciation to all the people that kindly support me through my Ph.D study. Especially, I would like to thank my advisor Prof. Guanghui Wang and Prof. Bo Luo. Their enthusiasm and devotion to academic research inspires me, and their valuable advice and insights keeps me moving forward. I also want to thank my committee members, Prof. Prasad Kulkarni, Prof. Taejoon Kim, Prof. Cuncong Zhong, Prof. Haiyang Chao for their suggestion and time. I appreciate all the people in ITTC who provides me the convenience in both techniques and working life. I am also grateful for Prof. Ziming Zhang for his mentorship and problem-solving qualities when we were cooperating and working together.

My sincere thanks are also given to my friends, Yuanwei Wu, Kaidong Li, Wenju Xu, Tianxiao Zhang, Jacob Camenzind, Charlie Carr, Jan Zwank, Bo Jiang and Siyu li for their accompanies and encouragement along this long but interesting and happy journey.

Last but not the least, I would like to express my most sincere thanks and blessing to my parents. Thank you for your selfless love all the way in my life and thank you for your support to my study, work, everything that means to me.

Abstract

The abilities of feature learning, semantic understanding, cognitive reasoning, and model generalization are the consistent pursuit for current deep learning-based computer vision tasks. A variety of network structures and algorithms have been proposed to learn effective features, extract contextual and semantic information, deduct the relationships between objects and scenes, and achieve robust and generalized model. Nevertheless, these challenges are still not well addressed. One issue lies in the inefficient feature learning and propagation, static single-dimension semantic memorizing, leading to the difficulty of handling challenging situations, such as small objects, occlusion, illumination, etc. The other issue is the robustness and generalization, especially when the data source has diversified feature distribution.

The study aims to explore classification and detection models based on hierarchical semantic features ("transverse semantic" and "longitudinal semantic"), network architectures, and regularization algorithm, so that the above issues could be improved or solved. (1) A detector model is proposed to make full use of "transverse semantic", the semantic information in space scene, which emphasizes on the effectiveness of deep features produced in high-level layers for better detection of small and occluded objects. (2) We also explore the anchor-based detector algorithm and propose the location-aware reasoning (LAAR), where both the location and classification confidences is considered for the bounding box quality criterion, so that the best-qualified boxes can be picked up in Non-Maximum Suppression (NMS). (3) A semantic clustering-based deduction learning is proposed, which explores the "longitudinal semantic", realizing the high-level clustering in the semantic space, enabling the model

to deduce the relations among various classes so as better classification performance is expected. (4) We propose the near-orthogonality regularization by introducing an implicit self-regularization to push the mean and variance of filter angles in a network towards 90° and 0° simultaneously, revealing it helps stabilize the training process, speed up convergence and improve robustness. (5) Inspired by the research that self-attention networks possess a strong inductive bias which leads to the loss of feature expression power, the transformer architecture with mitigatory attention mechanism is proposed and applied with the state-of-the-art detectors, verifying the superiority of detection enhancement.

Contents

1 Introduction **1**

1.1 Deep Learning and Vision Application 2

1.1.1 Object Detection 2

1.1.2 Deep Convolutional Neural Network and Regularization 3

1.2 Object Detection with Transformer 6

2 Background and Literature Review **8**

2.1 Deep Neural Networks 8

2.1.1 Deep Forward Networks 9

2.1.1.1 Gradient-based Learning 10

2.1.1.2 Cost Functions 11

2.1.1.3 Back-Propagation 12

2.1.2 Network Learning and Optimization 13

2.2 Convolutional Neural Network (CNN) 14

2.3 Object Detection with Convolutional Neural Network 15

2.4 Feature Expression and Inception Technology 17

2.4.1 Inception Technology 18

2.5 Regularization 19

2.6 Attention Mechanism and Transformers 20

3 Multi-Scale Deep Feature Learning Network for Object Detection 24

3.1 Multi-Scale Deep Feature Learning Network for Object Detection 24

3.1.1 Deep Feature Learning 26

3.1.1.1 Deep Feature Extraction and Analysis 26

3.1.1.2 Deep Feature Learning Inception Modules 29

3.1.1.3 Parameter Sharing 31

3.1.2 Multi-Scale Object Detection Scheme 31

3.1.3 Experimental Evaluations 34

3.1.3.1 Dataset 35

3.1.4 Detection Results on KITTI 36

3.1.5 Detection Results on PASCAL VOC Dataset 38

3.1.6 Detection Results on COCO 39

3.1.7 Efficiency Discussion 40

3.1.8 Limitation 42

3.1.9 Conclusion 43

4 Location-Aware Box Reasoning for Anchor-Based Single-Shot Object Detection 45

4.1 Introduction 45

4.2 Related Work 48

4.2.1 Object Detection 48

4.2.2 Detection Scoring and Correction 50

4.3 Location-Aware Box Reasoning 51

4.3.1 Motivation 51

4.3.2 Location Awareness 52

4.3.3 Localization Score Regression 54

4.3.4 Box Reasoning during Inference 55

4.4 Experiments 56

4.4.1 Implementation Details 57

4.4.2 Ablation Studies . . . 58
4.4.3 Quantitative Results . . . 59
4.4.4 Comparison and Discussion . . . 62
4.4.4.1 Fitter and tighter bounding boxes . . . 62
4.4.4.2 Better for hard objects . . . 62
4.4.4.3 Waiving low-quality boxes . . . 63
4.5 Conclusion . . . 63

5 Semantic Clustering based Deduction Learning for Image Recognition and Classification 64

5.1 Introduction . . . 64
5.2 Related Work . . . 67
5.2.1 Hierarchical Semantic Information . . . 67
5.2.2 Semantic Labeling in Noisy Cases . . . 68
5.3 Problem Setup . . . 69
5.4 Methodology . . . 71
5.4.1 Conventional Classification Learning . . . 71
5.4.2 Learning with Semantic Deduction . . . 72
5.4.3 Equal-Probability Search for Opposite Semantic. . . . 73
5.4.4 Learning with Smooth Semantic Clustering . . . 74
5.4.5 Optimal Learning . . . 76
5.5 Experiment . . . 77
5.5.1 Results in Original Data Environment . . . 79
5.5.2 Results in Noisy Data Environment . . . 83
5.6 Conclusion . . . 85

6 Self-Orthogonality Module:A Network Architecture Plug-in for Learning Orthogonal Filters 86

6.1 Introduction . . . 86
6.1.1 Related Work . . . 89
6.2 Self-Regularization as Internal Orthogonality Regularizer . . . 91
6.2.1 Formulation . . . 92
6.2.2 Implementation . . . 92
6.3 Experiments . . . 94
6.3.1 Multilayer Perceptron (MLP): PointNet . . . 96
6.3.2 Convolutional Neural Networks (CNNs): VoxNet . . . 99
6.4 Conclusion . . . 100

7 Miti-Detr: Object Detection based on Transformers with Mitigatory Self-Attention Convergence 102

7.1 Introduction . . . 103
7.2 Related Work . . . 105
7.2.1 Attention Mechanism . . . 105
7.2.2 Object Detection with Transformer . . . 106
7.3 Miti-DETR . . . 107
7.3.1 Attention Network Loses Rank . . . 107
7.3.2 Skip Connection and MLP Counteract Degeneration . . . 108
7.3.3 Residual Self-attention Network . . . 110
7.3.4 Transformer with Mitigatory Convergence in Object Detection . . . 112
7.4 Experiments . . . 112
7.4.1 Implementation Details . . . 113
7.4.2 Dataset . . . 113
7.4.3 Training and Learning . . . 113
7.4.4 Detection Results . . . 114
7.5 Conclusion . . . 115

8 Conclusion and Future Work **117**

List of Figures

1.1 Examples of object detection 2
1.2 Examples of visual tracking 4

2.1 The architecture of LeNet-5 14
2.2 (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel (source: (158)). 21
2.3 The Transformer - model architecture (source: (158)). 23

3.1 The flow-chart of the proposed fast object localization and tracking strategy. 25
3.2 **Deep feature learning inception modules.** (a) The core and basic deep feature transmission layer structure. (b) and (c) denote the two actual layer structures of information square and cubic inception modules. Red and green arrows indicate the way of parameter sharing. Ψ_{j-1} represents feature maps from previous layer and Ψ_j denotes the output feature maps from current layer. 30
3.3 **The architecture of MDFN.** The proposed deep feature learning inception modules are introduced in the layers of Conv_6, Conv_7, Conv_8 (and Conv_9 in MDFN_I2). Each one employs filters of multiple sizes, which are represented by the red, yellow, blue and green boxes. The white boxes refer to classification and localization regression layers for the jointly learning of multi-scale detection (99). 31

3.4 **Four sets of comparative detection examples of SSD and MDFN models on VOC2007 dataset.** In each set, the three images, from left to right, represent the results from SSD, MDFN-I1 and MDFN-I2, respectively. 39
3.5 The trade-off between the accuracy and speed. 42
3.6 **Comparative detection examples by the two MDFN models.** In each set, the left and right images refer to the detection results of MDFN-I1 and MDFN-I2 respectively. 42
3.7 **Four sets of comparative detection examples of SSD and MDFN on KITTI dataset.** In each set, the four images, from top to bottom, represent the original image and the results from SSD, MDFN-I1 and MDFN-I2 respectively. 44

4.1 Demonstrative detection results on MS COCO (97). The predicted bounding boxes have high classification scores while the localization is misplaced or interceptive. The left two images show misplaced cases in which the zebra is located by a much larger region, and the car is not an actual object. The airplanes in the right two images are partially located, while all of them have high confidence scores. Our method predicts the spatial relation between box proposals and their possible targets so that the interception and misplacement can be minimized. 46

4.2 AIoU Definition and the Locscore illustration. AIoU as the target of the proposed locscore only needs the input image and its corresponding ground truth. In the fashion of convolution, we evaluate the default boxes with multiple scales at different anchor bases, as the yellow and purple color lumps shown in this figure. For each default box (represented by the blue dotted rectangular), in addition to predicting the shape offsets and the confidence scores as conventional detectors, we also predict the locscore which assesses the possibility of how close the object is to the ground truth. The locscore is learned towards the AIoU calculated by the anchor box and the ground truth, which is denoted by the red angular box in this figure. Specifically, the locscore of a box proposal is learned to match the AIoU between its corresponding anchor box and a certain ground truth box. 49

4.3 The network architecture of object detection with location-aware anchor-based reasoning. The input image is fed into the backbone network to generate feature maps with RoI information. The Locscore Branch is the standard component of the improved model. It takes the output features from the backbone network as inputs and provides a predictive locscore at the end, where its layer structure just follows the one of the classification branch or the localization branch. 51

4.4 COCO cases compared between RetinaNet (1st row) and RetineNet+LC (2nd row), where LC version achieves higher localization accuracy. 59

4.5 COCO cases compared between RetinaNet (1st row) and RetineNet+LC (2nd row), where LC version performs better for hard objects. 61

4.6 COCO cases compared between RetinaNet (1st row) and RetinaNet+LC (2nd row), where LC version has better filtering for low-quality boxes. 61

5.1 The deduction progress of semantic clustering. Prior and Guide works as the prior information that is combined with the labels as the labeling input data. The Learning Timeline is the same as the normal classification learning process, but our model provides the possibility of doing high-level semantic clustering by the deduction progress as the aid for the classification task. The model, at the end of the learning timeline, is expected to provide better classification accuracy. 65
5.2 An overview of the proposed method. We use semantic prior based random search to produce opposite semantic so as to form the composite loss function, guiding the model to form semantic colonies. 73
5.3 Convergence performance of different models by training loss on CIFAR10. 80
5.4 Convergence performance comparison by training loss on CIFAR100. 81

6.1 Illustration of the angular distributions of pretrained models with no OR. 87
6.2 An example of integration of our self-regularization as a plug-in on a three hidden-layer DNN as backbone for training. Here "S-Net" denotes a sub-network consisting of non-hidden layers. . . . 93
6.3 Result comparison on ModelNet40: **(left->right)** default setting, $lr = 0.01$, $ts = 440$, and $bs = 2$. . 94
6.4 Result comparison on MNIST: **(left->right)** default setting, $lr = 0.01$, $ts = 200$, and $bs = 2$. 95
6.5 Comparison of the learned angular distributions on ModelNet40: **(left->right)** default setting, $lr = 0.01$, $ts = 440$, and $bs = 2$. 96
6.6 Result comparison on ModelNet10: **(left->right)** default setting, $lr = 0.01$, $ts = 400$, and $bs = 2$. . 98
6.7 Comparison of the learned angular distributions on ModelNet10: **(left->right)** default setting, $lr = 0.01$, $ts = 400$, and $bs = 2$. 99

7.1 Residual Attention Network in Transformer 104
7.2 Architecture Streamline of Miti-DETR 107
7.3 Comparison of the learning process distributions on COCO: **(left->right)** Learning Loss, Test Error, and Average Recall. 114

List of Tables

3.1 Layer structure of deep inception module layout in MDFN-I1 and MDFN-I2 33

3.2 Average precision(%) on KITTI validation set. The best and second best results are highlighted in bold-face and underline fonts, respectively. 36

3.3 Mean average precision on KITTI Car validation set for different IoU thresholds. . 37

3.4 Mean average precision on KITTI Pedestrian validation set for different IoU thresholds. 37

3.5 Mean average precision on KITTI Cyclist validation set for different IoU thresholds. 38

3.6 PASCAL VOC2007 test detection results. 38

3.7 Detection results on COCO test-dev 40

3.8 Comparison of inference time on KITTI, VOC2007 and COCO test datasets. 41

4.1 The mAP of RetinaNet on COCO val2017. R-50 indicates ResNet50 with FPN and R-Net refers to RetinaNet. 59

4.2 mAP of SSD on VOC2017. The category name indicates its corresponding AP result. 60

5.1 Classification accuracy on FashionMNIST. 81

5.2 Classification accuracy on CIFAR10. 81

5.3 Classification accuracy on CIFAR100. 83

5.4 Comparison with Tree-CNN on Cifar10, where SD refers to models that are applied with our proposed algorithm. VGG11 and Tree-CNN are trained by "old" and "new" data in an incremental way (135). 83

5.5 Comparison with Tree-CNN on Cifar100, where Test-Acc stands for the Test Accuracy. SD refers to the corresponding models that are applied with our proposed algorithm. VGG11 and Tree-CNN are trained by "old" and "new" data in an incremental way (135). 83
5.6 Classification on CIFAR10 with noisy labels. 84
5.7 Classification on CIFAR100 with noisy labels. 84

6.1 Best test accuracy comparison on ModelNet40. 94
6.2 Best test accuracy comparison on 2D point clouds of MNIST. 95
6.3 Best test accuracy comparison on ModelNet10. 99
6.4 Test accuracy comparison on CIFAR-10. Here "-" indicates that we cannot make OrthoReg work. . 100

7.1 Detection accuracy on COCO. 115

Chapter 1

Introduction

Recent development of the convolutional neural networks (CNNs) has brought significant progress in computer vision, pattern recognition, and multimedia information processing (131; 16; 28; 161; 183; 48; 63; 146). As an important problem in computer vision, object detection has a large range of applications, such as image retrieval, video surveillance, intelligent medical, and unmanned vehicle (165; 131; 60). Since the breakthrough on image classification achieved by (83) in 2012, deep learning has emerged as a powerful machine learning technique for computer vision tasks. In general, the recent progress is mainly contributed by the efficient convolution network architecture, multi-scaled multi-leveled data and feature representation, and effective learning, regularization algorithms. These three factors play the dominant roles in solving the problems existed in current deep learning based vision projects, especially for object detection and classification. The related techniques have greatly advanced the performance of the state-of-the-art visual recognition systems, such as Faster-RCNN (133), YOLO (127), SSD (99), DenseNet huang2017densely, Feature Pyramid Network (FPN) (95), and DETR (17), etc.

Object detection and classification has been extensively studied in the past decades. Researchers have been working on improving model performance with respect to hard objects, complicated scenes, good generalization, high efficiency and high-order intelligence. In my study, we try to solve the existing object detection and classification problems by hierarchical semantic

features, including "transverse semantic" (spatial semantic features) and "longitudinal semantic" (semantic space hierarchy) so that small scale, occlusion, deformation, viewpoint changes, illumination, intra-classes variability could be better handled. We explore the deduction learning algorithm to guide the network models automatically excavate semantic space structure so as to enable themselves with basic cognitive learning ability. In order to reduce network learning redundancy and enhance model robustness, we work on self-orthogonal regularization based on the angle distribution of filters. Inspired by the attention mechanism and transformer which have the best global feature association, I study the reason that leads to the feature expression loss inside the transformer layer and propose a further improved model. I hope the proposed ideas and the built technologies could be inspiring for further study in the literature.

1.1 Deep Learning and Vision Application

In this thesis, we study a series of problems in deep-learning based vision applications, ranging from object detection, image classification, semantic and context learning, and regularization algorithm. In the following part, related problems will be described. The existing challenges and the proposed approaches will be discussed.

1.1.1 Object Detection

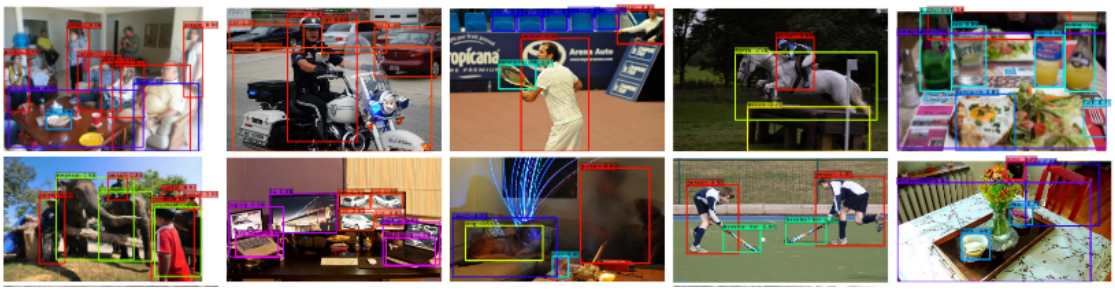

Figure 1.1: Examples of object detection by Single Shot MultiBox Detector (source: (99)).

Object detection in generic images contains recognition and localization towards the objects of

multiple categories in unconstrained situations, which is an open and challenging problem (see examples in Figure 1.1). Precise localization and accurate object description are the ultimate goal for scene understanding that determines **What** the objects are and **Where** they locate. Considering this problem, we encode the classification and localization as two regression progresses, respectively.

The progress of most recent models can be attributed to the powerful feature extraction ability of CNN through building deeper or wider hierarchical structures given the great advancement of computer hardware. The ideas of deep and residual connections (59; 67), network-in-network and inception structures (149), multi-box and multi-scale techniques (99) and dense blocks (66) consistently dedicate to enhancing the feature expression by multi-scale and wide-range feature representation, especially the focus of details from shallow layers. This is beneficial to the precise description towards objects (22) while is struggling for complicated situations with small and occluded objects. Efficient object detection depends not only on detail features, but also heavily on semantic and contextual information, which is able to describe both the relationships among different objects and the correlation between objects and their contexts (22). In my study, we intend to alleviate the above problem by making full use of deep features produced by the deep layers of a network so that the contextual information could be adequately integrated into the learning process through a single-shot framework. The details are discussed in Chapter 3. At the same time, I work on the current deep learning based detection algorithms and find that the quality criterion towards the bounding boxes entirely depends on the confidence of instance classification, regardless of the fact that bounding boxes indicate the spatial relations as well. Given this problem, we propose the location-aware box reasoning technique for anchor-based object detectors, referred in Chapter 4.

1.1.2 Deep Convolutional Neural Network and Regularization

As a fundamental step of many vision and multimedia process tasks, feature extraction and representation has been widely studied (43; 149; 148), especially at the level of network structures, that attracted a lot of attention in the deep learning field. Deep or wide networks amplify the differences among architectures and give full play to improve feature extraction ability in many computer vi-

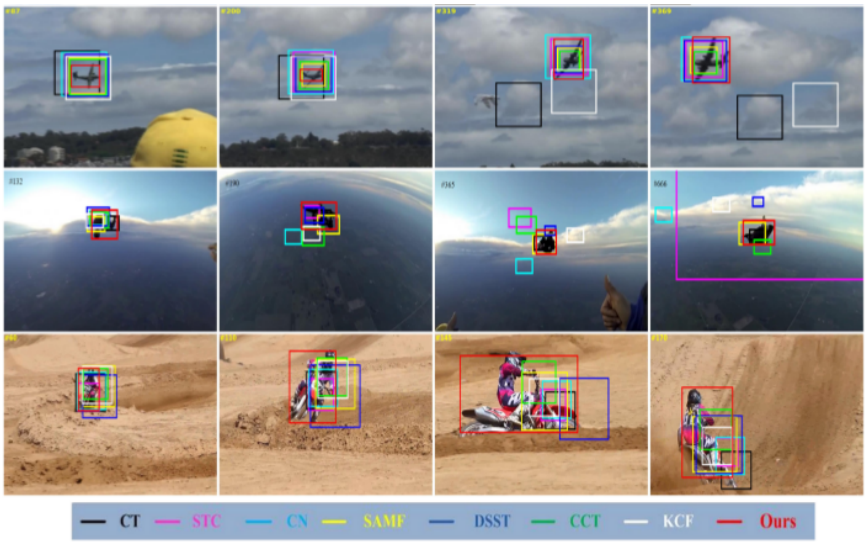

Figure 1.2: Several examples of visual tracking (source: (167)).

sion applications (28; 161). The skip-connection technique (59) solved the problem of gradient vanishing to a certain degree by propagating information across layers of different levels of the network, shortening their connections, which stimulates the popular research in constructing much deeper networks and has obtained improved performance. From the advent of LeNet5 (89) with 5 layers to VGGNet with 16 layers (137), to ResNet (59) which can reach over 1000 layers, the depth of networks has dramatically increased. ResNet-101 (59) shows its advantage of feature extraction and representation, especially when being used as a base network for object detection tasks. VGG network won second place in ImageNet Large Scale Visual Recognition Challenge(ILSVRC) 2014. It is shallow and thin with only 16 layers, which is another widely-used base network. Its advantage lies in the provision with a trade-off between the accuracy and the running speed.

Another approach to enhancing the feature extraction ability is to increase the network width. GoogleNet (149; 150) has realized the activation of multi-size receptive fields by introducing the inception module, which outputs the concatenation of feature-maps produced by filters of different sizes. GoogleNet ranked the first in ILSVRC 2014. It provided a feature expression scheme of the inner layer, which has been widely adopted in later works. The residual-inception and its variances (148; 150; 185) showed their advantage in error-rate over individual inception and

residual technique. SqueezeDet (165) achieved state-of-the-art object detection accuracy on the KITTI validation dataset with a very small and energy-efficient model, which is based on inner-layer inception modules and continuous inter-layer bottleneck filtering units in the year 2017.

However, extracted features by convolution have the possibility of being changed, attenuated, or merged through the long process of forward transmission in deep and complicated networks (94). Furthermore, by processing feature information through the complicated hierarchical structure and transmitting it across multiple layers would reduce the efficiency of feature learning and increase the computational load (164). The most pivotal is although the general feature extraction ability especially for details is powerful enough for most popular CNN models, they suffer from weak semantic expression and contextual inference, leading to poor performance when it comes to small or occluded objects in intense or complicated scenes.

A general factor that determines the performance of deep neural networks in many applications, not just computer vision, but like natural language processing, is actually the regularization. Regularization in deep learning plays an important role to help avoid the bad local minima on the loss surface. In the literature researchers have made great efforts on this topic from different perspectives, such as data (71; 91; 26), network architectures (187; 185; 59), losses (112), regularizers (134; 5; 109), and optimization. Intuitively, orthogonal filters are expected to best span the parameter space, since usually, the filter dimension is larger than the number of filters. However, with many noisy factors such as data samples and stochastic training, it may not be a good idea to strictly preserve the filter orthogonality in deep learning. Recent work has demonstrated that on benchmark datasets, classification accuracy using orthogonal filters (learned by PCA) is inferior to that using learned filters by backpropagation (BP). Another recent work finds that hard constraints on orthogonality can negatively affect the convergence speed and model performance in the training of recurrent neural networks (RNNs), but soft orthogonality can improve the training. We notice that recently orthogonal filters (ORs) have attracted more attention but they are evaluated together with other regularizers, such as weight decay, dropout, and batch normalization. We argue that such experimental settings are hard to identify how much ORs contribute to the

performance, compared with other regularizers. A similar argument has been addressed in where the author showed that l2 regularization has no regularizing effect when combined with batch or weight normalization, but has an influence on weights' scale. These factors determine that most previous regularization work would have a poor generalization and robust performance.

In this thesis, we propose the model that focuses on learning the deep features produced in high-level layers of the network, which fully learns the semantic and contextual information expressed by deep features. The proposed deep feature learning inception modules are able to simultaneously activate multi-scale receptive fields within a single layer level, which equips them with the ability to describe objects against various scenes. In contrast with previous regularization works, we firstly propose transferring the angular prior distribution into deep learning as regularization, which is demonstrated with better generalization than others. Moreover, the proposed method is directly operated on activation, rather than filters, to compute the filter angles efficiently. This method is more related to representation decorrelation and orthogonality regularizers in the literature. The details are discussed in Chapter 3 and Chapter 6.

1.2 Object Detection with Transformer

Modern detectors (133; 128; 96) generate dense anchors using the sliding window method to establish connections between object predictions and the ground truth. Post-processing methods like non-maximum suppression (NMS) is used to remove the redundancy of predictions. While these are hand-crafted components that are unable to achieve the actual end-to-end object detection. Recently, DETR (17) is proposed to build an end-to-end framework based on an encoder-decoder transformer (158) architecture with the bipartite matching loss, which directly predicts a set of bounding boxes without post-processing techniques. Furthermore, the transformer network is based on attention mechanism, which makes it the most general feature association technique in terms of deep learning literature, such as CNN and Multi-Layer Perceptron (MLP) (36). This versatile and powerful relation modeling capability of Transformers with the CNN backbone for feature expression enable DETR to become a significant pipeline for future detectors.

However, DETR comes with the challenges of training and optimization, where large-scale training data and an extremely long training schedule are required. Specifically, DETR has two issues: (1) It requires more training epochs to stable converge than previous modern detectors. (2) DETR delivers relatively low performance when detecting objects with small scales. Deformable DETR (197) tries to solve the two problems by replacing global scoped attention with local spatial attention so as to accelerate the training convergence and introducing multi-scale feature maps for better handling small objects. Up-detr (30) proposes the random query patch detection to pre-train the transformers in DETR for a better trade-off between classification and localization preferences in the pretext task and faster convergence and higher detection accuracy could be expected. While these models all solve the existing problems from the data end by pre-training or the diverse scales of feature maps, which are data-dependent and procedure-complicated for generalization. My research works on solving the problem by Transformers' inner working mechanism so that the proposed model could be data-independent and easy for generalization and inspire further study.

Chapter 2

Background and Literature Review

In this section, we introduce the basic background and review some of the state-of-the-art techniques related to the topics in this dissertation. We, firstly, will briefly lead the fundamental development of deep learning and discuss some import variants of deep neural networks for computer vision. Next, we will review the progresses and challenges inspired by deep learning for the applications of object detection and network regularization.

2.1 Deep Neural Networks

Deep learning is so compelling today that it is more of a mental innovation than a mechanical one. Deep learning seeks to automate intelligence bit by bit, and in the past few years it has achieved enormous success and progress in this endeavor, exceeding previous records in Computer Vision, Speech Recognition, and many other fields. Deep neural network is the powerful deep learning technique, which makes use of multi-layers artificial neurons in hierarchical architectures. In this section, we will discuss five main topics related to deep neural networks and computer vision, including Deep forward Networks (64; 49; 114), Convolutional Neural Networks (CNNs) (89), Inception Module (149), popular object detectors (128; 99), and Deep Learning Regularization (91; 52).

2.1.1 Deep Forward Networks

Deep feedforward networks, also called **feedforward neural networks**, or **multilayer perceptrons** (MLPs), are the quintessential deep learning models (52). The feedforward network aims to approximate some function f^*, like $y = f^*(\mathbf{x})$ which maps an input $\mathbf{x}$ to a category y. Thus, the mapping that a feedforward network defines can be represented as below,

$$\mathbf{y} = f(\mathbf{x}; \theta) \tag{2.1}$$

which learns the parameters θ that determine the best function approximation. The so-called **feedforward** refers to that information flows through the function being evaluated from $\mathbf{x}$, through the intermediate computations used to define f, and to the output $\mathbf{y}$ in the end. And there are no feedback connections in which outputs of the model are fed back into the model itself. For example, the convolutional networks used for object recognition from images are a specilized kind of feedforward network. If considering the feedback connections, the feedforward neural networks are called recurrent neural networks, which usually work for the natural language process.

The structures of neural networks are most commonly chain structures that can be expressed by the following form,

$$f(\mathbf{x}) = f^n(f^{n-1}(f^{n-2}(...f^2(f^1(\mathbf{x})))))) \tag{2.2}$$

where each function f refers to one layer. Thus, in this case, f^n indicates the n_{th} layer, f^{n-1} the $n-1_{th}$ layer, f^2 the second layer, and the f^1 the first layer. The overall length of the chain gives the depth of the model. The final layer of the feedforward network is called the output layer. The other layers except for the input and output layers are hidden layers, with which the learning algorithm implement an approximation of f^*. The dimensionality of these hidden layers determines the width of the network.

Linear models, such as logistic regression and linear regression, can be fit efficiently and reliably, either in closed form or with convex optimization. But they have the obvious defect that the model capacity is limited to linear functions that can not understand the interaction between

any two input variables. Thus, we apply the linear model not to **x** itself but to a transformed input gx, where g is a nonlinear transformation. We use the following nonlinear function to describe the features,

$$\mathbf{h} = g(\mathbf{W}^T\mathbf{x} + \mathbf{c}) \tag{2.3}$$

where **W** provides the weights of a linear transformation and **c** the biases. Now suppose we choose the linear model with θ consisting of **w** and b, a vector of weights and a scalar bias parameter respectively. The model is defined to be,

$$y = f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^T\mathbf{x} + b \tag{2.4}$$

Then combining with the nonlinear function of **h**, the complete operation for a single layer of a network can be specified as,

$$y = \mathbf{w}^T\mathbf{h} + b = \mathbf{w}^T(g(\mathbf{W}^T\mathbf{x} + \mathbf{c})) + b \tag{2.5}$$

In modern neural networks, the default recommendation for function g is to use the **rectified linear unit**, or ReLU (73; 114; 50).

2.1.1.1 Gradient-based Learning

To build a deep learning algorithm, we need to specify an optimization procedure, a cost function, and a model family. The neural network is usually trained by using iterative, gradient-based optimizer that merely drive the cost function to a very low value, rather than the linear equation solvers used to train linear regression models or the convex optimization algorithms with global convergence guarantees used to train logistic regression or SVMs. In the following sections, we will come the details of how to obtain the gradient using the back-propagation algorithm and modern generalizations of the back-propagation algorithm. To apply gradient-based learning, there should be a defined cost function and a represented output of the network model.

2.1.1.2 Cost Functions

In most cases, deep neural networks define a distribution $p(\mathbf{y}|\mathbf{x};\theta)$ and use the principle of maximum likelihood. This means the cross-entropy between the training data and the predictions is used as the cost function. The entire cost function used to train a neural network often combines one of the primary cost functions described here with a regularization term. For the most common training with maximum likelihood, the cost function is simply the negative log-likelihood, equivalently described as the cross-entropy between the training data and the model distribution. This cost function can be described as below.

$$J(\theta) = -\mathbb{E}_{\mathbf{x},\mathbf{y}} \log p(\mathbf{y}|\mathbf{x}) \tag{2.6}$$

The specific form of the cost function changes with different models, depending on the definition of $\log p$. The advantage of deriving the cost function from maximum likelihood is that specifying a model $p(\mathbf{y}|\mathbf{x})$ automatically determines a cost function $\log p(\mathbf{y}|\mathbf{x})$. The disadvantage property of the cross-entropy cost used to perform maximum likelihood estimation is it tends not to having a minimum value when applied in practice. If the model can control the density of the output distribution then it becomes possible to assign extremely high density to the correct training set outputs, leading to cross-entropy approaching negative infinity. Regularization techniques in the next section will discuss several ways of modifying the learning problem so as to avoid the related problem mentioned above.

Besides learning a full probability distribution $p(\mathbf{y}|\mathbf{x};\theta)$, we often want to learn just one conditional statistic of $\mathbf{y}$ given $\mathbf{x}$. Solving an optimization problem with respect to a function requires the calculus of variations. One derived result using calculus of variations is to solve the optimization problem,

$$f^* = \arg\min_f \mathbb{E}_{\mathbf{x},\mathbf{y}} \|\mathbf{y} - f(\mathbf{x})\|^2 \tag{2.7}$$

By minimizing the mean squared error cost function, it would give a function that predicts the

mean of **y** for each value of **x**. The other result derived using calculus of variations is,

$$f^* = \arg\min_f \mathbb{E}_{\mathbf{x},\mathbf{y}} \|\mathbf{y} - f(\mathbf{x})\|_1 \tag{2.8}$$

which yields a function that predicts the median value of **y** for each **x**.

However, the two derived functions above easily lead to poor results when used with gradient-based optimization, embodied by small gradients. This shows the reason that the cross-entropy cost function is more popular than them , even though it is not necessary, in some cases, to estimate the entire distribution $p(\mathbf{y}|\mathbf{x})$.

2.1.1.3 Back-Propagation

During training, forward propagation can continue onward until it produces a scalar cost $J(\theta)$. The **back-propagation** algorithm((136)) allows the information from the cost to flow backward through the network for computing the gradient. Back-propagation refers only to the method for computing the gradient, while other algorithms, like stochastic gradient descent, is to perform learning using this gradient. The most often gradient is of the cost function with respect to the parameters, $\nabla_\theta J(\theta)$.

Chain Rule of Calculus. The chain rule of calculus is used to compute the derivatives of functions by composing other functions whose derivatives are known. Back-propagation is an algorithm that computes the chain rule, with a specific order of operations that is certified highly efficient. Specifically, for the map that can be derived, we can use the variable parameters of the map to deduct and use the gradient descent method to slowly make the variable parameters close to what is expected which makes the loss smaller. Deep learning imposes multiple mappings and performs multiple mapping operations on the input. In theory, the performance of multiple mapping will be higher than a single mapping. After optimization, the loss can be made smaller. Suppose **x** is the input of the network, which is usually a vector or a matrix in the field of computer vision, and z represents the calculated loss from the cost function. Function net_1, net_2,..., net_{n-1}, and net_n

represent mappings of different layers, and $\mathbf{o}_1$, $\mathbf{o}_2$,...,$\mathbf{o}_{n-1}$, and $\mathbf{o}_n$ are their outputs respectively. The $\mathbf{x}$ here refers to the input data, but the meaning of the input value is not significant, because we can not change the data to make the target cost smaller. The actual situation is to change the variable contained in each map, like the weights and bias. For example, if the θ_m is the variable in map m, we can calculate $\partial z/\partial \theta_m$ by the following chain rule of calculus.

$$\frac{\partial z}{\partial \theta_m} = \frac{\partial z}{\partial \mathbf{o}_n}\frac{\partial \mathbf{o}_n}{\partial \mathbf{o}_{n-1}}\cdots\frac{\partial \mathbf{o}_{m+1}}{\partial \mathbf{o}_m}\frac{\partial \mathbf{o}_m}{\partial \theta_f} \tag{2.9}$$

2.1.2 Network Learning and Optimization

Deep neural networks learn a certain task by minimizing the loss value defined by the object function. This process relies on updating and finding the parameters θ of a neural network that significantly reduce the cost function $J(\theta)$. This is realized by an optimization technology, used as the training algorithm in this deep learning case.

Stochastic gradient descent (SGD) is probably the most used optimization algorithm for deep learning in general. It is possible to obtain an unbiased estimate of the gradient by taking the average gradient on a mini-batch of m examples. As for the learning rate, the crucial parameter of SGD, we usually gradually decrease it over iterations and denote it at iteration k as ε_k. It is common to decay the learning rate linearly until iteration τ. After τ, it is usually to leave ε constant. Suppose there is a mini-batch of m examples from the training set $\{\mathbf{x}^{(1)},...,\mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$. Then the estimated gradient, $\hat{\mathbf{g}}$ can be expressed as below, and be updated right after the iteration i,

$$\hat{\mathbf{g}} = \frac{1}{m}\nabla_\theta \sum_i L(f(\mathbf{x}_{(i)};\theta),\mathbf{y}^{(i)}) \tag{2.10}$$

$$\theta = \theta - \varepsilon_k \hat{\mathbf{g}} \tag{2.11}$$

The property of SGD that really matters during training is that the computation time per update does not grow with the number of training examples, which allows convergence even when the number of training examples becomes much larger.

2.2 Convolutional Neural Network (CNN)

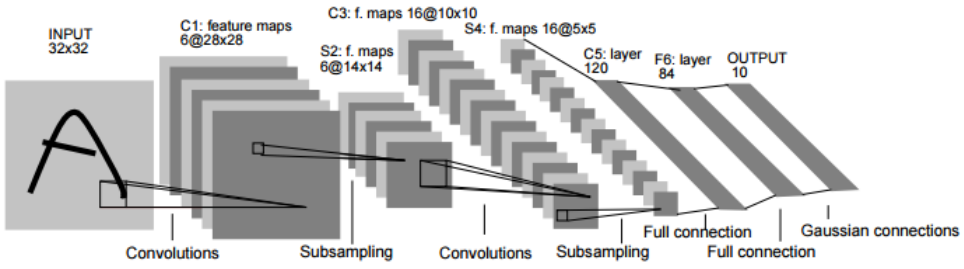

Figure 2.1: The architecture of LeNet-5 network (source: (89)).

The evolution of convolutional neural network (CNN) is inspired by the progress in visual perception mechanism of the living creatures (53). (42) introduced the concept of local receptive fields (70) into their neocognitron. (88) proposed the modern framework of CNN and it was later improved by (89). A classical CNN network LeNet-5 is shown in Figure 2.1.

Convolutional Neural Networks (CNN) process data that has a known grid-like topology, like time-series data and image data. They use convolution in place of general matrix multiplication in at least one of the layers. The convolutional layer is the core building block of a deep CNN. It consists of a set of learning filters with weights and biases. The most important idea that CNN leverages to improve the performance of deep learning system is **parameter sharing**, which provides a means for working with inputs of variable sizes and a solution for avoiding parameter explosion of traditional neural networks.

Traditional neural network layers use matrix multiplication where each output unit interacts with every input unit. This would introduce a huge burden of parameters by just several layers, which is a challenge for memory storing and severely limits the representation ability of a neural network. It is a reason why traditional neural networks hits its developing bottleneck decades ago. The development of CNN broke this bottleneck and realized the efficient processing for large-scale input data by parameter sharing. Parameter sharing uses the same parameter for more than one convolutional filters in a model layer. Each filter (kernel) is used at every position of the input. The parameter sharing used by the convolution operation means that rather than learning a separate

set of parameters for every location, we learn only one set.

As shown in Fig. 2.1, the region against the original input image that the filter activates is called the **receptive filed**. This is determined by the filter size. The third dimension of the filter is called the channel number, which depends on the input volume. After convolution, each filter will produce a 2-dimensional feature map as the activation response of a specific spatial location. An element-wise nonlinear activation function is commonly used on the feature map.

Considering the operation efficiency and the learning towards semantic information by larger receptive field, **the pooling layer** is usually introduced after several convolutional layers. This pooling layer, also known as subsampling layer, does not consist of learnable parameters. It reduces the spatial size of input volume by down-sampling. The typical subsampling methods are max pooling or average pooling, which extracts the maximum value or the average value from the subregion.

For final information fusion, **the fully connected layer** is usually introduced in the top part of the network. It has dense connection with its input volume. As shown in Fig. 2.1, the $F6$ layer has 84 neurons, and the output layer has 10 neurons. The 84 neurons on $F6$ layer are densely connected with neurons on the output layer.

2.3 Object Detection with Convolutional Neural Network

The biologically-inspired convolutional networks have recently attracted lots of attention in the computer vision community, where object classification and detection is one of the most popular applications. Conventionally, features are extracted from input images by robust hand-craft method (Harr by (119), SIFT by (102), HOG by (31)) for the task of object detection. There are two typical approaches that are based on region proposal and sliding window, respectively. Before the advent of deep neural networks, the state-of-the-art models for each approach are Selective search by (155) and Deformable Parts model (DPM) by (40). Selective search integrates exhaustive search and segmentation to generate all the possible object locations (155). DPM builds part representations of objects in a graphical model and learns a discriminative part-based models for variety of objects

classes (40).

The deep neural networks gained extensive attention since the success on large-scale image classification task by (83). R-CNN (48), as a region-based approach, dramatically boosted the performance improvement of object detection. R-CNN (48) leverages selective research to generate region proposals on a given image (155), and extracts features by a convolutional network. It uses Support Vector Machine (SVM), a discriminative classifier, to score the bounding boxes, and non-maximum suppression (NMS) to rank the proposed bounding boxes, eliminating duplicate detection. R-CNN, nevertheless, is time-consuming. Later on, the spatial pyramid pooling (SPP) layer is introduced to speed up the R-CNN by SPP-Net (58). It is able to adaptive to multi-scale region proposals and to share the features computed over feature maps generated at multiple image resolutions with the classification layers. (47) extends SPP-Net in Fast R-CNN, where we can fine-tune the network end-to-end in the framework of multi-task learning by minimizing the loss for both bounding box and confidence regressions.

The further development of object detection started from generating high quality region proposals by deep neural networks. In Faster R-CNN, (133) proposes a Region proposal network (RPN) by sharing the computation and generating the proposals using two neural networks. (16) proposes a multi-scale deep CNN for fast object detection by integration of two sub-networks, one is the proposal network and the other is the object detection network. (80) introduces a deep hierarchical network to handle region proposal generation and object detection jointly by aggregating hierarchical feature maps and compressing them into a uniform space. (186) learns a detector by integrating hierarchical features with object context and DeepMask object proposals (123). (182) proposes a cascaded rejection classifier to eliminate negative object proposals and to learn convolutional features. (142) learns a region-based object detector using an online hard example mining algorithm to simplify training.

We usually name the above methods by two-stage models. In OverFeat, (141) adopts the sliding window approach in a convolutional neural network to perform classification, localization, which adapts the localizer to predict bounding boxes directly for detection. YOLO (128) and SSD (99)

formulate object detection as a regression problem in a single-shot deep neural network, optimized via end-to-end training strategy. Those methods are usually called object-stage method. They are based on grid approach, where each image is separated into several grids during both training and test inference. YOLO predicts the confidences of multiple categories for each grad cell with an object and the bounding boxes using the whole topmost feature map. SSD adopts a fixed set of boxes for a certain possible object and predicts the confidence score of each object category within a box. (115) regresses the bounding boxes in a multi-scale grid framework. (8) uses spatial recurrent neural network to integrate the contextual information and leverages skip pooling to exploit the multi-scale representations of the objects. (10) learns a hierarchical object detector by a deep reinforcement learning agent. RetinaNet (96) and RefineDet (189), as one-stage methods, obtain the top object detection performances on the challenging benchmark COCO (97) in the recent two years. RetinaNet addresses the class imbalance by reshaping the standard cross entropy loss so as to down-weight the loss assigned to well-classified examples, and introduces the feature pyramid network in its backbone for an efficient, rich, and multi-scale feature expression from a single resolution input image. RefineDet proposes the anchor refinement by filtering negative anchors and adjusting the locations and sizes of them, and the transfer connection block by transferring the features in anchor refinement module to further improve the regression and multi-class label prediction.

2.4 Feature Expression and Inception Technology

As a fundamental and vital step of vision and multimedia tasks, feature expression has been widely studied (43; 149; 148), especially on the level of network structures. Deeper or wider networks amplify the differences among architectures and gives full play to improve feature extraction ability (28; 161).

The skip-connection (59) solved the problem of gradient vanishing by propagating information across different layers of the network, shortening their connections, which enables to construct much deeper networks. From the advent of LeNet5 (89) with 5 layers to VGGNet with 16

layers (137), to ResNet (59) reaching over 1000 layers, the depth of networks has dramatically increased. The ideas of residual connections (59; 67), dense connection (66) dedicate to substantially enhancing feature expression by strengthening feature propagation, and encouraging feature reuse, especially the details from low-level layers.

Another approach to enhancing the feature expression ability is to increase the network width. GoogleNet (149; 150) realizes the activation of multi-scale receptive fields by introducing the inception module, which outputs the concatenation of feature-maps produced by filters of different sizes. GoogleNet ranked the first in ILSVRC 2014. It provides a feature expression scheme of inner layer. The residual-inception and its variances (148; 150; 185) show their advantage in error-rate over simple inception and residual technique. SqueezeDet (165) achieves the state-of-the-art object detection accuracy on the KITTI validation dataset with a very small and energy efficient model, which is based on inner-layer inception and continuous inter-layer bottleneck filtering in the year of 2017.

2.4.1 Inception Technology

To boost CNN performance simply by going deeper or wider would easily bring about following problems:

- Explosive parameters which easily leads to over-fitting;
- Complex computation which tends to poor adaptation;
- Vanishing gradients which is liable to hard optimization.

Although reducing the number of parameters would solve the above problems, the change from full connection to sparse connection will not substantially boost the computation performance as the computation time towards sparse matrix is hard to decrease. The Inception proposed by GoogLeNet (149) is able to cluster sparse matrices to intense submatrices for efficient computation, which is a new "basic neuron" architecture for building sparse and efficient networks.

The initial inception deep convolutional architecture was named Inception-v1 (149). Later the Inception architecture was refined in various ways, first by the introduction of batch normalization (71) (Inception-v2), and second by additional factorization ideas in the third iteration (150) which is referred to as Inception-v3, and third by more inception blocks than Inception-v3 in Inception-v4 (148). Later the author combines the Inception architectures with residual connection to develop the Inception-ResNet (148) for further boosting the performance.

For Inception-v1 in Figure(a), it combines the common 1×1, 3×3, and 5×5 convolutions in the same layer level, which increases the width of the network and improves the adaptation towards scale variation at the same time. The pooling layer reduces data scale for alleviating over-fitting. However, the filter kernel with large size means more parameters although it provides with larger receptive field. For example, there are 25 parameters for 5×5 convolution kernel while only 9 parameters for $3x3$ convolution, where the former one is 2.78 times the latter one. Thus, in Inception-v2 shown in Figure(b), GoogLeNet proposes leveraging 2 continuous $3x3$ convolution to replace the single 5×5 convolution, which maintains the scope of receptive field and reduces the parameters. By intensive experiments, GoogLeNet certifies that this replacement will not loose the ability of feature expression. For Inception-v3, it introduces the "Fraction", where 7×7 is replaced by two 1D convolution, 1×7 and 7×1 as the illustration shows, similar operation for 3×3 as well. This accelerates the computation and deepen this module with benefit of strengthening the non-linear of the network. Inception-v4 combines inception and residual technique, bringing about higher speed and better performance.

2.5 Regularization

Regularization in deep learning plays an important role to help avoid the bad local minima on loss surface. In the literature researchers have made great efforts on this topic from different perspectives, such as data (71; 2; 26), network architectures (187; 59; 185), losses ((112)), regularizers (52; 134; 191; 4; 110), and optimization (65; 140; 68). There are many specific regularization techniques. For instance, weight decay is essentially an ℓ_2 regularizer over filters, dropout takes

random neurons for update, and BN utilizes the statistics from mini-batches to normalize the features. As our work is more related to representation decorrelation and orthogonality regularizers in the literature, more details below will be about them.

Representation Decorrelation: Cogswell (26) proposed a regularizer, namely DeCov, to learn non-redundant representations by minimizing the cross-covariance of hidden activation. Similarly, Gu (54) proposed another regularizer, namely Ensemble-based Decorrelation Method (EDM), by minimizing the covariance between all base learners (hidden activation) during training. Yadav and Agarwal (181) proposed regularizing the training of RNNs by minimizing non-diagonal elements of the correlation matrix computed over the hidden representation, leading to DeCov RNN loss and DeCov Ensemble loss. Zhu (198) proposed another decorrelation regularizer based on Pearson correlation coefficient matrix working together with group LASSO to learn sparse neural networks.

Orthogonality Regularizers: Harandi and Fernando (56) proposed a generalized BP algorithm to update filters on the Riemannian manifolds as well as introducing a Stiefel layer to learn orthogonal filters. Vorontsov (160) verified the effect of learning orthogonal filters on RNN training that is conducted on the Stiefel manifolds. Huang (68) proposed an orthogonal weight normalization algorithm based on optimization over multiple dependent Stiefel manifolds (OMDSM). Xie (173) proposed a family of orthogonality-promoting regularizer by encouraging the Gram matrix of the functions in the reproducing kernel Hilbert spaces (RKHS) to be close to an identity matrix where the closeness is measured by Bregman matrix divergences. Rodríguez (134) proposed a regularizer called OrthoReg to enforce feature orthogonality locally based on cosine similarities of filters. Bansal (4) proposed another two orthogonality regularizers based on mutual coherence and restricted isometric property over filters, respectively, and evaluated their gain in training deep models.

2.6 Attention Mechanism and Transformers

Attention mechanism is an integral part of compelling sequence modeling and transduction models in various tasks. It allows modeling of dependencies without considering their distance in the input

or output sequences (3; 77). The early most attention mechanisms are used in conjunction with a recurrent network (120).

The attention function maps a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is the weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key (158).

Dot-Product Attention: Dot-Product Attention is particularly called "Scaled Dot-Product Attention". Its input consists of queries and keys with the dimension of d_k, and values with the dimension of d_v. The query is used to compute the dot products with all keys, and then the dot product each is divided by $\sqrt{d_k}$, and finally a softmax function is applied to obtain the weights on the values. In practice, the attention function is computed simultaneously on a set of queries, packed together into a matrix Q. The keys and values are also packed together into matrices K and V . The output matrix could be described as,

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.12}$$

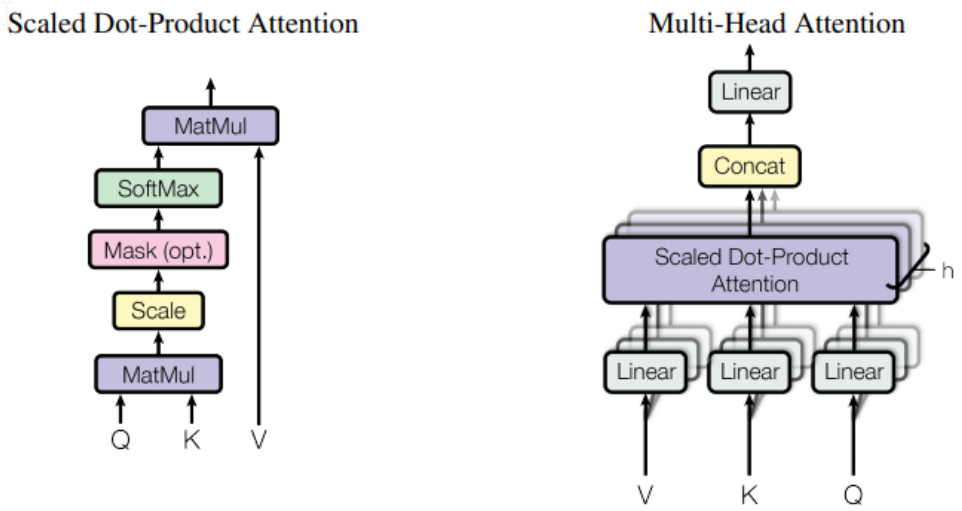

Figure 2.2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel (source: (158)).

Multi-Head Attention: It is found more beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions respectively, in comparison with conducting a single attention function with d_{model}-dimensional keys, values and queries (158). The attention function then is performed in parallel on each of these projected versions of queries, keys and values, which yields d_v-dimensional output values. These are concatenated and projected again, resulting in the final values, just as the depict in Figure 2.2. Multi-head attention enables the model to jointly attend to information from different representation subspace at different positions (158).

$$\begin{aligned} \text{MultiHead}(Q,K,V) &= \text{Concat}(head_1,\ldots,head_h)W^O \\ \text{where head}_\text{i} &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.13}$$

where the parameter matrices $W_i^Q \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^K \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^V \in \mathbb{R}^{d_{model}\times d_v}$ and $W_i^O \in \mathbb{R}^{hd_v\times d_{model}}$ are the projections.

Self-attention: Self-attention is the attention mechanism that relates different positions of a single sequence so as to compute a representation of the sequence (158). Self-attention has been successfully applied in a various tasks ranging from reading comprehension, abstractive summarization, textual entailment, learning task-independent sentence representations, and detection related vision tasks (24; 120; 122; 98; 158; 17).

Transformer: Transformer is a model architecture entirely relying on an attention mechanism to draw global dependencies between input and output. The Transformer enables significantly more parallel and can obtain a new state of the art in translation quality and a competitive detection accuracy (158; 17).

Transformer applies multi-head attention in the encoder-decoder attention layers, the queries coming from the previous decoder layer, and the memory keys and values produced from the output of the encoder. This enables every position in the decoder to attend over all positions in the input sequence. The encoder contains self-attention layer, where all of the queries, keys, and values come from the same place, the output of the previous layer in the encoder in this case.

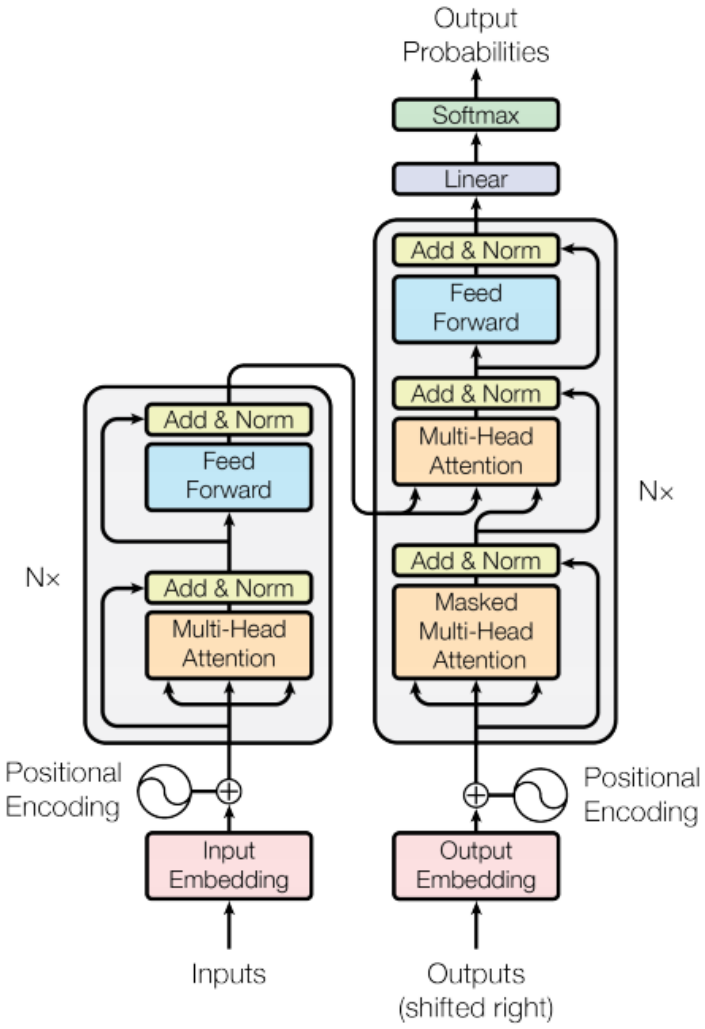

Figure 2.3: The Transformer - model architecture (source: (158)).

Each position in the encoder can attend to all positions in the previous layer of the encoder (158). Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. The details of Transformer connection architecture is depicted in Figure 2.3.

Chapter 3

Multi-Scale Deep Feature Learning Network for Object Detection

In this section, we discuss the transverse semantic, the spatial semantic features, exploring how it influences the feature expression and vision application. We show some preliminary results of deep feature learning based object detection and near-orthogonality regularization via angular prior distribution transfer.

3.1 Multi-Scale Deep Feature Learning Network for Object Detection

In this section, we illustrate our results for object detection using the proposed deep feature learning partly based on our paper by (106).

This paper proposes an innovative object detector by leveraging deep features learned in high-level layers. Compared with features produced in low-level layers, the deep features are better at expressing semantic and contextual information. The proposed deep feature learning scheme shifts the focus from concrete features with details to abstract ones with semantic information. It considers not only individual objects and local contexts but also their relationships by building multi-scale deep feature learning network (MDFN). MDFN efficiently detects the objects by the proposed deep feature leanring inception modules into the high-level layers, which employ

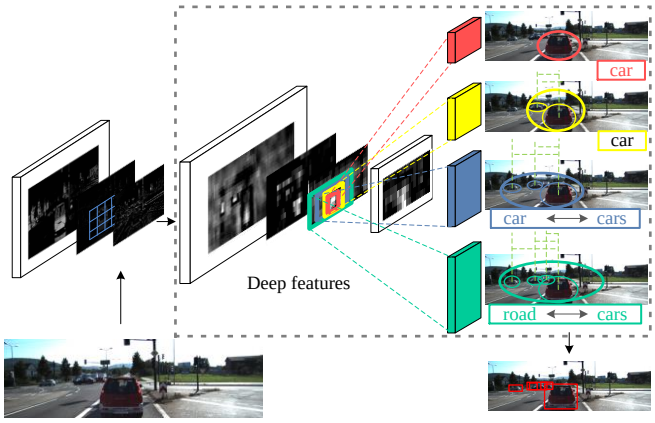

Figure 3.1: The flow-chart of the proposed fast object localization and tracking strategy.

parameter-sharing to enhance the computational efficiency. MDFN provides a multi-scale object detector by integrating multi-box, multi-scale and multi-level technologies. MDFN achieves better or competing detection results than those models with macro hierarchical structures, by building a simple framework with a relatively small base network (VGG-16).

MDFN integrates the semantic and contextual information into the learning process through a single-shot framework. As shown in Figure 3.1, the abstract features with semantic and contextual expression can be activated by multi-scale receptive fields on feature maps. The red, yellow, blue and green components represent four sizes of filters, which correspond to different object descriptions. For example, the red one tends to be sensitive only to the red vehicle in the middle, while the yellow and the blue ones also cover the small cars around it, providing the correlations among different object cars. The green one has the largest activation range, and it not only detects all vehicles but also the road, passing out the relationships between objects and their background. This process that conveys various semantic expression can be realized in deep layers where the receptive fields are able to cover larger scenes and the feature maps are equipped with strong abstract ability (22; 94). Detail discuss and analysis are shown in the following sections.

3.1.1 Deep Feature Learning

The feature maps produced in deep layers are believed to be used for removing irrelevant contents and extracting semantic and the most important characteristics of objects against background. In contrast, the activation towards feature maps from earlier layers are supposed to extract various details, such as textures or contours of objects or concrete background description (143; 107). Currently, most deep convolutional neural networks suffer from the detection of small and occluded objects, which has not been well solved even with much more complicated models (99). In our study, we claim that the detection of small and occluded objects depends not only on detail features but also on sufficient semantic and contextual expression (163; 162). Deep features have better expression towards the main characteristics of objects and more accurate semantic description of the objects in the scenes (94; 164).

3.1.1.1 Deep Feature Extraction and Analysis

With the increase of layers, feature maps produced from the deep part of the network become abstract and sparse, with less irrelevant contents, serving for the extraction of the main-body characteristics of the objects (143; 107). These feature maps have smaller scales but correspond to larger receptive fields. This determines their function of deep abstraction to distinguish various objects with better robust abstraction so that features are relatively invariant to occlusion (130). Since feature maps from intermediate levels retrieve contextual information either from their shallower counterparts or from their deeper counterparts (130), the extraction of deep features from consecutive layers is necessary. Our model will directly process high-resolution feature information from base network and their outputs will be directly fed to the output layers for shortening the path of feature propagation and enhancing the efficiency of feature usage. The following analysis will provide the theoretical support for this scheme.

Feature extraction in CNN can be expressed as a series of non-linear filtering operations as follows (130).

$$\Psi_n = f_n(\Psi_{n-1}) = f_n(f_{n-1}(...f_1(X))) \tag{3.1}$$

where Ψ_n refers to the feature map in layer n, and f_n denotes the n-th nonlinear unit which transforms the feature map from layer $n-1$ to the n layer.

$$\hat{O} = O(T_n(\Psi_n), ..., T_{n-l}(\Psi_{n-l})), n > l > 0 \tag{3.2}$$

In equation (2), T_n is the operation transmitting the output feature maps from the n-th layer to the final prediction layer. Thus, equation (2) is an operation of multi-scale output; and O stands for the final operation that considers high-level outputs.

According to (130), equation (2) performs well relying on the strong assumption that each feature map being fed into the final layer has to be sufficiently sophisticated to be helpful for detection and accurate localization of the objects. This is based on the following assumptions: 1) These feature maps should be able to provide fine details especially for those from earlier layers; 2) the function that transforms feature maps should be extended to the layers that are deep enough so that the high-level abstract information of the objects can be built into feature maps; 3) the feature maps should contain appropriate contextual information such that the occluded objects, small objects, blurred or overlapping ones can be inferred exactly and localized robustly (130; 133; 144; 22; 94). Therefore, the features from both the shallow and deep layers play indispensable roles for the object recognition and localization. Moreover, the feature maps from the intermediate levels retrieve contextual information either from their shallower counterparts or from their deeper counterparts (130). Thus, some work tries to make the full utilization of the features throughout the entire network and realizes connections across layers as many as possible, so as to maximize the probability of information fusion, like DenseNet (66).

Although maximizing information flow across the entire network would be able to make full use of feature information, there is a possibility that this intense connection across layers does not achieve the expected effectiveness as the negative information would also be accumulated and passed during the transmission process, especially in the deep layers (159; 164). Moreover, features with low intensity values are easy to be merged (100), and the heavy computation load

can not be avoided. This analysis can be shown in the following equation.

$$\Psi_n + \delta_n = f_n(\Psi_{n-1} + \delta_{n-1}) = f_n(f_{n-1}(...f_{n-p}(\Psi_{n-p} + \delta_{n-p}))), n > p > 0 \tag{3.3}$$

where δ_n is the accumulated redundancy and noise that existed in layer n, and δ_{n-p} is the corresponding one in shallow layers.

Based on the above analysis, in order to efficiently exploit the detected feature information, another constrained condition should be considered so as to prevent the features from being changed or overridden. We claim that the feature transmission across layers should decrease the probability of features being changed by drift errors or overridden by the irrelevant contents, and should minimize the accumulation of the redundancy and noise especially in the deep layers. Thus, feature transmission within the local part of the network or direct feature-output should be a better solution to effectively employ this information. To this end, we propose the following multi-scale deep feature extraction and learning scheme, which will support the above strong assumptions and satisfy the related constrained conditions.

$$\Psi_m = F_m(\Psi_{m-1}) = F_m(F_{m-1}(...F_{m-k}(\Psi_n))), m-k > n \tag{3.4}$$

$$\begin{aligned} F_j = S(f_j(f_j(f_j(\Psi_{j-1}))), f_j(f_j(\Psi_{j-1})), \\ f_j(\Psi_{j-1}), \Psi_{j-1}; W_j), m-k \leq j \leq m \end{aligned} \tag{3.5}$$

$$\hat{O} = O(T_m(\Psi_m), T_{m-1}(\Psi_{m-1}), ..., T_{m-k}(\Psi_{m-k}), T_{m-j}(\Psi_{m-j})), j > k \tag{3.6}$$

where m indicates high-level layers, Ψ_m is the corresponding output feature maps of layer m. The function F_j maps Ψ_{j-1} to multi-scale spatial responses in the same layer j. F is functioned by S, the feature transformation function, and weighted by W. All feature information produced in high-level layers would be directly fed to the final detection layer by the function T. Ψ_{m-j} represents the feature map from some shallow layer. The inputs of function O include feature maps from low-level layers and those from high-level layers.

The above scheme considers feature maps produced from shallow layers which have high reso-

lution to represent fine details of objects. This is in accordance with the first mentioned assumption. F_m are designed for deep layers of the network to introduce deep abstraction into the output feature maps. Moreover, multi-scale receptive fields within a single deep layer are sensitive to most important features and objects in the contexts of different sizes, which makes the output powerful enough to support the detection and localization, and directly responds to the strong suggestion. At the same time, several continuous deep inception units provide the probability that feature maps from intermediate levels can retrieve contextual information from both lower and deeper counterparts. This is beneficial to detecting the exact locations of overlapped objects, occluded, small, and even blurred or saturated ones which need to be inferred robustly (133; 144). This satisfies the above assumptions 2) and 3).

Instead of building connections across layers, the consecutive deep inception realizes the same function of multi-scale feature maps, abstraction and contextual information built-in and simultaneously avoids the problem of introducing redundancy and noise as described in the proposed constrained condition. Moreover, the multi-scale inceptions would produce more variety of information, rather than simply increase the information flow by connections across the layers. Based on the above analysis, the proposed model makes the training smoother and achieves a better performance of localization and classification.

3.1.1.2 Deep Feature Learning Inception Modules

Deep feature learning inception modules capture the direct outputs from the base network. Our basic inception module makes full use of the deep feature maps by activating multi-scale receptive fields. In each module, we directly utilize the output feature information from the immediate previous layer by 1×1 filtering. Then, we conduct 3×3, 5×5 and 7×7 filtering to activate various receptive fields on the feature maps so as to capture different scopes of the scenes on the corresponding input images. We realize the multi-scale filtering only with the 1×1 and 3×3 filters in practice to minimize the number of parameters (150; 106). We build two types of power operation inception modules for the high-level layers: one is information square inception module, and the

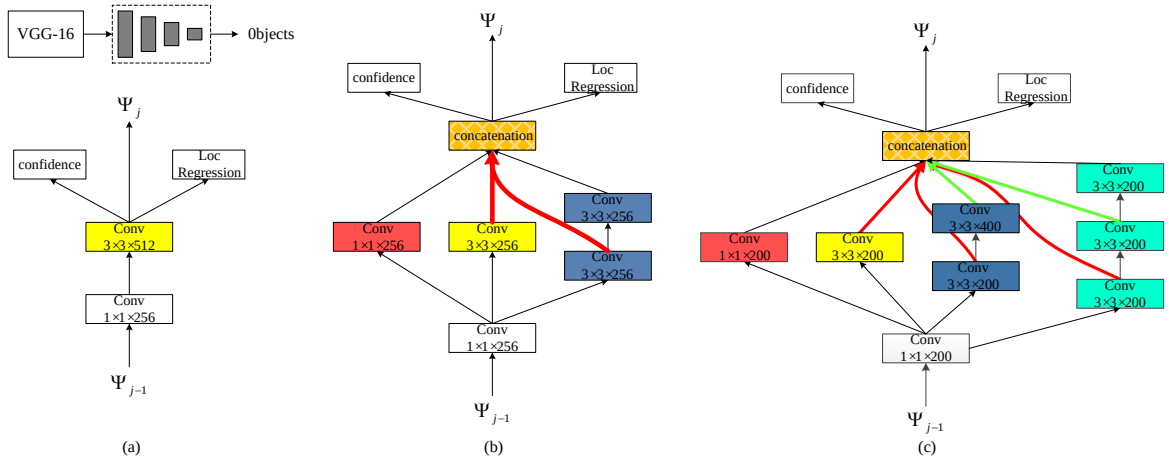

Figure 3.2: **Deep feature learning inception modules.** (a) The core and basic deep feature transmission layer structure. (b) and (c) denote the two actual layer structures of information square and cubic inception modules. Red and green arrows indicate the way of parameter sharing. Ψ_{j-1} represents feature maps from previous layer and Ψ_j denotes the output feature maps from current layer.

other is information cubic inception module, as shown in Figure 3.2. We build these two modules by assigning weights to different filters as given in the following equations, where the two operations are denoted by F_j^2 and G_j^3, respectively.

$$F_j^2(\Psi_{j-1}) = f_j(f_j(\Psi_{j-1})) + 2 \times f_j(\Psi_{j-1}) + \Psi_{j-1}, m-k \leq j \leq m \tag{3.7}$$

$$\begin{aligned} G_j^3(\Psi_{j-1}) =& g_j(g_j(g_j(\Psi_{j-1}))) + 3 \times g_j(g_j(\Psi_{j-1})) + \\ & 3 \times g_j(\Psi_{j-1}) + \Psi_{j-1}, m-k \leq j \leq m \end{aligned} \tag{3.8}$$

where the 5×5 filer is replaced by two cascaded 3×3 filters and the 7×7 filter is replaced by three cascaded 3×3 filters. This replacement operation has been verified to be efficient in (150). The number of parameters of the two cascaded 3×3 filters only accounts for 18/25 of that of one single 5×5 filter (150). By manipulation, the expressions of (7) and (8) can be approximated by the following information square and cubic operations, respectively.

$$F_j^2(\Psi_{j-1}) = (f_j^2 + 2 \times f_j + 1)(\Psi_{j-1}) = ((f_j + 1)^2)(\Psi_{j-1}) \tag{3.9}$$

$$G_j^3(\Psi_{j-1}) = (g_j^3 + 3 \times g_j^2 + 3 \times g_j + 1)(\Psi_{j-1}) = ((g_j + 1)^3)(\Psi_{j-1}) \tag{3.10}$$

3.1.1.3 Parameter Sharing

The proposed information square and cubic inception modules can be implemented efficiently by sharing parameters. For example, we share parameters between the 3×3 and 5×5 filtering units by extracting outputs from the first 3×3 filter of the 5×5 unit and concatenate it with the parallel outputs from the 3×3 filtering unit. Then, the number of output tunnels of the 3×3 filtering operation is implicitly doubled while the set of filters are only used once, as indicated by the red arrows in Figure 3.2 (b). This parameter-sharing can be further used in the cubic inception module as shown in Figure 3.2 (c). The outputs of the 3×3 filtering operation come from the 3×3, 5×5, and 7×7 filtering unit respectively as indicated by the three red arrows in Figure 3.2 (c). Similarly, those outputs of the 5×5 filtering operation come from the 5×5 and the 7×7 filtering unit respectively as shown by the two green arrows.

3.1.2 Multi-Scale Object Detection Scheme

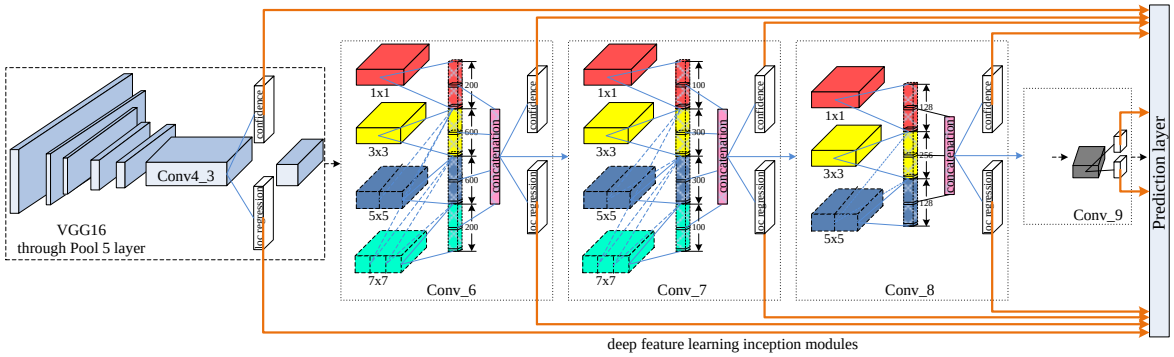

Figure 3.3: **The architecture of MDFN.** The proposed deep feature learning inception modules are introduced in the layers of Conv_6, Conv_7, Conv_8 (and Conv_9 in MDFN_I2). Each one employs filters of multiple sizes, which are represented by the red, yellow, blue and green boxes. The white boxes refer to classification and localization regression layers for the jointly learning of multi-scale detection (99).

Single shot multi-box detector (SSD) (99) is a popular and effective object detection model.

One of the key techniques proposed in SSD is multi-box matching and sifting, which discretizes the output space of bounding boxes into a group of default boxes with various aspect ratios and scales per feature map location (99). In the test process, SSD produces scores for the presence of every object category in each default box and generates several choices of the bounding boxes to better match the object shape. Specifically, the offsets relative to the default boxes and the scores of every classes, in each feature map cell, indicating the existence of object class instance in each box are predicted.

Inspired by SSD, we propose a multi-scale object detection scheme. In our model, given k boxes to each given location, calculates c class scores and four offsets of the four vertexes of each box relative to the default box. This ends up with a total of $k(c+4)$ filters serving for each location inside the feature map. Thus, the number of outputs should be $k(c+4)mn$ for each feature map with the dimension of $m \times n$. It has been verified in (99) that using various default box shapes would facilitate the task of predicting boxes for the single-shot network, which increases the accuracy of object localization and classification. We adopt this multi-box technique as the first property of our multi-scale scheme.

MDFN combines feature maps with different resolutions from both shallow and deep layers of the network, as shown in Figure 3.3. Our deep feature learning inception modules are applied in the four consecutive high-level layer units so that the extracted abstract information from each unit will be naturally built in the current layer outputs. These four layer units transmit their outputs directly to the final prediction layer. From the perspective of training, these shortened connections make the input and output ends of the network closer to each other, which benefits the model training (66). The direct connection between the high-level layers and the final prediction layer alleviates the problem of vanishing gradient and strengthens the feature propagation (66). This process is the second property of our multi-scale scheme.

Besides, we use the multi-scale filters to active the receptive fields of various sizes, aimming to enhance the extraction of the semantic and contextual information from high-level layers. As in most networks, feature maps would be downsampled gradually with the increase of depth, our

incepted deep feature learning modules will have less computational burden as the resolutions of feature maps in high-level part are much smaller than those in earlier layers. This is the third property of the proposed multi-scale scheme.

To quantitatively analysize the influence from inception depth, we propose two deep feature learning network architectures, referred as MDFN-I1 and MDFN-I2 respectively. we introduce the proposed information cubic inception modules into the first two layers, layer *conv*_6 and layer *conv*_7, in both the two MDFN models, of which the output feature maps are with the resolutions of 19×19, 10×10 respectively. Thus, considering the input feature maps' sizes of the next two layers are already relatively small, we introduce the information square module both into the latter two high-level layers, layer *conv*_8 and layer *conv*_9 in MDFN-I2 and only introduce information square module into layer *conv*_8 in MDFN-I1. We will compare the performances of the two models and analyze the impact on the deep feature learning ability. The specific configuration of inception modules are shown in Table 3.1.

<table>
<tr><td>Model</td><td colspan="4">MDFN-I1</td><td colspan="4">MDFN-I2</td></tr>
<tr><td rowspan="4">Conv_6</td><td colspan="4">1×1,200</td><td colspan="4">1×1,200</td></tr>
<tr><td rowspan="3">1×1,200</td><td rowspan="3">3×3,200</td><td>3×3,200</td><td>3×3,200</td><td rowspan="3">1×1,200</td><td rowspan="3">3×3,200</td><td>3×3,200</td><td>3×3,200</td></tr>
<tr><td>3×3,200</td><td>3×3,200</td><td>3×3,200</td></tr>
<tr><td>3×3,400</td><td>3×3,200</td><td>3×3,400</td><td>3×3,200</td></tr>
<tr><td rowspan="4">Conv_7</td><td colspan="4">1×1,100</td><td colspan="4">1×1,100</td></tr>
<tr><td rowspan="3">1×1,100</td><td rowspan="3">3×3,100</td><td>3×3,100</td><td>3×3,100</td><td rowspan="3">1×1,100</td><td rowspan="3">3×3,100</td><td>3×3,100</td><td>3×3,100</td></tr>
<tr><td>3×3,100</td><td>3×3,100</td><td>3×3,100</td></tr>
<tr><td>3×3,200</td><td>3×3,100</td><td>3×3,200</td><td>3×3,100</td></tr>
<tr><td rowspan="3">Conv_8</td><td colspan="4">1×1,128</td><td colspan="4">1×1,128</td></tr>
<tr><td>1×1,128</td><td>3×3,128</td><td colspan="2">3×3,128</td><td>1×1,128</td><td>3×3,128</td><td colspan="2">3×3,128</td></tr>
<tr><td colspan="2">3×3,128</td><td colspan="2">3×3,128</td></tr>
<tr><td rowspan="3">Conv_9</td><td colspan="4">1×1,128</td><td colspan="4">1×1,128</td></tr>
<tr><td colspan="4" rowspan="2">3×3,256</td><td rowspan="2">1×1,128</td><td rowspan="2">3×3,128</td><td colspan="2">3×3,128</td></tr>
<tr><td colspan="2">3×3,128</td></tr>
</table>

Table 3.1: Layer structure of deep inception module layout in MDFN-I1 and MDFN-I2

3.1.3 Experimental Evaluations

We employ the VGG-16 pre-trained on ImageNet as the base network of our proposed MDFN models. Then the models are fine-tuned on target dataset: KITTI, PASCAL VOC2007 or COCO.

The training was conducted on a computing cluster environment. The proposed MDFN employs the stochastic gradient descent (SGD) algorithm for training (14; 194). Due to the limit of GPU memory, our models are trained with the mini-batch size of 16 on KITTI and 32 on PASCAL VOC and COCO. The momentum is set to 0.9, and the weight decay is 0.0005 for all the datasets, which are identical to the training approach of Liu *et al*. (99). The overall number of training iterations is set to 120,000 for KITTI and PASCAL VOC and 400000 for COCO. We maintain a constant learning rate decay factor, which multiplies the current learning rate by 0.1 at 80,000 and 100,000 iterations for KITTI and PASCAL VOC, and at 280000, 360000 for COCO. MDFN-I1 and MDFN-I2 models adopt the learning rate of 0.0006, 0.0007 respectively on KITTI and COCO, and 0.0008, 0.0007 respectively on Pascal VOC2007.

MDFN matches the default boxes to any ground truth box with the Jaccard overlap higher than the threshold of 0.5. MDFN imposes the set of aspect ratios for default boxes as {1,2,3,1/2,1/3}. We minimize the joint localization loss by smooth L1 loss (47) and confidence loss by Softmax loss, shown as below.

$$L(x,c,l,g) = \frac{1}{N}(L_{conf} + \alpha L_{loc}) \tag{3.11}$$

where N refers to the number of matched default boxes and the weight term α is set to 1 (99). L_{conf} and L_{loc} stand for the confidence loss and localization loss, respectively.

For data augmentation, we adopt the same method as the original SSD model (99). We do not use the recent random expansion augmentation trick used by the latest SSD related frameworks (41; 99).

We empirically demonstrate the effectiveness of MDFN on the prevailing KITTI (44), PASCAL VOC (38) and Microsoft COCO (97) benchmarks. We analyze the object detection accuracy

in terms of average precision (AP), and object detection efficiency in terms of speed and model sizes. We also perform a thorough comparison between the MDFN and the state-of-the-art models on these datasets. The proposed framework is implemented using Caffe (75), compiled with the cuDNN (25) computational kernels. The test speed is obtained under Titan XP GPU.

3.1.3.1 Dataset

KITTI: KITTI object detection dataset is designed for autonomous driving, which contains challenging objects like small and occluded cars, pedestrians and cyclists. KITTI for object detection contains 7,481 images for training and validation, and 7,518 images for testing, providing around 40,000 object labels classified as easy, moderate, and hard ones based on how much objects are occluded and truncated. Since the ground truth of the test set is not publicly available, we follow the way in (170; 165), randomly splitting the 7,481 training and validation images evenly into a training and a validation set. We evaluate the proposed MDFN models on the validation set and report the average precision (AP) on it at the three difficulty levels following the suggestion in (44; 170). For KITTI experiments, we scale all the input images to 1242$\times$375 and use the batch size of 16. Our models are trained to detect 3 categories of objects, including car (merged with motors), pedestrian, and cyclist. The thresholds for car, pedestrian, and cyclist are 70%, 50% and 50%, respectively. All methods shown in Table 3.2 are obtained with the same rules described above.

PASCAL VOC 2007: In VOC experiments, we follow the normal practice in the literature, the models are trained on the union of PASCAL VOC 2007 and 2012 trainval set (16,551 images) and tested on PASCAL VOC 2007 test set (4,952 images). We scale all the input images to 500$\times$500. Our models are trained to detect 20 categories of objects on VOC. The overlap threshold for each category in VOC is set to 0.5. All the methods listed in Table 3.6 follow the same rules as above except for the scale of input images.

COCO: Microsoft COCO (97) is a widely used visual recognition dataset focusing on full scene understanding. Objects in COCO contains multifarious scales and occlusion situations, where objects are smaller scaled than PASCAL VOC. We utilize the trainval35k (8) for training and follow the strategy in (99). Experiments are implemented on two image scales, 300×300 and 512×512. The results are shown on COCO test-dev2015 and are evaluated based on the COCO-style average precision (AP).

3.1.4 Detection Results on KITTI

Model	Car			Pedestrian			Cyclist			mAP
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	
RPN	82.91	77.83	66.25	83.31	68.39	62.56	56.36	46.36	42.77	-
SubCNN	**95.77**	86.64	74.07	**86.43**	69.95	64.03	74.92	59.13	55.03	-
MS-CNN	90.0	89.0	76.1	83.9	73.7	68.3	84.1	75.5	66.1	78.5
PNET	81.8	83.6	74.2	77.2	64.7	60.4	74.3	58.6	51.7	69.6
Pie	89.4	**89.2**	74.2	84.9	73.2	67.6	84.6	76.3	67.6	78.6
SqueezeDet	90.2	84.7	73.9	77.1	68.3	65.8	82.9	75.4	72.1	76.7
SqueezeDet+	90.4	87.1	78.9	81.4	71.3	68.5	**87.6**	80.3	78.1	80.4
VGG16 + ConvDet	93.5	88.1	79.2	77.9	69.1	65.1	85.2	78.4	75.2	79.1
ResNet50 + ConvDet	92.9	87.9	79.4	67.3	61.6	55.6	85.0	78.5	76.6	76.1
SSD	86.6	86.0	80.5	75.7	71.8	69.3	83.7	83.0	77.1	81.6
MDFN-I1 (ours)	88.5	87.7	**80.7**	77.2	<u>74.6</u>	<u>72.4</u>	86.5	**86.2**	**83.5**	**83.9**
MDFN-I2 (ours)	87.9	87.1	<u>80.5</u>	77.5	**74.7**	**73.0**	86.0	<u>85.8</u>	<u>79.5</u>	<u>83.8</u>

Table 3.2: Average precision(%) on KITTI validation set. The best and second best results are highlighted in bold-face and underline fonts, respectively.

Average Precision: Table 3.2 shows the average precision (AP) of object detection on KITTI from the proposed frameworks as well as the 10 state-of-the-art models, including RPN (47; 133), SubCNN (170), MS-CNN (16), PNET (165), Pie (165), SqueezeDet (165), SqueezeDet+ (165), VGG16 + ConvDet (165), ResNet50 + ConvDet (165) and SSD (99). From Table 3.2, the proposed MDFN-I1 and MDFN-I2 networks obtain the significant improvements in terms of AP, especially for the detection of pedestrian and cyclist. It is noticeable that MDFN models perform the best in detecting objects that belong to moderate and hard levels for all the three categories, and the increased performance for the moderate and hard objects contributes to the best final mAP. The

average precision for pedestrian by MDFN surpasses the SqueezeDet+ by over 4%, and the AP for cyclist exceeds the SqueezeDet+ by over 5%. Based on the above experiment, it is evident that MDFN performs better in the detection of small and occluded objects in cluttered scenes.

Figure 3.7 shows some detection examples of SSD, MDFN-I1, and MDFN-I2 on KITTI dataset. There are four sets of images from four different scenes. In each set, the top image is the original image and the other three, from top to bottom, represent the results of SSD, MDFN-I1 and MDFN-I2, respectively. These four examples demonstrate the superior ability of MDFN in detecting small and occluded objects in a visualized way.

Performance under Multiple IoU: We adopt mAP with different IoU thresholds for further evaluation. In Table 3.3, Table 3.4 and Table 3.5, we provide the performances of SSD and MDFN models with IoU from 0.5 to 0.8 (in steps of 0.05) for Car, Pedestrian and Cyclist, respectively. It is evident that MDFN-I2 has significant advantage when the IoUs are higher than 0.65 for Cyclist and Pedestrian. This experiment further demonstrates that the proposed MDFN models have more accurate and robust detection performance.

Methods	**Network**	**0.5**	**0.55**	**0.6**	**0.65**	**0.7**	**0.75**	**0.8**
SSD	VGG	89.6	89.5	89.4	89.0	87.4	80.3	78.0
MDFN-I1	VGG	90.0	90.0	89.8	**89.6**	**88.6**	**80.4**	78.5
MDFN-I2	VGG	90.0	90.0	**89.9**	89.5	88.5	80.3	78.5

Table 3.3: Mean average precision on KITTI Car validation set for different IoU thresholds.

Methods	**Network**	**0.5**	**0.55**	**0.6**	**0.65**	**0.7**	**0.75**	**0.8**
SSD	VGG	74.8	72.4	66.1	60.8	51.8	38.2	26.7
MDFN-I1	VGG	76.9	**75.3**	70.2	64.5	54.8	43.4	26.4
MDFN-I2	VGG	**77.2**	75.2	70.2	**64.8**	**55.4**	**44.1**	25.9

Table 3.4: Mean average precision on KITTI Pedestrian validation set for different IoU thresholds.

Methods	Network	0.5	0.55	0.6	0.65	0.7	0.75	0.8
SSD	VGG	82.6	81.7	77.6	76.1	68.8	61.9	46.4
MDFN-I1	VGG	**86.2**	**85.8**	**82.7**	78.4	77.4	66.7	53.8
MDFN-I2	VGG	85.8	84.7	79.1	**78.8**	**77.5**	**68.0**	**54.5**

Table 3.5: Mean average precision on KITTI Cyclist validation set for different IoU thresholds.

3.1.5 Detection Results on PASCAL VOC Dataset

Table 3.6 shows the average precision of detection on the PASCAL VOC 2007 test benchmark. The leaderboard provides current state-of-the-art detection results, including Faster I (133), Faster II (59), ION (8), MR-CNN (46), R-FCN (29), YOLOv2 352×352 (129), SSD300* (99), SSD 321 (41), SSD512* (99), SSD300 (99), SSD500 (99), YOLOv2 544×544 (129), CC-Net (118) and BlitzNet512(s8) (37). From this result we can see that the MDFN models obtain leading performance in terms of mAP and achieve top average precision in the object class of train. MDFN models are the only ones that exceed 88% in the detection of trains. MDFN-I1 ranks the second in detecting plant. The proposed models also achieve very competing performance, though not the best, in other categories. Please note that some methods that outperform ours adopt very deep ResNet as their base network. Figure 3.4 shows some comparative detection examples from the VOC dataset. It can be seen that the MDFN models perform better in the complicated scenes.

Model	Network	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster I	VGG	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster II	Residual-101	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
ION	VGG	75.6	79.2	83.1	77.6	65.6	54.9	85.4	85.1	87.0	54.4	80.6	73.8	85.3	82.2	82.2	74.4	47.1	75.8	72.7	84.2	80.4
MR-CNN	VGG	78.2	80.3	84.1	78.5	70.8	68.5	88.0	85.9	87.8	60.3	80.52	73.7	87.2	86.5	85.0	76.4	48.5	76.3	75.5	85.0	81.0
R-FCN	Residual-101	80.5	79.9	87.2	81.5	72.0	69.8	86.8	88.5	89.8	67.0	88.1	74.5	89.8	90.6	79.9	81.2	53.7	81.8	81.5	85.9	79.9
YOLOv2 352×352	Darknet	73.7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SSD300*	VGG	77.5	79.5	83.9	76.0	69.6	50.5	87.0	85.7	88.1	60.3	81.5	77.0	86.1	87.5	83.97	79.4	52.3	77.9	79.5	87.6	76.8
SSD512*	VGG	79.5	84.8	85.1	81.5	73.0	57.8	87.8	88.3	87.4	63.5	85.4	73.2	86.2	86.7	83.9	82.5	55.6	81.7	79.0	86.6	80.0
SSD300	VGG	72.1	75.2	79.8	70.5	62.5	41.3	81.1	80.8	86.4	51.5	74.3	72.3	83.5	84.6	80.6	74.5	46.0	71.4	73.8	83.0	69.1
SSD500	VGG	75.1	79.8	79.5	74.5	63.4	51.9	84.9	85.6	87.2	56.6	80.1	70.0	85.4	84.9	80.9	78.2	49.0	78.4	72.4	84.6	75.5
SSD321	Residual-101	74.8	76.0	84.9	74.6	62.4	44.8	84.9	82.9	86.2	57.6	79.9	71.2	86.2	87.4	83.4	77.0	45.5	74.1	75.9	86.1	75.4
MDFN-I1-321 (ours)	Residual-101	75.9	76.8	83.5	74.7	65.8	46.5	85.2	83.7	88.2	59.9	78.3	74.3	86.7	87.4	83.8	78.1	47.4	76.1	81.0	86.2	74.3
MDFN-I2-321 (ours)	Residual-101	77.0	78.0	86.0	78.0	67.5	49.4	86.2	83.8	87.6	59.7	80.8	76.6	86.8	87.5	85.0	78.5	49.6	75.5	80.3	86.3	76.3
YOLOv2 544×544	Darknet	78.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CC-Net	CC-Net	80.4	83.0	85.8	80.0	73.4	64.6	88.3	88.3	89.2	63.2	86.0	76.8	87.6	88.2	83.4	84.1	54.9	83.7	77.7	86.0	83.6
BlitzNet512(s8)	ResNet-50	80.7	87.7	85.4	83.6	73.3	58.5	86.6	87.9	88.5	63.7	87.3	77.6	87.3	88.1	86.2	81.3	57.1	84.9	79.8	87.9	81.5
MDFN-I1 (ours)	VGG	79.3	81.2	87.0	79.2	72.3	57.0	87.3	87.1	87.5	63.1	84.2	76.7	87.6	88.8	85.6	81.0	56.0	80.4	79.9	88.0	77.0
MDFN-I2 (ours)	VGG	78.3	82.5	85.9	78.0	70.5	54.0	87.9	87.4	88.6	60.3	82.6	73.7	86.7	87.5	85.0	80.6	52.3	77.9	80.6	88.1	76.6

Table 3.6: PASCAL VOC2007 test detection results.

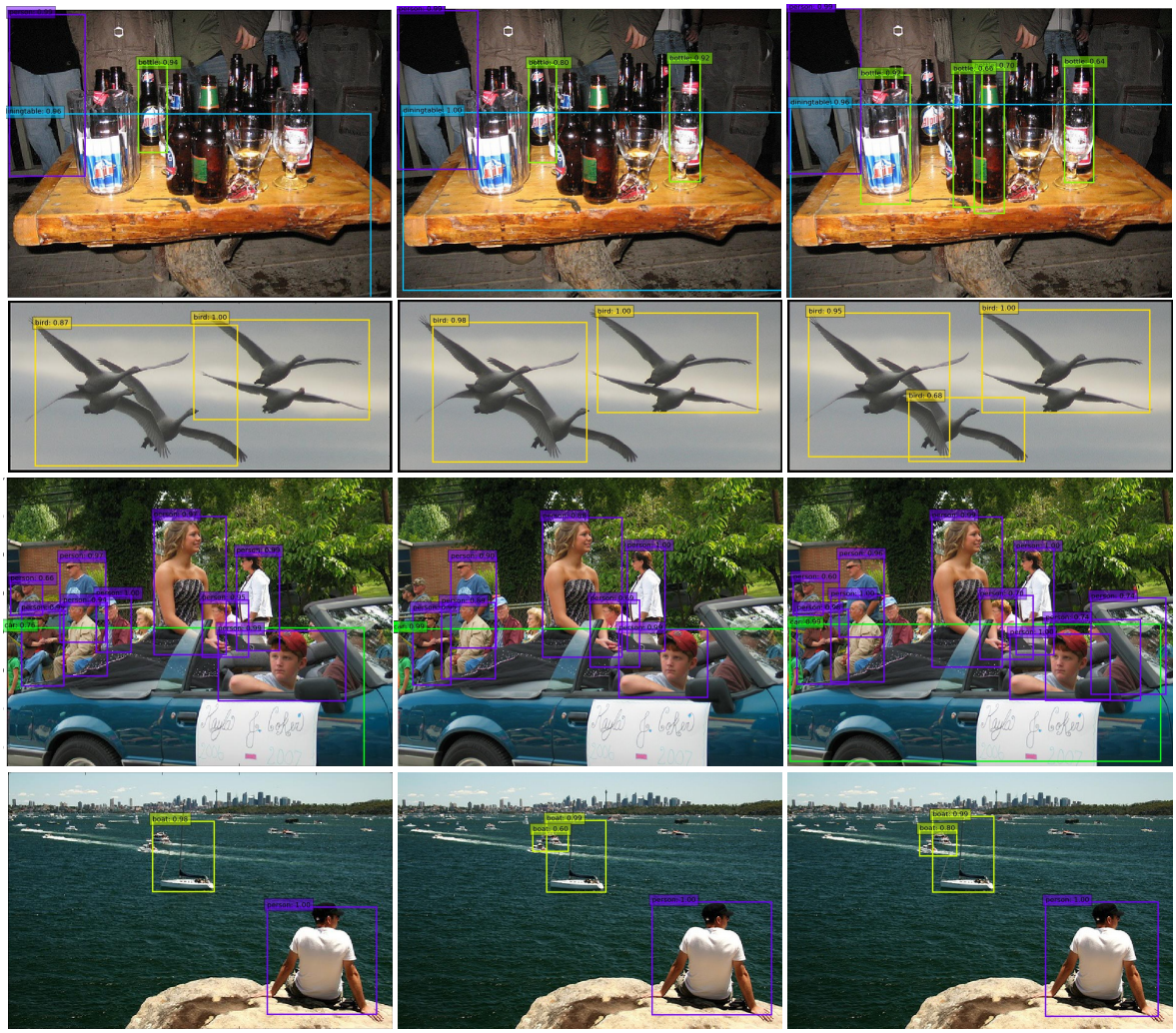

Figure 3.4: **Four sets of comparative detection examples of SSD and MDFN models on VOC2007 dataset.** In each set, the three images, from left to right, represent the results from SSD, MDFN-I1 and MDFN-I2, respectively.

3.1.6 Detection Results on COCO

The average precision of detection on COCO benchmark is shown in Table 3.7, where comparative results of mainstream detection models are provided as well, including Faster I (133; 99), Faster II (99), YOLOv2 (129), SSD300 (99) and SSD512 (99). The proposed MDFN-I2 model with the image resolution of 512 leads the board with the highest average precision under the three criteria of AP, AP_{50} and AP_{75}, where AP refers to the average precision over 10 IoU levels on 80 categories (AP@[.50:.05:.95]: start from 0.5 to 0.95 with a step size of 0.05). It is clear that on

both image scales of 300 and 512, MDFN obtains consistent better performance in comparison with SSD and other counterparts. The results of SSD and MDFN support the viewpoint that deep features are semantically abstract and suitable for extracting global visual primitives. Specifically, MDFN-I2 performs better than MDFN-I1, due to the deeper multi-scale feature learning. This is more obvious for higher IoU threshold, like the significant improvement of 4.6% from SSD300 to MDFN-I2-300 at 0.75 IoU compared with the 2.2% improvement at 0.5 IoU. For image scale of 500, MDFN show the same advantage. Such results agreed with the theoretical analysis in Section III and match the performance on the other two benchmarks.

Method	**Data**	**Backbone**	**AP**	**AP$_{50}$**	**AP$_{75}$**
Faster I	trainval	VGG	21.9	42.7	-
Faster II	trainval	ResNet-101	24.2	45.3	23.5
YOLOv2	trainval35k	DarkNet-19	21.6	44.0	19.2
SSD300	trainval35k	VGG	23.2	41.2	23.4
MDFN-I1-300 (ours)	trainval35k	VGG	26.3	42.9	26.9
MDFN-I2-300 (ours)	trianval35k	VGG	27.1	43.4	28.0
SSD512	trainval35k	VGG	26.8	46.5	27.8
MDFN-I1-512 (ours)	trainval35k	VGG	28.9	47.6	29.9
MDFN-I2-512 (ours)	trainval35k	VGG	29.6	48.1	30.8

Table 3.7: Detection results on COCO test-dev

3.1.7 Efficiency Discussion

In Table 3.8, we show a comparison of the inference time on KITTI, VOC2007 and COCO datasets. The compared frameworks include SSD (99), Faster I (133) with VGG-16, Faster II (59) with Residual-101, R-FCN (29), YOLOv2 (129), SSD300* (99), SSD512* (99), and SSD321 (41). For KITTI and COCO, although the introduction of deep feature learning modules brings around 10% increase of the number of parameters, the running speed of the MDFN models only decrease less than 4% and the Flops only increase around 2%. As shown in Figure 3.5, MDFN makes a good trade-off between the accuracy and speed on all the three benchmarks, where notations A,B and C form the similar pattern like a triangle on each benchmark and B,C are not far from A along

Method	Data	Resolution	network	#Params	GPU	FPS	FLOPS
SSD	KITTI	1242×375	VGG-16	24.0M	Titan Xp	30	157.4G
MDFN-I1	KITTI	1242×375	VGG-16	26.8M	Titan Xp	28	158.2G
MDFN-I2	KITTI	1242×375	VGG-16	27.0M	Titan Xp	27	158.3G
Faster I	VOC2007	1000×600	VGG-16	144.8M	Titan X	7	184.0G
Faster II	VOC2007	1000×600	Residual-101	734.7M	K40	2.4	93.0G
R-FCN	VOC2007	1000×600	Residual-101	-	Titan X	9	-
YOLOv2	VOC2007	352×352	Darknet	21.8M	Titan X	81	977.9M
YOLOv2	VOC2007	544×544	Darknet	21.8M	Titan X	40	2.3G
SSD300*	VOC2007	300×300	VGG-16	-	Titan X	46	-
SSD512*	VOC2007	512×512	VGG-16	-	Titan X	19	-
SSD300	VOC2007	300×300	VGG-16	26.3M	Titan Xp	84	31.4G
SSD321	VOC2007	321×321	Residual-101	52.7M	Titan Xp	30	22.1G
MDFN-I1-321	VOC2007	321×321	Residual-101	61.6M	Titan Xp	26	22.55G
MDFN-I2-321	VOC2007	321×321	Residual-101	64.2M	Titan Xp	26	22.56G
SSD	VOC2007	500×500	VGG-16	26.3M	Titan Xp	45	87.2G
MDFN-I1	VOC2007	500×500	VGG-16	30.8M	Titan Xp	39	87.9G
MDFN-I2	VOC2007	500×500	VGG-16	31.1M	Titan Xp	38	88.0G
SSD	COCO	300×300	VGG-16	34.3M	Titan Xp	75	34.4G
MDFN-I1	COCO	300×300	VGG-16	44.6M	Titan Xp	60	35.1G
MDFN-I2	COCO	300×300	VGG-16	45.6M	Titan Xp	58	35.1G
SSD	COCO	500×500	VGG-16	34.3M	Titan Xp	41	98.7G
MDFN-I1	COCO	500×500	VGG-16	44.6M	Titan Xp	35	100.4G
MDFN-I2	COCO	500×500	VGG-16	45.6M	Titan Xp	35	100.6G

Table 3.8: Comparison of inference time on KITTI, VOC2007 and COCO test datasets.

the speed axis, indicating the robustness and stability of MDFN and the prominent speed and accuracy trade-off. This supports our claim in Section III that the processing towards deep features contributes less computational load compared with the processing for feature maps from earlier layers. Therefore, enhancing the learning ability for deep features is definitely a high-productive choice.

From the performance on the VOC2007 dataset, the base network plays a key role in determining the running speed. Models with Residual-101 as base network consistently run much slower than those with VGG-16 base. From Table 3.6 and Table 3.8, the MDFN models significantly outperform those which do not adopt Residual-101, while the mAP of MDFN is very closed to those with Residual-101. The results demonstrate that, by introducing deep feature learning, MDFN has achieved a better balance between the higher detection accuracy and more efficient operational capability.

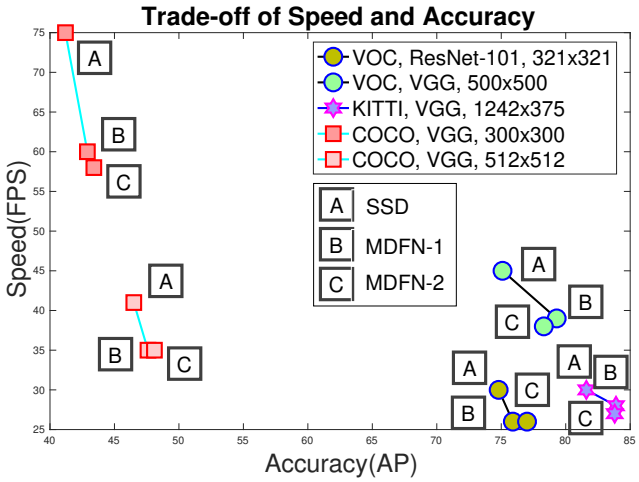

Figure 3.5: The trade-off between the accuracy and speed.

3.1.8 Limitation

Here different network depth is defined as the maximum layer depth that deep multi-scale features are extracted from, where the deep layers refer to those after the base network. For MDFN-I1, its multi-scale feature depth is 3 and for MDFN-I2, it is 4. Theoretically, MDFN-I2 is supposed to enhance the ability of feature expression and scene understanding. However, according to the mAP results on both KTTII and VOC2007 datasets, MDFN-I2 does not surpass MDFN-I1. While if we increase the threshold of IoU, MDFN-I2 shows its advantage of higher accuracy, which can be evidenced especially from Table 3.4 and Table 3.5.

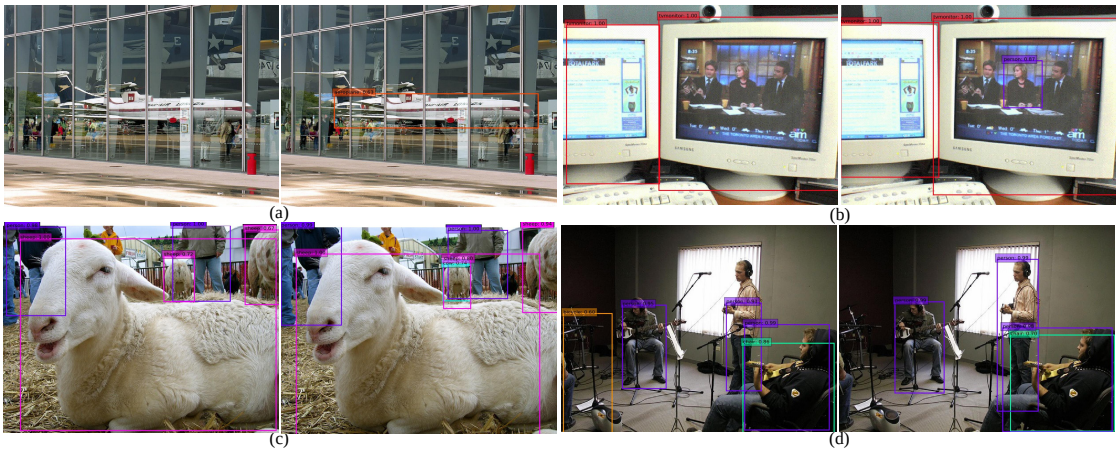

Figure 3.6: **Comparative detection examples by the two MDFN models.** In each set, the left and right images refer to the detection results of MDFN-I1 and MDFN-I2 respectively.

From Figure 3.6 (a) and (b), MDFN-I2 is capable of detecting the reflected objects, like the airplane in (a) and the displayed lady on the screen in (b). This is one reason that would lower the final mAP. From Figure 3.6 (c), MDFN-I2 yields multiple choices for some uncertain objects like the sheep at the back, which was detected as sheep or cow. This also leads to some decrease of mAP, which may be due to the overfitting. Nevertheless, MDFN-I2, by considering more context information, brings about more accurate localization in most situations like the bandsman in (d), where MDFN-I2 locates his figure completely compared to the part localization given by MDFN-I1. As the limited space, we do not show other examples. But MDFN-I2 occasionally fails to detect the partially occluded person, like the man behind the flowering shrubs only showing his head, whose lower half is occluded by flowers, or a lady only showing her head. This leak detection is more likely for MDFN-I2 which considers the context information more than MDFN-I1. It reveals the limited generalization ability.

3.1.9 Conclusion

In this work, we propose a novel multi-scale deep feature learning convolution neural network (MDFN) for object detection. It makes full use of highly abstract along with abundant semantic and contextual expression of deep features by integrating the proposed deep feature learning inception modules into the high-level layers of the network. Extensive experiments show that MDFN achieves more accurate localization and classification results on general object detection task (VOC), autonomous driving task (KITTI) and full scene understanding task (COCO), resulting in consistent and robust semantic representation. To our best knowledge, MDFN is the first single shot object detector that specifically focuses on deep feature learning. The proposed approach advances the state-of-the-art techniques in object detection and classification.

Figure 3.7: **Four sets of comparative detection examples of SSD and MDFN on KITTI dataset.** In each set, the four images, from top to bottom, represent the original image and the results from SSD, MDFN-I1 and MDFN-I2 respectively.

Chapter 4

Location-Aware Box Reasoning for Anchor-Based Single-Shot Object Detection

4.1 Introduction

Deep networks have been dramatically driving the progress of computer vision, bringing out a series of popular models for different vision tasks (192)(179), like image classification (18)(168), object detection (180)(92), crowd counting (139), depth estimation (60), and image translation (177). Object detection plays an important role and serves as a prerequisite for numerous computer vision applications, such as instance segmentation, face recognition, autonomous driving, and video analysis (61; 106; 11; 57; 69). In recent years, the performance of object detectors has been dramatically improved due to the advancement of deep network structure, well-annotated datasets, and effective optimization algorithms (96; 190).

In this paper, we aim at single-shot object detectors that yield a better trade-off between accuracy and speed, indicating a trend for future frameworks (99; 105). We reveal the problem of an inadequate quality criterion for anchor-based bounding box candidates, which is very important for model optimization and detection evaluation. The reason lies in that the quality of bounding boxes should reflect both the spatial location accuracy and the classification probability. While as far as we know, in current deep learning-based object detection pipelines, the scores of the bound-

Figure 4.1: Demonstrative detection results on MS COCO (97). The predicted bounding boxes have high classification scores while the localization is misplaced or interceptive. The left two images show misplaced cases in which the zebra is located by a much larger region, and the car is not an actual object. The airplanes in the right two images are partially located, while all of them have high confidence scores. Our method predicts the spatial relation between box proposals and their possible targets so that the interception and misplacement can be minimized.

ing boxes are shared with box-level classification confidence, which is predicted on the proposed features by the classifier. Most importantly, we cannot obtain the location assessment during the inference stage due to the deficiency of labels. It is insufficient to use the classification confidence to measure the bounding box quality since it only serves for distinguishing the semantic categories of proposals, while it is not aware of the assessment towards localization accuracy. The misalignment between classification confidence and bounding box quality is illustrated in Figure 4.1, from which we can see that, although the object instances obtain a high classification confidence score, the box-level localization is not unanimously accurate. If a predicted object is not scored properly, it might be mistaken as a false positive or negative, affecting the NMS process and leading to a decrease of average precision (AP). It is evident that the lack of effective scoring metrics towards the localization quality tends to impair the evaluation.

In this work, we focus on a more reasonable and effective scoring metric for anchor-based bounding box proposals. Different from most previous works that either pursue high-quality classi-

fication boxes or focus on score correction working on two-stage object detectors, we demonstrate that there is room of further improvement for popular anchor-based single-shot object detection models by introducing calibrated quality scores that take the location confidence into consideration. Compared with RPN-based frameworks, single-shot object detectors highly depend on qualified box proposals as there is no pre-sift scheme. As anchor-based methods, they are sensitive to location information, which brings challenges for box sifting. To solve this issue, we propose a calibrated quality score (**CQS**) for each box proposal to realize the location awareness. The localization score indicates the spatial relation to its most-probable target ground truth and ranks the proposals based on the calibrated quality score rather than the classification confidence.

In anchor-based single-shot object detection, the bounding box proposals are regressed by the space shift relative to the anchors. Thus, the spatial relationship of an anchor and an object ground truth depicts an expectation or estimation of the location relationship between the corresponding box proposal and the target, as depicted in Figure 4.2. Inspired by the Average Precision (AP) metric of object detection using pixel-level Intersection-over-Union (IoU) between the predicted bounding box and its ground truth to describe the quality of predictions, we propose a network module to learn the IoU between the anchors and the ground truth directly. For the convenience of discussion, we call it **AIoU**. We adopt the proposed locscore to learn this AIoU during the training time, and when given the locscore in the test phase, the quality of bounding boxes is reevaluated by integrating locscore into the classification confidence so that the reasoned box proposals are aware of both the location information and the semantic categories.

Compared with localization and classification regressions that take the ground truths from the labeled dataset, the learning for AIoU only needs to calculate the IoU between the anchors and the ground truths as a target, without further labeling the dataset. Within a detection model, we implement the locscore prediction network as the locscore head, which takes the feature outputs and the calculated AIoU as inputs, and is trained with a common regression loss. We implement object detection experiments with the proposed location-aware anchor-based box reasoning module on popular single-shot object detectors. The results demonstrate that our method can promote

the performance of object detection and yield consistent and robust detection results. The main contributions of our paper include:

C1. We propose a novel bounding box reasoning method that is aware of the spatial relationship between the box proposals and the probable target ground truth. It is one of the first algorithms that address the issue caused by scoring bounding box proposals only by the classification probability.

C2. This is the first location-aware detection framework designed for the single-shot networks that naturally take the pools of anchor-based box proposals as candidates, ensuring a one-shot learning fashion.

C3. The proposed plug-in locscore head can be integrated with any single-shot detection networks and regressed easily in an end-to-end fashion. By calibrating the detection quality with locscore, the bounding boxes can be penalized if it has high classification confidence while relatively poor localization accuracy.

C4. We demonstrate the effectiveness of the location-aware anchor-based box reasoning scheme through extensive experiments. By introducing the proposed calibrated quality score into the evaluation metric of box proposals, the detection performance is further improved.

4.2 Related Work

4.2.1 Object Detection

Multiclass object detection is a core task in the context of deep learning based computer vision projects, which is the joint work of the classification towards contents and the localization towards bounding boxes of instances. Most of these methods adopt the CNN (89) based bounding box and classification regressions, followed by a Non-Max Suppression (NMS) algorithm to sift best-qualified box proposals.

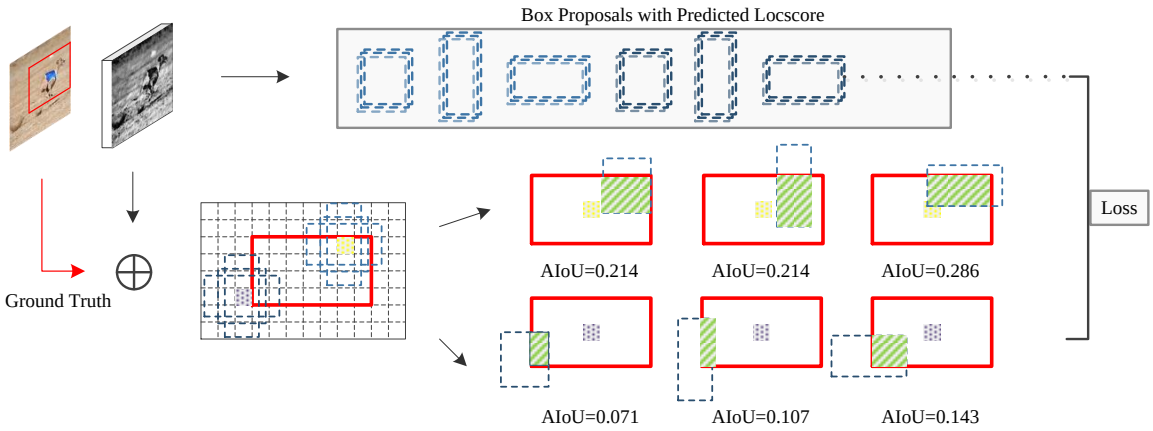

Figure 4.2: AIoU Definition and the Locscore illustration. AIoU as the target of the proposed locscore only needs the input image and its corresponding ground truth. In the fashion of convolution, we evaluate the default boxes with multiple scales at different anchor bases, as the yellow and purple color lumps shown in this figure. For each default box (represented by the blue dotted rectangular), in addition to predicting the shape offsets and the confidence scores as conventional detectors, we also predict the locscore which assesses the possibility of how close the object is to the ground truth. The locscore is learned towards the AIoU calculated by the anchor box and the ground truth, which is denoted by the red angular box in this figure. Specifically, the locscore of a box proposal is learned to match the AIoU between its corresponding anchor box and a certain ground truth box.

Bounding box regression was first introduced in R-NN (48). It enables regions of interest (ROI) to estimate the updated bounding boxes with the purpose of better matching the nearest object instance. Prior works, from Fast R-CNN, Faster R-CNN (133), R-FCN (29), to YOLO (128), SSD (99), RetinaNet (96), and RefineDet (189), have demonstrated that the detection task can be improved with multiple bounding box regression stages (46), flexible anchor matching (190), the increase of the number of anchors, and the enlargement of the input image resolution, including image pyramids (95). Among them, the most widely-used and efficient technique is the anchor-based multibox algorithm that can handle scale variation, one of the challenging problems for one-shot object detection. Anchor boxes are designed for discretizing the continuous space of all possible instance boxes into a finite number of boxes with predefined locations, scales, and aspect ratios (196). The created instance boxes are regressed to match the ground truth bounding boxes based on the Intersection-over-Union (IoU) overlap, by location shift at certain base anchor with the predefined locations.

However, there exist underlying limitations. On one hand, the quality of the proposed bounding boxes is only measured by the classification score, leading to the misalignment between the box score and box quality. Due to the unreliability of the box score, a proposal with higher IoU against the ground truth will be ranked with low priority if it obtains lower classification confidence. In this situation, the Average Precision (AP) can be degraded. On the other hand, compared with RPN (133), the anchor-based technique is more sensitive to box quality especially for the consistency of classification confidence and location accuracy since there is no pre-sift scheme for box proposals in the one-stage case.

4.2.2 Detection Scoring and Correction

The misalignment of the box score and actual quality has aroused much attention and several correction methods have been proposed in recent years. Tychsen-smith *et al.* (154) presented a Fitness NMS that corrects the detection score by learning the statistics of best matching detected bounding boxes with the ground truth as a corrective factor. It formulates box IoU statistics prediction as the classification task. It is specifically designed for Denet (153), which restrains its application to arbitrary object detection frameworks.

Jiang *et al*. (76) proposed a standalone IoU-Net which is based on a similar R-CNN structure with a proposal pre-sift scheme to predict IoU between the predicted boxes and the ground truths. It manually designs bounding box filtering as an addition to the data pool of box proposals. The IoU-guided NMS ranks bounding boxes by the predicted localization confidence rather than the conventional classification confidence. Cheng *et al*. adopted a separate network to correct the scores of samples by processing false-positive samples (23). SoftMax (12) proposed to use the overlap between two boxes to correct the low score box. Neumann *et al.* (116) proposed a relaxed softmax to predict the temperature scaling factor in standard softmax for safety-critical pedestrian detection. Both of the two approaches are designed for the two-stage R-FCN based models, relying on the clean proposal data pool. Wu *et al*. proposed the IoU-aware approach scores the location results while it is merely a RetinaNet based detector (166), not an arbitrary method.

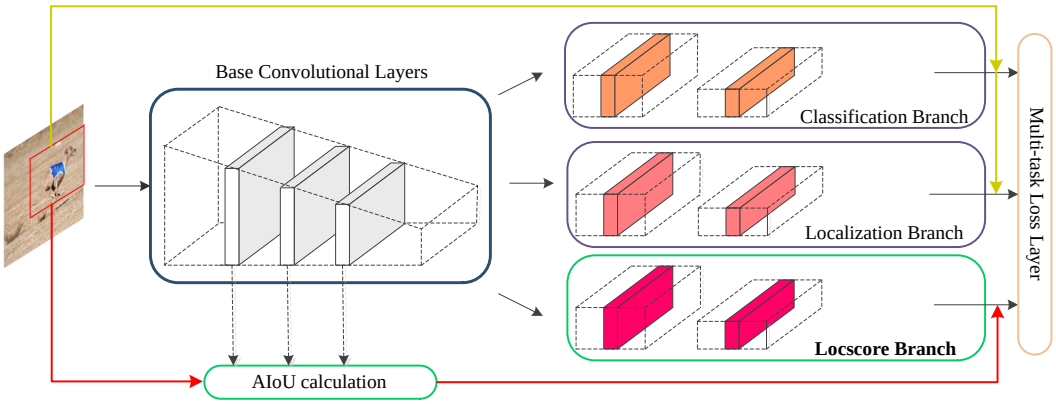

Figure 4.3: The network architecture of object detection with location-aware anchor-based reasoning. The input image is fed into the backbone network to generate feature maps with RoI information. The Locscore Branch is the standard component of the improved model. It takes the output features from the backbone network as inputs and provides a predictive locscore at the end, where its layer structure just follows the one of the classification branch or the localization branch.

Different from the above methods, this study focuses on the essence of the evaluation towards anchor-based box reasoning in single-shot frameworks. We assign each predicted box with a location score by taking aware of the spatial relation between its based anchor and the ground truth. The proposed approach takes both the classification confidence and the location accuracy into consideration to create a complete evaluation of instance box quality so as to narrow the gap between the box score and the actual quality. Furthermore, we build an independent regression branch in the single-shot object detection framework that learns the location confidence specifically and merges this information into the box quality evaluation metric of NMS so as to obtain a more reliable priority ranking.

4.3 Location-Aware Box Reasoning

4.3.1 Motivation

In current object detection frameworks, the classification and localization regressions are taken as two independent processes. The evaluation towards a detection hypothesis, the detected bounding box, is determined by the highest-ranked element in the classification scores. However, there exist

certain situations where the predicted box with a high classification score has low localization accuracy. This kind of hypothesis is harmful in most detection evaluation protocols, such as MS COCO. It is important that a detector can determine when the detection results are trust-worthy and when they are not. This motivates us to integrate the localization score by location-aware anchor-based reasoning for every predicted bounding box based on an anchor position.

Most previous methods do not consider location confidence as one of the evaluation factors that contribute to the box quality (154)(23) (12)(116) and the majority of them are designed as specific detectors or merely applied for two-stage detectors with the pre-sift scheme. Although IoUNet obtains competing results with the proposed IoU-guide NMS algorithm that takes localization confidence into consideration, it ranks the boxes only by the localization confidence which highly relies on the clean data pool of proposals produced by two-stage detection models. It is hard to apply it in a single-shot fashion. We propose the location-aware anchor-based box reasoning that focuses on arbitrary single-shot detectors for a better trade-off between accuracy and efficiency.

We instantiate the location-aware reasoning module by showing how to apply it to the anchor-based single-shot detectors. Without loss of generality, we apply the proposed module to the state-of-the-art RetinaNet and SSD with an additional Locscore head that learns the IoU between the anchor and ground truth, and demonstrates our design from the following aspects: 1) how to realize location awareness for anchors; 2) how to create the branch of location score in the network; and 3) how to generate location-aware anchor-based reasoning during inference time.

4.3.2 Location Awareness

From the perspective of conception, location awareness is simple. In anchor-based detectors, the introduction of location awareness supplements the evaluation towards the quality of the bounding boxes from the perspective of location accuracy. It is realized by learning the IoU between the anchors and ground truths, producing the localization scores.

Localization Score We begin with briefly reviewing the evaluation metric towards the bounding box proposals. Following the anchor-based detection, the proposals are created based on anchors with various scales at different positions on the feature maps. The network extracts features from the backbone and performs proposal classification and bounding box regressions respectively. The former one yields confidence scores regardless of the location reference, while the latter one regresses the space migration of candidates. Although the predicted bounding boxes with classification confidence and location prediction, the quality of the box candidates can only be evaluated by the confidence score, without localization assessment. This is due to the lack of ground truth as a reference during the inference stage. Thus, there is a gap between the current metrics and the actual need for evaluating the quality of box proposals.

We define $P(lc|a_j)$ as the localization score by learning the pixel-level IoU between bounding box a_j and any object ground truth b_j. We define it as "AIoU".

$$AIoU = IoU(a_j, b_j) \tag{4.1}$$

In ideal anchor-based detectors, the object is detected by three elements: anchor, box proposal, and object ground truth. Conventionally, we build the direct correspondence between the anchors and the boxes by regressing space shift, and the relations between the boxes and the object ground truths by matching features. However, there is no direct depiction of the relationship between the anchors and object ground truths, where an underlying location link exists. The introduction of the location score complements the relational structure of the three essential elements and further calibrates the quality criterion of bounding box proposals by the definition expressed as below.

$$S(a_j) = P(c|a_j) \cdot P(lc|a_j) \tag{4.2}$$

where $P(c|a_j)$ denotes the confidence score, and $S(a_j)$ is defined as the calibrated quality criterion of the bounding box proposals. Thus, $S(a_j)$ should work well on two tasks: indicating the right category that the box belongs to and regressing the IoU of the proposals and the foreground objects.

4.3.3 Localization Score Regression

Locscore Head Conventionally, classification and regression are two independent branches for all object categories. Without loss of generality, we introduce the Locscore Head as the third independent branch to predict the IoU between anchors and object ground truths. This head simply follows the layer structure of the other two existing heads so as to save the network characteristics and the advantages of the framework. Thus, it can be implemented as a plug-in module and be integrated with any arbitrary single-shot object detection model.

The Locscore Head receives the concatenation of features from the output layer of the network as its inputs. It predicts the localization score for each anchor box on the feature maps, which depicts the location relations between the box and the target ground truth. In anchor-based object detection, each predicted bounding box is created based on a certain anchor and regressed location migration. Thus, according to this correspondence among the three elements, each predicted bounding box proposal is corresponding to one anchor so that the box would be assigned a localization score indicating the maximum possibility that it is related to an object from the perspective of localization.

Based on the analysis above, since the Locscore Head shares the same concatenated features with the other two Heads, classification, and box regression, the predicted three elements have an inner congruent relationship. The Locscore Head, thus, can be taken as an independent regression branch and treated as an individual learning task.

We define the Locscore Loss to regress the Localization Score. It follows the loss definition for classification regression. Then the Locscore Head is integrated into an anchor-based object detection framework, and the whole network can be trained end-to-end. Specifically, we define the classification loss and box regression loss as L_{cl} and L_{bb}, respectively. In addition, we introduce locscore loss L_{lc} as another penalty item to the cost function, as shown below,

$$L^* = \lambda_1 * L_{cl} + \lambda_2 * L_{bb} + \lambda_3 * L_{lc} \tag{4.3}$$

where L^* denotes the final loss function. In all experiments, we adopt equal weights for the three loss items in consideration for stability, so that $\lambda_1 = \lambda_2 = \lambda_3 = 1$. The locscore loss forms a lower bound in the space localization, and by training, we further pull down this lower bound.

4.3.4 Box Reasoning during Inference

Calibrated Quality Score (CQS) We define and propose a calibrated quality score by introducing the localization confidence into the assessment of the predicted bounding box proposals. Thus, the quality score is disintegrated into two data spaces, where both classification confidence and location accuracy are taken into consideration, as shown in Equation 4.2. This CQS during inference time becomes a new criterion for the sifting of qualified candidates. It complements the defect of independent classification and location regressions that lead to the deficiency of localization reference during inference time.

Inference At the inference stage, the proposed candidates with coordinates, conferences, and localization scores are integrated into the non-maximum suppression (NMS) algorithm. Different from its classical counterparts, the NMS in our model does not rank the box proposals merely by classification probabilities in the first step. Instead, we use the CQS as the ranking criterion so as to push the box candidates to indicate the spatial relations with potential objects, in terms of the initial idea that the quality of bounding boxes is tightly correlated to both the spatial information and classification confidence. We assume the saved boxes after the above sifting are qualified candidates. But in order to weaken the sensitivity towards the less qualified proposals for the single-shot models, we adopt the confidence cluster (76) to further enhance the reliability of the sifted boxes by updating the confidence score S_i of box i with $S_i = max(S_i, S_j)$, where j indicates box j that is deleted by box i in NMS. Details are shown in Algorithm 1.

Specifically, suppose the network outputs N bounding boxes, the NMS firstly rank them by the proposed CQS , then we follow the same procedure to remove the candidate boxes which overlap each other over a threshold of $\varepsilon = 0.5$. At last, the top-k scored boxes are selected and fed into the

output head to generate multi-class boxes.

Algorithm 1 Location-Aware Box Reasoning. Classification confidence and localization score are independently regressed during the training time but the two values are taken as combined consideration during the inference time when evaluating the anchor-based box proposals.

Input: B, P_c, P_{lc}, ε.
B: set of anchor-based bounding box proposals.
P_c: classification confidence by mapping f_c.
P_{lc}: localization score by mapping f_{lc}.
ε: IoU threshold in NMS.
Output: D, set of detected boxes with classification confidence P_c.

$D \leftarrow \varnothing$
while $B \neq \varnothing$ **do**
    $S \leftarrow P_c \times P_{lc}$
    $b_m \leftarrow \arg\max_i S(b_i)$
    $S_m \leftarrow S(b_i)$
    $P_m \leftarrow P_c(b_i)$
    $B \leftarrow B \setminus b_m$
    for $b_j \in B$ **do**
        if IoU$(b_m, b_j) > \varepsilon$ **then**
            $B \leftarrow B \setminus b_j$
            if classification cluster **then**
                $P_m \leftarrow max(P_c(b_j), P_m)$.
    $D \leftarrow D \cup \{\langle b_m, P_m \rangle\}$

4.4 Experiments

We conduct experiments on the detection tasks of the MS COCO (97) and PASCAL VOC (39) datasets. MS COCO contains 80 object categories, we follow COCO 2017 settings, using the 115k images *train* split for training, 5k *validation* split for results analysis. The COCO results are reported by its evaluation metrics AP (Average Precision over IoU thresholds), including, AP@0.5 (IoU equals to 0.5), AP@0.75 (IoU equals to 0.75), AP (averaged on AP over IoU thresholds from 0.5-0.95 with a step size of 0.05), AP_S, AP_M, AP_L (AP at different scales of objects). In the VOC experiments, we follow the same practice as in the literature, the models are trained on the union of PASCAL VOC2007 and 2012 trainval set (16,551 images) and tested on PASCAL VOC 2007 test set (4952 images). The overlap threshold for each one of the 20 categories in VOC is set to

0.5.

4.4.1 Implementation Details

We adopt the consistent location-aware box reasoning (LAAR) framework for all experiments. We use the ResNet-50 based FPN network and VGG-16 based SSD network for ablation study, respectively. For ResNet-50 FPN, the input images are resized with a minimum 608px along the short axis and a maximum of 1024px along the long axis for both training and test. We train the network and choose the model at the epoch that yields the best performance. The learning rate is reduced by the Plateau strategy, the same as that of the original RetinaNet. For SSD, the input images are resized to 300×300 for both training and test, as the common rule in literature. The rest of all configurations are identical to the realization in (99). We train the network for 120,000 iterations and decrease the learning rate after 80,000 and 100,000 iterations. The optimizer for RetinaNet experiments is Adam with an initial learning rate of 0.00001, and for SSD experiments is SGD with momentum 0.9. In the test, all the results are evaluated by the NMS, where the top-100 score detection is retained for each image.

Learning Scenarios In order to identify the gains of locscore regression constraint and location-aware box reasoning respectively, we intentionally design independent learning scenarios where we do solo locscore regression constraint without quality score calibration, the complete LAAR detection framework with CQS, and the complete LAAR detection with CQS and confidence cluster. We list them as follows:

- *Independent Locscore Constraint (ILC):* We introduce the locscore regression during training time while do not consider the predicted locscore to calibrate the classification score during the inference period.
- *Locscore Constraint with CQS (LC):* We conduct complete location-aware box reasoning with the given detector, which means we add locscore regression constraint during training and introduce the calibrated quality score by the predicted locscore during the test time.

- *Locscore Constraint with CQS and classification cluster (LC + CS):* Based on LC, we introduce the classification cluster after calibrating quality score by the predicted locscore, as shown in Algorithm 1.

4.4.2 Ablation Studies

We evaluate the contribution of one important element to our location-aware box reasoning for object detection, the constraint brought by the Localization Score Regression.

Locscore constraint for better optimization Compared with conventional object detection frameworks, we introduce the additional constraint term in the loss function by doing the localization score regression. As far as we know, this is the first time to directly explore the relationship between the anchors and the ground truths in single-shot fashions. The proposed procedure explores the predefined prior information of anchor boxes and the ground truths from the perspective of spatial location. As is known, anchor-based fashion defines fixed anchor positions and their multiple-ratio variations on a feature map, which means for a certain image, there exist predefined location relations between the anchor boxes and the ground truths. We introduce this relation in the penalty function to help constrain the classification and localization regressions. Especially for the latter one during training, when the coordinates are learned in a deflected direction, there exists a correction by the locscore regression. This constraint results in better optimization as demonstrated in Table 4.1 and 4.2.

In Table 4.1, the ILC version leads the original RetinaNet in most cases. For $AP_{0.75}$, ILC improves the detection accuracy by a rough 1% than the original ResNet. Although the ILC falls behind AP_S, it leads by large margins in both AP_M and AP_L. In Table 4.2, we obtain consistent results. In this VOC setting, we list the AP results for all the 20 categories. The ILC shows better mAP than the original SSD and yields clearly higher AP for most categories. These results support the idea that the introduction of locscore regression yields effectively positive constraint towards the other two regressions. Thus, the locscore branch boosts the mutual promotion of classification

	R-Net	R-Net+ILC	R-Net+LC	R-Net+LC+CS
Backbone	R-50	R-50	R-50	R-50
AP	30.4 %	30.9 %	30.7 %	**30.9** %
$AP_{0.5}$	47.3%	**47.7** %	47.2%	47.4 %
$AP_{0.75}$	32.1 %	33.0 %	32.9%	**33.2** %
AP_S	**13.9**%	13.0 %	12.9 %	13.0 %
AP_M	33.1 %	**34.0** %	33.8%	33.9 %
AP_L	43.7%	44.1 %	44.2%	**44.3** %

Table 4.1: The mAP of RetinaNet on COCO val2017. R-50 indicates ResNet50 with FPN and R-Net refers to RetinaNet.

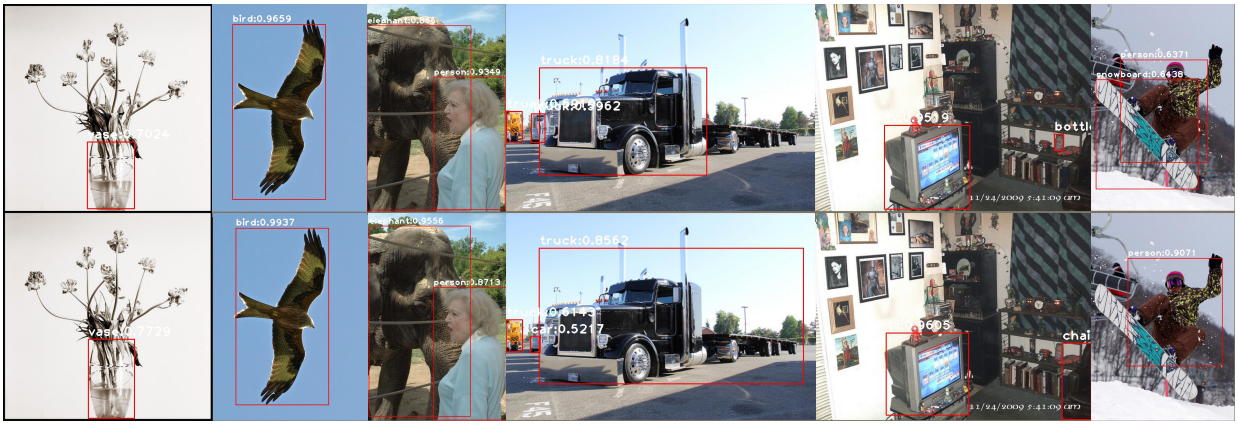

Figure 4.4: COCO cases compared between RetinaNet (1st row) and RetineNet+LC (2nd row), where LC version achieves higher localization accuracy.

and localization, helping to improve the situation when confidence score and localization accuracy are opposite, such as when confidence score is 0.6 and localization accuracy is 0.2 compared to that when confidence score is 0.2 and localization accuracy is 0.6, which guarantees the feasibility of the proposed approach.

4.4.3 Quantitative Results

We extensively evaluated the proposed method with two popular detectors, SSD and RetinaNet, on Pascal VOC 2007 and COCO val2017, respectively. We also compared the proposed algorithm with state-of-the-art methods, SoftMax (154) and IoUNet (76). Since these methods are confined to the application of R-CNN based two-stage frameworks, they are highly dependent on the pre-sift

	SSD	SSD+ILC	SSD+LC	SSD+LC+CS
Backbone	VGG-16	VGG-16	VGG-16	VGG-16
aeroplane	82.97 %	81.87 %	82.43 %	82.41 %
bicycle	84.18%	83.61 %	83.13%	83.18 %
bird	75.34 %	75.23 %	74.23%	75.56 %
boat	70.98%	70.35 %	70.62 %	70.71 %
bottle	50.44 %	51.97 %	51.62%	51.90 %
bus	84.29%	86.05 %	84.49%	86.03 %
car	86.32%	85.30 %	84.60%	85.53 %
cat	88.12%	88.36 %	87.45%	88.26 %
chair	61.54%	62.18 %	58.82%	62.02 %
cow	79.94%	83.03 %	82.28%	83.18 %
diningtable	77.12%	75.80 %	72.40%	76.20 %
dog	85.05%	84.31 %	82.38%	84.13 %
horse	87.60%	87.03 %	85.92%	87.68 %
motorbike	82.84%	82.95 %	81.83%	83.50 %
person	78.98%	79.09 %	77.71%	79.12 %
pottedplant	52.17%	51.94 %	50.27%	51.45 %
sheep	78.61%	77.40 %	75.40%	77.05 %
sofa	78.33%	80.29 %	77.49%	80.12%
train	87.88%	86.65 %	84.80%	87.71%
tvmonitor	76.33%	77.29 %	74.93%	77.10%
mAP	77.45%	77.53 %	76.14%	**77.64%**

Table 4.2: mAP of SSD on VOC2017. The category name indicates its corresponding AP result.

scheme by RPN, where the proposal pool could already be regarded as clean data. The proposed method aims at anchor-based single-shot detection models without the pre-sift scheme, where the anchor-based produced boxes can be regarded as rough candidates. We conduct experiments and verify that we can not directly introduce the above algorithms in the single-shot fashion as a comparison, like the one proposed by IoUNet by ranking the boxes using localization confidence. It leads to a sharp drop in mAP. It is understandable that the produced boxes by anchors are rough candidates that are unreliable for quality ranking. Therefore, these methods can not be directly applied to single-shot models.

To make a fair comparison, we integrate the core idea of IoU-guided NMS produced by IoUNet to merge the classification cluster into our algorithm and form the 'LC+CS', described in lines 11 and 12 of Algorithm 1. We then conduct comparison and show quantitative results in Table 4.1 and 4.2. From Table 4.1 we can see that, compared with RetinaNet, LC model achieves stable improvement in most cases. Specifically, LC obtains better AP, $AP_{0.75}$, AP_M, and AP_L. For $AP_{0.75}$, LC achieves an enhancement of 0.8%, and for AP_M, LC improves by a margin of 0.7%. We can

Figure 4.5: COCO cases compared between RetinaNet (1st row) and RetineNet+LC (2nd row), where LC version performs better for hard objects.

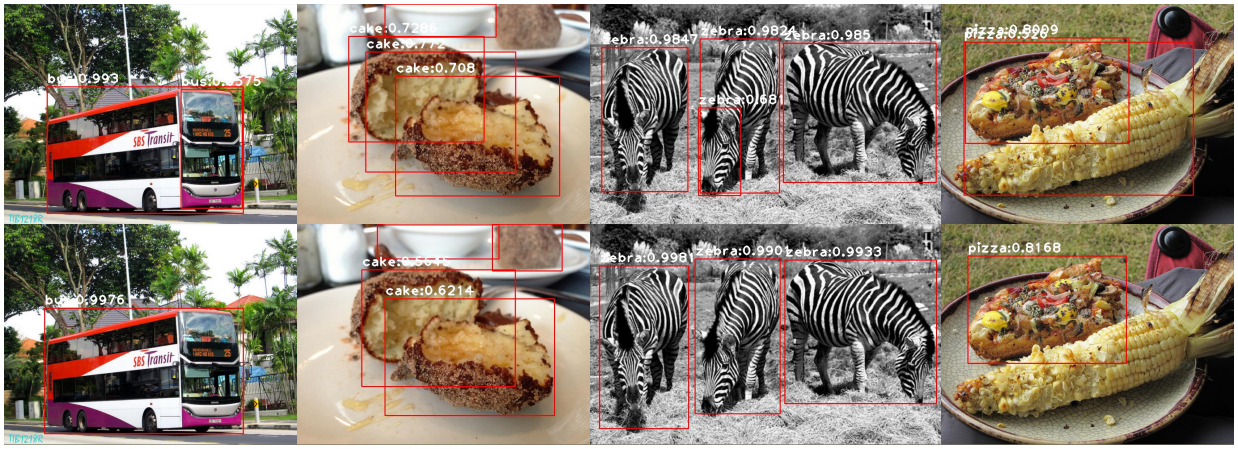

Figure 4.6: COCO cases compared between RetinaNet (1st row) and RetinaNet+LC (2nd row), where LC version has better filtering for low-quality boxes.

conclude that, for the detection accuracy with high request and objects with the most common sizes, our method exhibits clear advantages. In Table 4.2, the LC version does not lead the ranking although its corresponding ILC version performs better than the original SSD model. This reflects the fact that the anchor-based single-shot models are sensitive to location accuracy and the produced rough candidates have less reliable location outputs than the expectation. From Tables 4.1 and 4.2 we can see that 'LC+CS' achieves the best results in both experiments, with over 1% enhancement in some cases, such as in RetinaNet at $AP_{0.75}$. The results demonstrate that the introduction of classification clustering could compensate for the uncertainty caused by the location

outputs.

4.4.4 Comparison and Discussion

In this section, we first discuss the quality of the predicted bounding boxes and investigate the upper bound of the performance of the LAAR model, and analyze the benefits of locscore learning. Here, all the results are evaluated on COCO2017 validation set using RetinaNet and ReinaNet with the LC models.

4.4.4.1 Fitter and tighter bounding boxes

In Figure 4.4, It is evident that the LC model predicts higher-qualified boxes than RetinaNet, and most of the boxes are tighter. Specifically, the boxes for the vase, the eagle, and the truck, are more accurate, and the box for the chair shows better performance in occluded cases. This demonstrates that the introduction of locscore learning can improve the accuracy of bounding boxes, and help select the best proposals that have the maximum alignment between the quality score and box quality. Tighter boxes in practice can help clear up some current dilemmas in industry.

4.4.4.2 Better for hard objects

In Figure 4.5, it can be seen that the LC model is able to detect harder objects, like the small backpacks, the occluded cellphone, and the persons in the audience. These small or occluded objects are easy to be overlooked during the suppression process, as their classification scores can be small due to the deficiency of the effective features. While with the locscore, the detection score can be calibrated as these hard objects could probably have high localization scores if they are labeled. Thus, the actual detection accuracy can be raised by calibrating the box quality score especially when the original classification confidence is low.

4.4.4.3 Waiving low-quality boxes

In addition, from Figure 4.6, we can see the LC model is better at sifting out some low-qualified boxes, like the smaller bus box, the redundant cake, and zebra box. Similarly, this can be explained by the calibrated quality score. Although some boxes' classification scores are high, if their location scores are low, these boxes can still be regarded as low-quality objects, which could be disregarded in NMS.

4.5 Conclusion

This paper reveals the problem of object detection score as one of the primary limitations of current anchor-based single-shot object detectors. To address this issue, we have proposed the localization score (locscore) regression and location-aware box reasoning, where the classification score is aligned with the predicted locsocre so that the localization accuracy is taken into the assessment of the quality of the bounding box proposals, which has been overlooked in most popular object detection frameworks. Extensive experimental results show that the proposed approach can consistently improve the detector's performance to yield reliable bounding boxes. The proposed module can be directly applied to any single-shot object detection models to improve their performance in both classification and localization.

Chapter 5

Semantic Clustering based Deduction Learning for Image Recognition and Classification

In this section, we discuss the longitudinal semantic, realizing the high-level clustering in the semantic space, enabling the model to deduce the relations among various classes so as better classification performance is expected.

5.1 Introduction

The powerful ability for feature expression and semantic extraction of deep Convolutional Neural Networks (CNNs) has dramatically pushed the flourishing development of computer vision (66) (59) (60). At the same time, large-scale labeled data samples ensure the effectiveness of supervised learning, which enables the deep learning models to efficiently extract abstract but highly-semantic information for complicated vision tasks (105) (178) (192). Undoubtedly, future learning models should be complex, robust, knowledge-driven, and cognition-based (108) (19). This defines them with the cognitive ability of self-enhancing, synthesizing knowledge from multiple sources, and deducing based on knowledge and experiences (108).

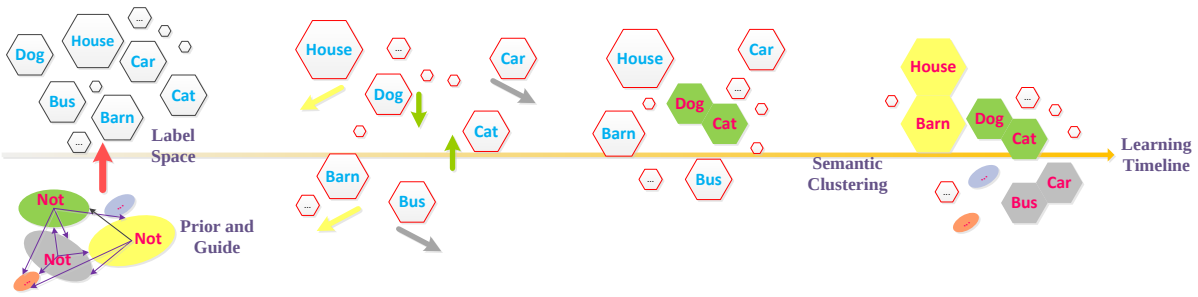

Figure 5.1: The deduction progress of semantic clustering. Prior and Guide works as the prior information that is combined with the labels as the labeling input data. The Learning Timeline is the same as the normal classification learning process, but our model provides the possibility of doing high-level semantic clustering by the deduction progress as the aid for the classification task. The model, at the end of the learning timeline, is expected to provide better classification accuracy.

Some complementary and weak supervision information has been exploited to boost the learning performance of models (93) (104). Such complementary supervision includes early side information(175), privileged information (157), and weak supervision based on semi-supervised data (87) (55), noisy labeled data (51) (113), and complementary labels (72) (184) (78). Most of these methods supplement extra direct labeling information or replace expensive accurate labels with cheap labeling information. These complementary labels, in fact, increase the labeling cost as a direct mapping from label space to sample space, named as “hard labeling" in the later sections. Most importantly, these methods are unable to equip deep models with the ability of self-enhancement, synthesis, and deduction.

In this paper, we leverage the wide-applied but fundamental supervised image classification and propose deduction learning by semantic clustering. We introduce semantic prior, high-level clustering information, represented by different colors in Figure 5.1, although no names are given for each color. For example, we expect the model know the cat and the dog should be closed to each other, though the model would never know they should be called “animal". Semantic prior (Prior and Guide to Label Space in Figure 5.1) is thus introduced into the classification learning models, guiding them to form effective semantic clustering so that they are able to deduce high-level semantic expression (Same color cells go attached together in Figure 5.1), as shown in Figure 5.1.

Inspired by the idea of negative learning (78) (184), we propose to guide the classifier to learn the opposite class that does not belong to the same cluster with the accurate label. For example, if a sample is labeled by “cat", then our algorithm will tell the classifier that the image is not “car" or any other random label that belongs to a different cluster other than “cat", during one learning shot.

This random search for the opposite label is in accordance with the semantic prior that is fed into the model along with other inputs that specifically refer to the images and their corresponding labels in this work. Statistically, the opposite semantic labels corresponding to a certain accurate label should be chosen with equal probability given the number of learning periods (epochs) is large enough. Theoretically, this proposed method enables a smooth clustering in the semantic space and an effective deduction, which makes the model able to deduce that “cat" should be one element of an abstract cluster, although the model would never know it can be called “animal", as shown in the second stage of Figure 5.1, where the colors “Green, Grey, Yellow", each represents a higher hierarchical category. Each specific class, like the cat, would be learning that if it belongs to “Green", then it is totally on the opposite side of other classes that belong to “Yellow" and “Grey".

Finally, it is noticed that the proposed method does not give up the conventional label learning by introducing one composite loss function. This ensures the label learning and the semantic clustering in the same timeline during the learning process. It conforms to the requirement of cognition learning (108). By this setting, the model could finish high-level semantic expressions, capturing the concepts, similar to “animal", “vehicle", “buildings", etc., as shown in the third stage in Figure 5.1, where sample classes accomplish clusters.

The major contributions of this paper are summarized below:

- *Semantic Clustering:* We propose a high-level semantic mapping within semantic space, enhancing the semantic expression and providing a certain level of independence for overcoming the limitation of convolution operation at the pixel level. It is realized by introducing a semantic prior which could guide the model to find the opposite semantic label that is not from the same semantic colony with the given true label.

- *Deduction Learning:* Deduction learning is realized by the semantic prior and the proposed random search for opposite semantic, which ensures the smoothness of semantic clustering and the robustness of classification. It could be implemented as a plug-in module that could play in arbitrary classification models by introducing a composite loss function.

- *Robust Improvement:* We achieved stable convergence and robust classification performance on mainstream classification models. It is also verified by working on noisy data environment where there exists a certain ratio of incorrect labels.

- *Wide Applicability:* In the proposed method, label learning and semantic clustering follow the same learning timeline, equipping the model with the ability of deduction and cognition. It can be taken as a plug-in module for broad deep learning applications, such as few-shot learning, zero-shot learning or even semi-supervised learning.

5.2 Related Work

5.2.1 Hierarchical Semantic Information

At first, the research in this field focuses on exploring or utilizing the inherent relations among label classes, or looking for the intermediate representations between classes. (1) formed a label-embedding problem where each class is embedded in the space of attribute vectors so that the attributes act as intermediate representations that enable parameter sharing between classes. Another research in (32) uses a label relation graph to encode flexible relations between class labels by building the rich structure of real-world labels. The idea of incremental learning by hierarchical label training has been explored recently by a few other papers. Progressive Neural Networks (138) learn to solve complex sequences of task by leveraging prior knowledge with lateral connections. "iCaRL" allows learning in a class incremental way: only the training data for a small number of classes is present at the same time and new classes can be added progressively (126). Tree-CNN (135), proposes training root network by general classes and then learning the fine classes

by corresponding growth-network (mainly learned by leaf structure of the network). While this research direction solves hierarchical semantic learning based on an independent timeline for each stage. Our proposed idea shares the same timeline with the normal classification task throughout the entire learning process which works as an exploration towards cognitive learning. At the same time, the methods above directly provide concrete class relation structure on the basis of the original class labels for training, without exploring the deduction ability of the networks.

Learning with real, concrete complementary labeling information was proposed by (72) for the image classification task. It was based on an assumption that the transition probability for complementary labels is equal to each other. It modified the traditional one-versus-all (OVA) and pairwise-comparison (PC) losses so that it is suitable for the uniform probability distribution, working as an unbiased estimator for the expected risk of true-labeled classification. Later on, the work (184) argued that there are two unsolved problems in the previous work. The first one lies in the fact that the complementary labels tend to be affected by annotators' experience and limited cognition. The other one is the proposed modified OVA and PC losses can not be generalized to more popular losses, such as the cross-entropy loss. Thus, they proposed the transition matrix setting to fix the bias from the biased complementary labels. At the same time, they provided intensive mathematical analysis to prove their proposed setting can be generalized to many losses which directly provides an unbiased estimator for minimizing expectation risk. These works expect better semantic learning by introducing intensive complementary labeling while they do not explore the deduction ability of the networks themselves as well. They are essentially regular label learning. The work in (78) automatically generated complementary labels from the given noisy labels and utilized them for the proposed negative learning, incorporating the complementary labeling into noisy label learning.

5.2.2 Semantic Labeling in Noisy Cases

Some researchers attempt to aid learning in noisy cases by introducing effective semantic label learning. Some attempt to create noise-robust losses by introducing transition probabilities to the

field of classification and transfer learning (45) (193). Some propose to use the transition layer to modify deep neural network (62). In other studies, researchers try to re-weight the training sample based on the reliability of the given label (132) (90). Some other approaches try to prune the correct samples from the softmax outputs (34) (151). Different from them, this paper dedicates to the research on how self-clustering and deduction learning ability of networks would influence the robustness in noisy labeling cases.

This paper tries to explore the self-deduction ability of networks in the semantic space and focuses on guiding the models to fetch effective hierarchical semantic information in a self-learning way by semantic clustering and cognitive accumulation. First, it could completely free the confinement problem of transition probabilities. The proposed semantic prior based random search for opposite semantic ensures the equal probability, providing the mapping independence in semantic space. Second, the semantic clustering boosts positive label learning. For example, if the sample “cat" has a low classification probability, the semantic clustering could help enhance this confidence by guiding this model to realize that the object is at least an animal, not a “car". Third, our proposed method shares the same timeline with conventional label learning, enabling effective cognitive accumulation. Moreover, there is no need for specifically defining loss functions for the proposed models. Following the loss formations of the original label learning in specific models is all we need, potentially leading to better generalization.

5.3 Problem Setup

People can make deduction independent of the actual vision behavior. Thus, in deep learning, we expect the model with similar independence to ensure the realization of high-level mapping in semantic space.

Semantic Space for Image Classification Semantic space is originally proposed in the natural language domain, aiming to create representations of natural language that are capable of capturing meanings (6). In computer vision, the concept of semantic space is much more abstract. Current

semantic extraction is limited both by spatial size and by the individual data sample. However, it should aim to overcome the limitations of convolution-based or receptive-field based approaches operating at the pixel level. Convolution-based deep learning models are fixed at the pixel level and are poor for generalization, which would easily break down if the individual image differs from or is strange to those in the training materials used for the statistical models. Compared to spatial feature learning that performs at the pixel level, semantic learning should be a relatively independent process that works on the semantic element, which is the common description for a class of objects. Moreover, the semantic expression could have multi-levels that describe the relevant or diverging characters of semantic elements. For example, the “cat" as a semantic element could be clustered to the high-level semantic expression, something similar to an “animal".

[Semantic Space] Without loss of generality, let $\mathscr{C}$ be the semantic space, $c \in \mathbb{Z}^+$ be the semantic element in $\mathscr{C}$ that appears as one semantic label indicating a specific object class. The semantic relation of different c is defined by r. [c] = {1,...,c} signifies the set of semantic labels. Then, we have

$$\mathscr{C} \stackrel{def}{=} \langle [c], r \rangle \tag{5.1}$$

where element c is uniformly sampled from $\mathscr{C}$. Tuple $\langle [c], r \rangle$ expresses the fact that semantic elements $c \in [c]$ are linked to each other by the relation r, forming the abstract spatial distribution in $\mathscr{C}$.

Semantic Cell In order to better describe the abstract relation distribution in $\mathscr{C}$, we propose *Semantic Cell* as the semantic unit that could label a group of objects that have similar features in feature space $\mathscr{X}$, which corresponds to the element $c \in [c]$ in Definition 6.2. It realizes a multi-to-one mapping that bridges the link between feature space $\mathscr{X}$ and semantic space $\mathscr{C}$. [Semantic Mapping] Let $g(\mathbf{x})$ be the mapping function of a given multi-class classification learning model that estimates the classification probabilities based on the input sample $\mathbf{x}$ in feature space $\mathscr{X}$. $f(\mathbf{x})$ predicts the classification label y based on the maximum probability principle, mapping the feature

sample $\mathbf{x}$ to the corresponding semantic cell c in $\mathscr{C}$.

$$f(\mathbf{x}) \stackrel{def}{=} \arg\max_{i \in [c]} g_i(\mathbf{x}) \tag{5.2}$$

where $f : \mathscr{X} \to \mathscr{C}$, the maximum probability of g and $f(\mathbf{x}) \in \mathscr{C}$. $g_i(\mathbf{x})$ realizes the estimation towards $P(y = i|\mathbf{x})$.

Semantic Colony Semantic Colony θ takes semantic cell c as individual sample. It clusters $c \in \mathscr{C}$ that hold related semantic information as θ. Based on which, it defines the intra-class relation and inner-class differentiation to realize clustering in semantic space $\mathscr{C}$ with high-order semantic expression. [Semantic Clustering] Without loss of generality, let Θ be the distribution of semantic colonies θ in $\mathscr{C}$. H conducts clustering for semantic cell $c \in \mathscr{C}$ into semantic colony $\theta \sim \Theta$. $\mathbf{c}$ is the vector with the elements of semantic cells $c \in [c]$. Then, we have

$$\theta \stackrel{def}{=} H(\mathbf{c}, r_{\mathbf{c}}) \tag{5.3}$$

where $H : [c] \to \Theta$, $\mathbf{c}$ consists of semantic cells c in [c] that are semantically related, and H maps $\mathbf{c}$ to $\theta \sim \Theta$ in accordance with the corresponding semantic relation $r_{\mathbf{c}}$.

5.4 Methodology

In this section, we first introduce the general approach that deep neural networks learn optimal classification with hard labels. Then, we discuss the learning with semantic deduction and propose corresponding training and test model.

5.4.1 Conventional Classification Learning

In multi-class classification, we aim to learn a classifier $f(\mathbf{x})$ that predicts the classification label y for a given observation sample $\mathbf{x}$. Typically, the classifier directly maps $\mathbf{x}$ into the label space $\mathscr{Y}$

by the following function:

$$f(\mathbf{x}) = \arg\max_{i \in \mathscr{Y}} W_i^T \mathbf{x} \tag{5.4}$$

where $f : \mathscr{X} \to \mathscr{Y}$ and W_i refers to the learning parameters of the classifier f, with the estimation of $P(y = i|\mathbf{x})$.

In supervised learning, loss functions are proposed to measure the expectation of the predicting $f(\mathbf{x})$ for y (7). It is typically defined as the expected risk (184) for various loss functions.

$$R(g) = \mathbb{E}_{\mathbf{x},y \sim P(\mathbf{x},y)}[\ell(f(\mathbf{x}), y)] \tag{5.5}$$

A well-trained classifier f^* minimizes this expected risk $R(g)$,

$$f^* = \arg\min_{f \in \mathscr{F}} R(f) \tag{5.6}$$

where $\mathscr{F}$ is the distribution space of f.

5.4.2 Learning with Semantic Deduction

In semantic space, the description of hard labels towards objects is limited. To better describe an object or a scene, people usually enumerate related features and associate their prior cognition and experience for a reasonable deduction. Current deep learning models realize feature sensing and learning but lack the proper deduction that could enrich the description of objects. Our previous analysis shows that hard labels in semantic space could potentially build more links, as the discussion in Section 2.1. We introduce the semantic prior, guiding the model to learn the semantic links by deduction. The overview of our method is depicted in Figure 7.2. The overall inputs include training sample images, corresponding labels, and the semantic prior information which provides the high-level semantic hierarchy of current classification labels. The classification model is trained in the same way as the original network. For the green part in Figure 7.2, given label y, the model finds the corresponding opposite semantic label for the sample image according to the

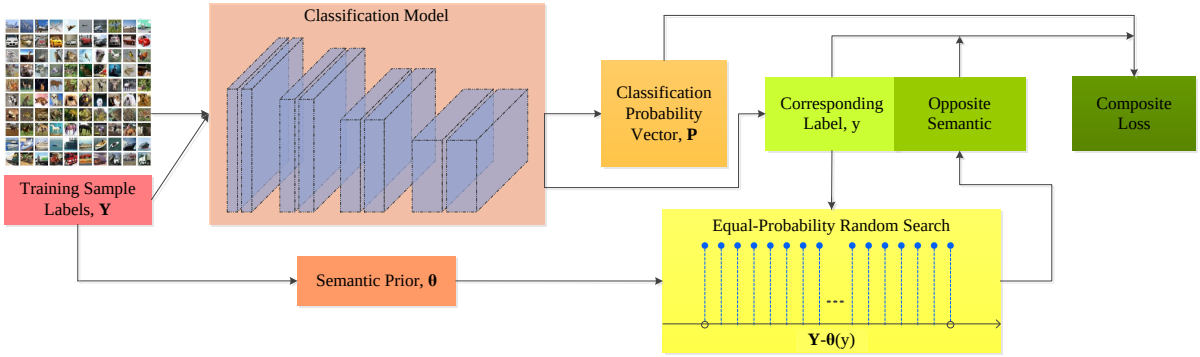

Figure 5.2: An overview of the proposed method. We use semantic prior based random search to produce opposite semantic so as to form the composite loss function, guiding the model to form semantic colonies.

semantic prior by an equal-probability random search, shown as the yellow block. Then both the true label and the opposite semantic label are fed into the composite loss we defined. The output of the proposed method is expected of better classification performance in the way of classification accuracy.

First, the semantic prior works as the criterion for colonies' formation in semantic space $\mathscr{C}$. For example, a cat labeled by $c_i \in [c]$ should be grouped into "animal" colony, if denoted by θ_m. Similarly, a car labeled by c_j could be grouped into the "vehicle" colony θ_n. Second, the semantic deduction is fully performed in semantic space $\mathscr{C}$, instead of defining complementary labels as weak supervision. Thus, we do not need any tedious and laborious labeling work, which would avoid labeling bias from human beings' bias (184), and the problem that the complementary labeling is essentially non-uniformly selected from the $c-1$ classes other than the true label class ($c > 2$).

5.4.3 Equal-Probability Search for Opposite Semantic.

We assume that the variables $(\mathbf{x}, c, \theta)$ are defined in the space $(\mathscr{X} \times [c] \times \Theta)$, with the joint probability measure $P(\mathbf{x}, c, \theta)$.

Given a sample $(\mathbf{x}, c, \theta) \in (\mathscr{X} \times [c] \times \Theta)$, its opposite classification label $\bar{c}$ is randomly selected from $[c]\backslash\theta$. When the sampling frequency in a complete learning period is greatly larger than the class number $n_{[c]}$, the probability for each $\bar{c} \in [c]\backslash\theta$ that indicates how likely it is selected can be expressed as

$$P_i(\bar{Y} = \bar{c}|X = \mathbf{x}, Y = c) = \frac{1}{n_{([c]\backslash\theta)}} \tag{5.7}$$

where $n_{([c]\backslash\theta)}$ is the number of semantic cells in $[c]\backslash\theta$. This conclusion verifies that the proposed semantic-prior based random search method for the opposite semantic label $\bar{c}$ is statistically consistent, and it realizes the independency of $\bar{c}$ with respective to feature space $\mathscr{X}$ conditioned on c and θ. Thus we have,

$$P(\bar{Y} = \bar{c}|X = \mathbf{x}, Y = c) = P(\bar{Y} = \bar{c}|Y = c) \tag{5.8}$$

The optimal classifier can be found under the uniform assumption, which has been proven in previous work (72). Meanwhile, the uniform selection means equal probability, ensuring the smooth clustering and the stability and robustness of the learning process. While for man-made complementary labels, they are confined by the fact that $\bar{Y}$ is assumed to be independent of feature $\mathscr{X}$ (184) (72).

Based on the exist of independence, the complete mapping from $\mathbf{x}$ to $\bar{y}$ can be set up as the following formula, $\forall i, j \in [c]$,

$$\begin{aligned} P(\bar{y}|\mathbf{x}) &= \sum_{i \in \theta_i, j \notin \theta_i} P(\bar{y} = j, y = i|\mathbf{x}) \\ &= \sum_{i \in \theta_i, j \notin \theta_i} P(\bar{y} = j|y = i, \mathbf{x}) P(y = i|\mathbf{x}) \\ &= \sum_{i \in \theta_i, j \notin \theta_i} P(\bar{y} = j|y = i) P(y = i|\mathbf{x}) \end{aligned} \tag{5.9}$$

5.4.4 Learning with Smooth Semantic Clustering

Conventionally, the classifier is trained to learn that the input image belongs to a specific, single class label. Let $\mathbf{x} \in \mathscr{X}$ be the input image, $y \in [c]$ denotes its label. $f(\mathbf{x}, W)$ maps the input $\mathbf{x}$ to

the score space: $\mathscr{X} \to \mathbb{R}^c$, as equation (5.4) shows. The training process is guided by the cross entropy loss (most popular classification cost function) of f as

$$\mathscr{L}_{\mathbb{P}}(f,y) = -\sum_{m=1}^{c} \mathbf{y}_m \log \mathbf{p}_m \tag{5.10}$$

where $\mathbf{y} \in \{0,1\}^c$ is the one-hot vector form of y. $\mathbf{p}_m$ is the m^{th} element of probability vector $\mathbf{p}$. The conventional learning process is to optimize the probability $\mathbf{p}_m$ according to the given exact label $\mathbf{y}_m$ so that $\mathbf{p}_m \to 1$. Based on which, we propose a learning algorithm with smooth high-level clustering by guiding f to learn the semantic prior from the opposite label. Inspired by (78), the opposite semantic should push f to optimize the corresponding classification probability $\bar{\mathbf{p}}_m \to 0$.

$$\mathscr{L}_{\mathbb{O}}(f,y) = -\sum_{m=1}^{c} \bar{\mathbf{y}}_m \log(1-\bar{\mathbf{p}}_m) \tag{5.11}$$

where $\mathbf{y}_m \in \theta_m$, $\bar{\mathbf{y}}_m \in [c]$ and $\bar{\mathbf{y}}_m \notin \theta_m$. $\bar{\mathbf{p}}_m$ is the corresponding classification possibility of label $\bar{\mathbf{y}}_m$ in vector $\mathbf{p}$. Thus, the random selection of $\bar{\mathbf{y}}_m$ comes from $[c]\backslash\theta$ in every iteration during the training process, shown in Algorithm 1.

Algorithm 2 Smooth Semantic Clustering

Input: Training label $y \in \mathscr{Y} = [c]$, semantic prior $\hat{\theta} \sim \hat{\Theta}$

while iteration **do**
  if $y \in \hat{\theta}_i$ **then**
    $\bar{y}$ = Select randomly from $[c]\backslash\hat{\theta}_i$
  There exists another semantic colony θ_j
  if $\bar{y} \in \theta_j$ **then**
    $y \notin \theta_j$

Output: Opposite semantic label $\bar{y}$ and the learned semantic colony $\theta \sim \Theta$

From Algorithm 1, we can observe that the learning for clustering in the semantic space $\mathscr{C}$ is synchronous with image classification. Thus, we can define a composite loss function for an

end-to-end semantic clustering classifier.

$$\begin{aligned}\mathscr{L} &= \alpha_1 \mathscr{L}_{\mathbb{P}} + \alpha_2 \mathscr{L}_{\mathbb{O}} \\ &= -\alpha_1 \sum_{m=1}^{c} \mathbf{y}_m \log \mathbf{p}_m - \alpha_2 \sum_{m=1}^{c} \bar{\mathbf{y}}_m \log(1-\bar{\mathbf{p}}_m)\end{aligned} \tag{5.12}$$

where α_1 and α_2 are weights defining the ratio of $\mathscr{L}_{\mathbb{P}}$ and $\mathscr{L}_{\mathbb{O}}$ respectively.

For a specific input image, there is not only a semantic label y but also other semantic description $\theta \sim \Theta$, and θ is the high-level semantic expression corresponding to y, which builds a new semantic attribute with a larger range. Since the opposite semantic is randomly selected with equal probability, the clustering hyperplane in $\mathscr{C}$ can be smooth.

5.4.5 Optimal Learning

In the case of $\mathscr{L}$, we define the expected risk $\bar{R}(f)$ with the mapping $f : \mathscr{X} \to \{[c], \Theta\}$. If we can find an optimal f^* such that $f^* = P(Y = i|X), \forall i \in [c]$, then in theory, we expect that we can find the optimal $\bar{f}^*$ such that $\bar{f}^* = P(\bar{Y} = i|X), \forall i \in [c]$, where $P(\bar{Y}|X) = \sum_{i \in \theta_i, j \notin \theta_i} P(\bar{Y} = j, Y = i|X)$ according to equation (5.9). If the above idea can be proved, with sufficient training samples, the proposed algorithm with $\bar{R}(f)$ is capable of simultaneously learning a good classification and clustering for (X, Y, θ).

Following (184), we will prove that the proposed semantic clustering learning with its corresponding loss function $\mathscr{L}$ is able to identify the optimal classifier. First, we introduce the following assumption (184), The optimal learning with mapping f^* satisfies $f_i^*(X) = P(Y = i|X), \forall i \in [c]$ by minimizing the expected risk $R(f)$.

Based on this assumption, we are able to prove that $\bar{f}^* = f^*$ following the theorem below (184).

Theorem 1 *Suppose that Assumption 1 is satisfied, then the minimum solution $\bar{f}^*$ of $\bar{R}(f)$ is also the minimum solution f^* of $R(f)$, i.e., $\bar{f}^* = f^*$.*

Based on Assumption 1, loss function $\mathscr{L}$, and function (5.9) for the learning in the proposed

smooth semantic clustering, we have

$$\begin{aligned} f_i^*(X) &= P(\bar{Y}=j|X) \\ &= \sum_{i\in\theta_i} P(\bar{Y}=j, Y=i|X), \forall i,j\in[c], j\notin\theta_i \end{aligned} \tag{5.13}$$

Let $\bar{\mathbf{s}}(X)=[P(\bar{Y}=1|X),\cdots,P(\bar{Y}=c)|X)]$ and $\mathbf{s}(X)=[P(Y=1|X),\cdots,P(Y=c)|X)]$. According to the discussion of (184), we rewrite $\bar{R}(f)$ as

$$\begin{aligned} \bar{R}(f) &= \int_X \sum_{j=1}^{c} P(\bar{Y}=j)P(X|\bar{Y}=j)\mathscr{L}(f(X),\bar{Y}=j)dX \\ &= \sum_{j=1}^{c} P(\bar{Y}=j)\int_X P(X|\bar{Y}=j)\mathscr{L}(f(X),\bar{Y}=j)dX \\ &= \sum_{j=1}^{c} P(\bar{Y}=j)\bar{R}_j(f) \end{aligned} \tag{5.14}$$

where $P(\bar{Y}=j)$ is given when we have $Y=i$, distributed as $P(\bar{Y}=j|Y=i)$ according to Algorithm 1. $\bar{R}_j(f)=\int_X P(X|\bar{Y}=j)\mathscr{L}(f(X),\bar{Y}=j)dX$. Thus, if we use $\mathbf{C}$ to denote the operation form of $P(\bar{Y}=j|Y=i)$, according to function (5.9) and the above convergence analysis, we have

$$\bar{\mathbf{s}}(X)=\mathbf{C}^T\mathbf{s}(X) \tag{5.15}$$

where $P(\bar{Y}=j|Y=i)$ is realized based on the random search with semantic prior. Equation (5.15) ensures that

$$\bar{f}^*(X)=\arg\max_i \mathbf{C}^T\mathbf{s}_i(X)=\mathbf{C}^T\arg\max_i \mathbf{s}_i(X)=\mathbf{C}^T f^*(X) \tag{5.16}$$

where $i\in[1,c]$. Thus, we have $\bar{f}^*\Longleftrightarrow f^*$. The proof is completed.

5.5 Experiment

In this section, we study the impact of the proposed semantic deduction algorithm on popular image classifiers using mainstream benchmark datasets. In order to show that our algorithm is

able to generalize to complex or disordered data environment with better robustness, we follow each specific experimental setting of the baseline methods, and only vary the data environment by producing noisy labels at certain ratios.

Learning Scenarios To identify the gain of the proposed deduction learning algorithm, we design fairly comparable learning scenarios where only the deduction related hyper-parameters are changed from the default original setting while keeping all the rest unchanged. The assignment for the weights of α_1 and α_2 in equation5.12 is based on the experiment performance. We introduce the most core algorithm idea of the current state-of-the-art works of complementary supervision information designed for various fields (78) (184) (72) into our experiment setting as one of the baselines. Details are listed below:

- *Default Setting (OT):* In this setting, we train the original baseline classification models and keep all the hyper-parameters unchanged as in the corresponding published papers and public code. We take both classical and state-of-the-art CNN classifier networks into consideration, including Multilayer Perceptron (MLP)(117), VGG (117), ResNet(59), DenseNet (66), Wresnet (185), ResNext(174). All of them are trained and compared with our proposed methods fairly.

- *Random Opposite Semantic (RT):* Under this setting, we exploit the opposite semantic label $\bar{y} \in [c]$ that corresponds to the original accurate label $y \in [c]$, satisfying $\bar{y} \neq y$. We use random search for the opposite label in the label pools $[c]$ (78) instead of hard labeling so as to avoid bias (184) (78). Thus, this setting does not refer to the semantic prior when looking for the opposite semantic label $\bar{y}$. All other settings follow the Default Setting.

- *Semantic Deduction (SD):* We implement the proposed deduction learning by semantic clustering. The opposite semantic label $\bar{y}$ is randomly selected from $[c] \backslash \hat{\theta}_i$, where $[c]$ is the set of semantic labels. $\hat{\theta}_i$ is the *i_th* semantic colony (details in Algorithm 1). Thus, it naturally satisfies $\bar{y} \neq y$, y referring to the original accurate label $y \in [c]$. It strictly follows the training setting with the identical hyper-parameters to those in the Default Setting.

Data Sets

- *Fashion-MNIST:* Fashion-MNIST is a new image classification benchmark with different data classes of clothing(171). The dataset has an image size of 28×28, input channels of 1, and the number of classes of 10. In our SD setting, we provide the semantic prior for it to group the 10 classes fashion clothing into three groups: “clothes", “shoes", and “bags".

- *CIFAR10:* CIFAR10 consists of $50,000$ training images and $10,000$ test images of dimension 32×32. It has a total of 10 general classes(81). In the SD setting, we group the 10 classes into two groups, “vehicles" and “animals".

- *CIFAR100:* CIFAR100 has $50,000$ training images and $10,000$ test images of the resolution of 32×32. It has a total of 100 classes, with 500 training images in each class (81). For the SD setting, we provide two schemes, “SD_v1" and “SD_v2. The former one divides classes into “7" groups, including “people", “animal", “man-made stuff", “transportation", “plants", “building", and “nature". The latter contains 8 groups: “people", “animal", “life appliances", “transportation", “food", “plants", “building", and “nature", isolating “food" from the “man-made" as an independent expression.

5.5.1 Results in Original Data Environment

We first evaluate our proposed algorithm in the original data environment, directly using the images from the data sources. From the mathematical analysis in Section 4, the identification of optimal learning depends on stable convergence performance. Thus, we summarize the learning behaviors of each approach on CIFAR10 and CIFAR100 in Figure 5.3 and Figure 5.4, respectively.

Convergence Performance To obtain a fair comparison, we normalize the loss distribution to $[0,1]$ for all scenarios. **(a)** Our algorithm generally shows consistent convergence with different classifiers, as shown in the red or yellow solid lines in Figure 5.3 and 5.4. We can see that SD usually converges faster than RT as the black arrows shown in almost every case. This consistent

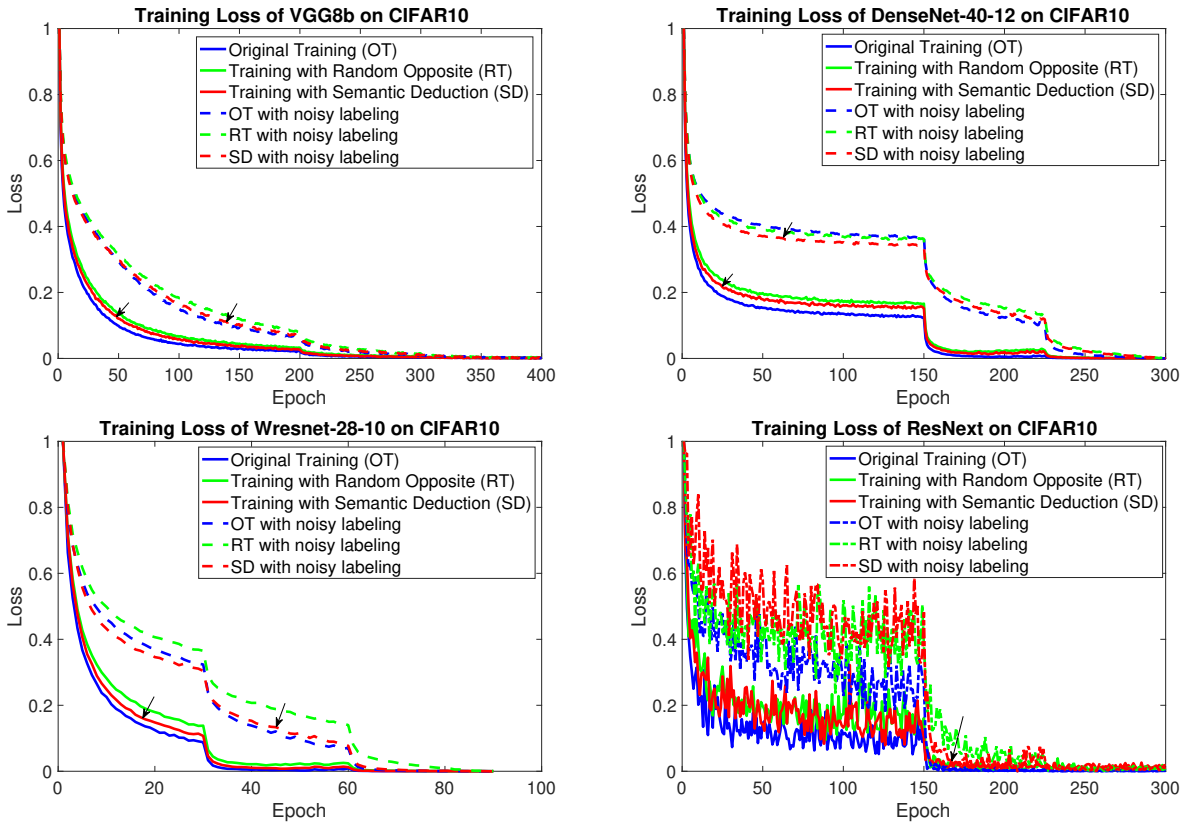

Figure 5.3: Convergence performance of different models by training loss on CIFAR10.

performance verifies that the proposed self-clustering learning process helps speed up convergence, assisting the classifier to execute the right decision, although there is no additional labeling information fed into these models. **(b)** From all the sub-figures in both Figure 5.3 and 5.4, although SD converges a little bit slower than the original baseline (solid blue line) at the first stage, they finally obtain similar stability. This is due to the introduction of the additional learning process, semantic clustering. **(c)** Although we design two semantic prior schemes, SD_v1 and SD_v2, they both show very consistent convergence, where the red and black solid lines even overlap with each other in Figure 5.4. **(d)** The fluctuation in ResNext is due to the non-averaged loss value in the original code for each epoch. From the above observation, it is evident that the introduction of semantic clustering achieves stable and fast convergence, theoretically qualified to yield an optimal classification mapping.

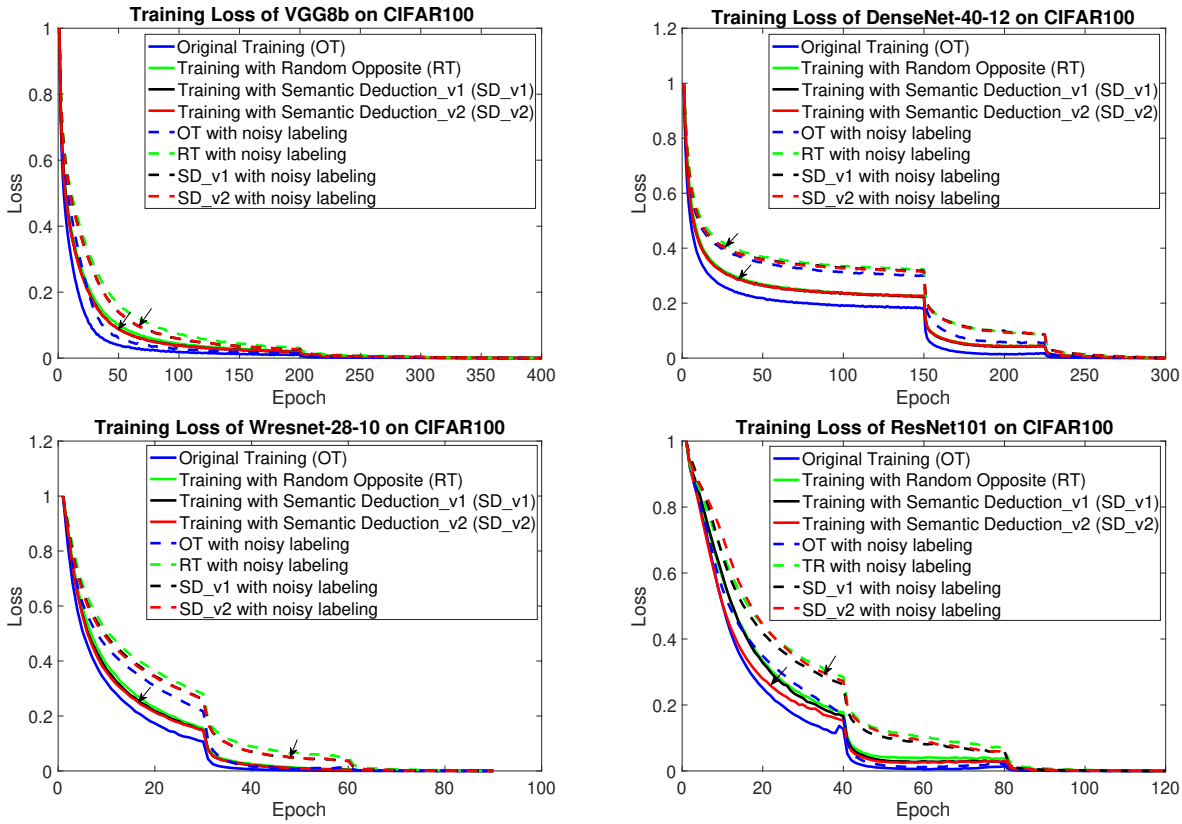

Figure 5.4: Convergence performance comparison by training loss on CIFAR100.

Method	Solver	α_1	α_2	OT	RT	SD(ours)
MLP-3(117)	adam	1	0.5	91.5	91.77	**91.78**
VGG8b (117)	adam	1	0.3	95.45	95.47	**95.53**
VGG8b(multi=2.0) (117)	adam	1	0.3	95.33	95.52	**95.54**

Table 5.1: Classification accuracy on FashionMNIST.

Method	Solver	α_1	α_2	OT	RT	SD(ours)
VGG8b (117)	adam	1	0.5	94.12	94.14	**94.32**
ResNet18(59)	adam	1	0.5	93.45	93.57	**93.62**
DenseNet-40-12 (66)	sgd	1	0.5	94.68	94.79	**94.92**
Wresnet-28-10 (185)	sgd	1	0.5	94.52	94.58	**94.80**
ResNext (174)	sgd	1	0.5	96.16	96.26	**96.30**

Table 5.2: Classification accuracy on CIFAR10.

Classification Accuracy We summarize the classification accuracy in Table 5.1, 5.2, and 5.3. **(a)** Generally, SD obtains almost the highest classification accuracy across the three benchmarks for all the compared classifiers. These classifiers include two mainstream solvers, adam (79) and sgd (15), but SD leads the performances in both situations. **(b)** Although the improvement brought by SD is limited for Fashion MNIST, this is mainly due to the relatively simple classification task and the limited number of classes. When it comes to CIFAR100 as shown in Table 5.3, SD always yields 1-3% increase in accuracy compared with OT. **(c)** We can observe that RT in some special situations achieves high performance, such as RT winning SD in the case of Wresnet-28-10. However, its performance is not as stable as SD, which even yields lower classification accuracy than OT, such as that in the case of ResNet101. These observations imply that the proposed smooth semantic clustering algorithm can effectively enhance the performance of state-of-the-art classifiers, preserving a very stable learning state at the same time, potentially leading to its broader applicability.

Compared with the recent publication (135), which proposes a network learning algorithm that organizes the incrementally learning data into feature-driven super-class and improves upon existing hierarchical CNN models by introducing the capability of self-growth, so that the finer classification is done. This idea, to a certain degree, shares a similar concept with our idea, except that we do not need to label data with super-class and keep the same hierarchical structure during the learning process. We compare its results with ours in Table 5.4 and Table 5.5, respectively. It can be seen from them that, although the Tree-CNN models provide a competitive accuracy with its base network VGG-11, it shows no advantages over our SD models. SD models obtain a more than 4% advantage over incremental learning methods (VGG11 and Tree-CNN in Table 5.4) on CIFAR 10 and averagely 5% higher than incremental learning methods on CIFAR100 considering the test classification accuracy. It demonstrates that our proposed high-level semantic clustering algorithm, in a direct supervised learning, could further improve the adaptive ability towards data, and keep a stable learning process, which is further verified in the following sections. Most importantly, we focus on the exploration towards the self-deducing ability of CNN models, which is different from

Method	Solver	α_1	α_2	OT	RT	SD_v1	SD_v2
VGG8b (117)	adam	1	0.5	73.85	74.78	**74.95**	74.83
ResNet50 (59)	sgd	1	0.5	73.78	76.36	75.59	**76.64**
DenseNet-40-12 (66)	sgd	1	0.5	74.89	75.82	**76.26**	75.73
Wresnet-28-10 (185)	sgd	1	0.5	76.98	**77.62**	77.54	77.59
ResNet101 (59)	sgd	1	0.5	75.3	74.45	75.51	**76.29**
ResNet152 (59)	sgd	1	0.3	72.21	73.25	74.38	**74.40**

Table 5.3: Classification accuracy on CIFAR100.

Model	VGG11	Tree-CNN	VGG8b	ResNet18-SD	DenseNet-SD	WresNet-SD
Test Accuracy	90.51	86.24	94.32	93.62	94.92	94.80

Table 5.4: Comparison with Tree-CNN on Cifar10, where SD refers to models that are applied with our proposed algorithm. VGG11 and Tree-CNN are trained by "old" and "new" data in an incremental way (135).

all the above-mentioned ideas.

5.5.2 Results in Noisy Data Environment

In this section, we evaluate the proposed algorithm in noisy data environments. We produce a noisy data environment by adding noise labels to the original data sources. Specifically, we implement this operation on CIFAR10 and CIFAR100, where 10% of the training data in each data set are randomly labeled by incorrect labels that belong to the same colony with the correct labels. For example, if the image is labeled correctly by “cat", then we randomly search another class label in the “animal" cluster such as “dog" as the replacement of the label “cat".

Convergence Performance The comparative results are shown in Figures 5.3 and 5.4, from which we can see that **(a)** SD maintains the same learning stability as that in original environment.

Model	VGG11	Tree-CNN5	Tree-CNN10	Tree-CNN20	VGG8b-SD	Wresnet-28-10-SD
Test-Acc	72.23	69.85	69.53	68.49	74.95	77.54

Table 5.5: Comparison with Tree-CNN on Cifar100, where Test-Acc stands for the Test Accuracy. SD refers to the corresponding models that are applied with our proposed algorithm. VGG11 and Tree-CNN are trained by "old" and "new" data in an incremental way (135).

Method	Solver	α_1	α_2	OT	RT	SD(ours)
VGG8b (117)	adam	1	0.3	89.71	**90.52**	90.33
ResNet18 (59)	adam	1	0.3	89.22	**90.71**	90.32
DenseNet-40-12 (66)	sgd	1	0.5	91.47	92.16	**92.25**
wresnet-28-10(185)	sgd	1	0.5	89.07	87.67	**89.21**
ResNext(174)	sgd	1	0.3	91.29	92.17	**92.53**

Table 5.6: Classification on CIFAR10 with noisy labels.

Method	Solver	α_1	α_2	OT	RT	SD_v1	SD_v2
VGG8b (117)	adam	1	0.5	67.68	68.72	68.89	**68.95**
ResNext (59)	sgd	1	0.5	75.48	74.51	75.03	**75.65**
DenseNet-40-12 (66)	sgd	1	0.5	70.25	**72.80**	72.61	72.09
wresnet-28-10 (185)	sgd	1	0.5	71.42	71.79	71.60	**72.59**
ResNet101(59)	sgd	1	0.5	68.93	67.97	68.71	**69.75**

Table 5.7: Classification on CIFAR100 with noisy labels.

It even surpasses the baseline OT by convergence speed in some cases, such as DenseNet-40-12 and Wresnet-28-10 on CIFAR10. **(b)** SD generally converges faster than RT, especially in the case of Wresnet-28-10. It shows SD works better assisting the classifier to execute reasonable classification decisions in noisy situations, which exhibits good robustness of the proposed algorithm. **(c)** SD with the composite loss function "$\mathscr{L}$" shows perfect robustness across both shallow and deep networks. Thus, SD is expected to identify the optimal classification theoretically.

Classification Accuracy The comparative results are shown in Tables 5.6 and 5.7. We can observe that **(a)** SD, in general, surpasses OT by 1-2%. **(b)** Although RT surpasses SD in some cases, their results are very close. SD is always consistent for all the compared models. **(c)** RT is less robust than SD for its poor performance in some cases with much lower accuracy than OT, such as Wresnet-28-10 on CIFAR10, and ResNext and ResNet101 on CIFAR100.

These observations indicate that the proposed deduction learning by semantic clustering not only enhances the classification performance but also improves the generalization for a given classifier. From the above experiments, it is evident that the proposed semantic clustering method can help the model achieve more accurate classification decisions. Although the semantic prior-based

opposite label search provides rough information, it can aid the model to deduce high-level semantic expression along with the entire learning process, realizing the experience accumulation and basic cognitive learning. Thus, it could be an excellent plug-in module that could be applied in other supervised learning, few-shot learning, zero-shot learning, or even semi-supervised learning where each learning stage could be a better fit, generalized, and becoming much more robust. In the meanwhile, from the perspective of calculation, the proposed mechanism of deduction learning by the opposite semantic constraint only introduces one more loss item, which is only the tenth level of the order of magnitudes. Compared with matrix multiplication of any two layers during the training process which has the million level of the order of magnitudes, our proposed model is capable of keeping the time complexity of calculation, while its superior stability and robustness make it easy to be generalized to other computer vision tasks.

5.6 Conclusion

In this paper, we have proposed a deduction learning approach to boost the gain of high-level semantic clustering. We have demonstrated that if a classifier can perform further independent mapping in the semantic space, it will help the model achieve higher classification performance with better generalization ability and robustness. The proposed smooth semantic clustering algorithm ensures label learning and semantic deduction being processed in the same timeline so as to form a basic cognition. Extensive experiments across various classifiers on different datasets demonstrate the superiority of the proposed method toward further enhancing state-of-the-art classification performance.

Chapter 6

Self-Orthogonality Module:A Network Architecture Plug-in for Learning Orthogonal Filters

6.1 Introduction

Nowadays deep learning has been achieving the state-of-the-art performance in many applications such as computer vision and natural language processing. Regularization in deep learning plays an important role to help avoid bad solutions. In the literature researchers have made great efforts on this topic from different perspectives, such as data (71; 2; 26), network architectures (187; 59; 185), losses (112), regularizers (52; 134; 191; 4; 110), and optimization (Bottou; 65; 140; 68). Please refer to (84) for a review.

To better understand the effects of regularization in deep learning, our work in this paper is mainly motivated by the following two basic yet important questions:

Q1. *With the help of regularization, what structural properties among the learned filters (weights for convolutional and full-connected (FC) layers*[1]*) are good deep models supposed to have?*

[1]For simplicity, in the rest of the paper we refer to a convolutional or FC layer as a hidden layer.

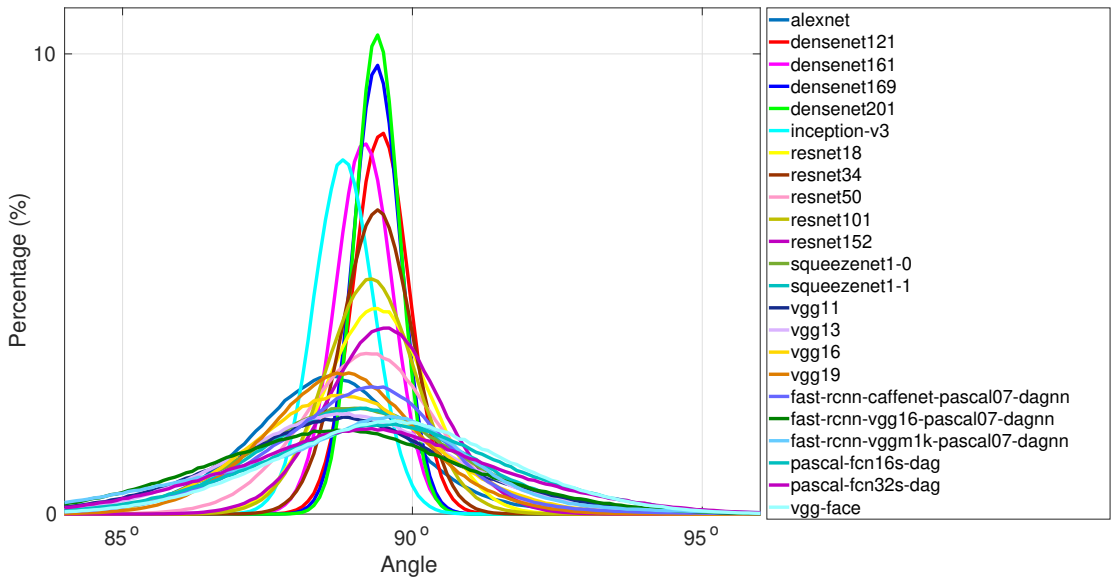

Figure 6.1: Illustration of the angular distributions of pretrained models with no OR.

To answer this question (partially), we try to explore the angular properties among learned filters. We compute the angles of all filter pairs at each hidden layer in different deep models and plot these *angular distributions* in Fig. 6.1. To generate each distribution, we first uniformly and randomly draw a sample from the angle pool per hidden layer, and average all the samples to generate a model-level angular sample. We then repeat this procedure for 10^6 times, leading to 10^6 samples based on which we compute a (normalized) histogram as the angular distribution by quantization from 0 to 180, step by 0.1. All the 23 deep models[2] are properly pretrained on different data sets with weight decay (52), dropout (65), and batch normalization (BN) (71).

As shown in Fig. 6.1, all the angular distributions overlap with each other heavily and behave similarly in Gaussian-like shapes with centers near 90 with small variances. Intuitively orthogonal filters are expected to best span the parameter space, especially in the high dimensional spaces where the filter dimensions are larger than the number of filters. Empirically, however, with many noisy factors such as data samples and stochastic training it may not be a good idea to strictly preserve the filter orthogonality in deep learning. In fact, the recent work in (85) has demonstrated that on benchmark data sets, classification accuracy using orthogonal filters (learned by PCA) is

[2]See `http://www.robots.ox.ac.uk/~albanie/mcn-models.html`.

inferior to that using learned filters by backpropagation (BP). Similarly another recent work in (160) finds that hard constraints on orthogonality can negatively affect the convergence speed and model performance in training of recurrent neural networks (RNNs), but soft orthogonality can improve the training.

In summary, the comparison on the angular distributions of pretrained deep models in Fig. 6.1 reveal that deep learning itself may have some internal mechanism to learn nearly orthogonal filters due to its high dimensional parameter spaces, even without any external orthogonal regularization (OR).

Q2. *What are the intrinsic benefits from learning orthogonal filters in deep learning based on OR?*

We notice that recently OR has been attracting more and more attention (134; 188; 160; 68; 4; 85), some of which (134; 160; 68) have released their code. Interestingly, from their code we find that the proposed OR is evaluated together with other regularizers such as weight decay, dropout, and BN. We argue that such experimental settings cannot help identify how much OR contributes to the performance, compared with other regularizers, especially as we observe that the performances with or without OR are very close. Similar argument has been addressed in (156) recently where the author showed that ℓ_2 regularization has no regularizing effect when combined with batch or weight normalization, but has an influence on the scale of weights, and thereby on the effective learning rate.

In summary, it is unclear to us from existing works what is the real gain from OR in deep learning.

Contributions: This paper aims to identify the real gain from OR in training different deep models on different tasks. To do so, we conduct comprehensive experiments on point cloud classification. In contrast to previous works, we separate OR from other regularization techniques to train the same networks respectively. We observe that, however, no significant improvement in accuracy occurs from existing OR techniques, statistically speaking, compared with the conventional training algorithm based on weight decay, dropout, and batch normalization. In fact, we find that, even without any regularization, a workable deep model can achieve the near orthogonality among

learned filters, indicating that OR may not be necessarily useful in deep learning to improve accuracy.

What we do observe is that sometimes the training stability using OR is improved, leading to faster convergence in training and better accuracy at test time. We manage to identify this by intentionally designing experiments in extreme learning scenarios such as large learning rate, limited training samples, and small batch sizes. Such observation, however, is not strong overall. We conjecture that this is mainly because existing OR techniques influence the deep learning *externally* and cannot be integrated as a part of network architectures *internally*.

To verify our conjecture, we propose a *self-regularization* technique as a plug-in to the network architectures so that they are able to learn (nearly) orthogonal filters even without any other regularization. We borrow the idea from locality sensitive hashing (LSH) (21) to approximately measure the filter angles at each hidden layer using filter responses from the network. We then push the statistics of such angles (mean and variance) towards 90 and 0, respectively, as an orthogonality regularizer. We demonstrate that our internal self-regularization significantly improves the training stability, leading to faster convergence and better generalization.

6.1.1 Related Work

As summarized in (84), there are many regularization techniques in deep learning. For instance, weight decay is essentially an ℓ_2 regularizer over filters, dropout takes random neurons for update, and BN utilizes the statistics from mini-batches to normalize the features. Our work is more related to representation decorrelation and orthogonality regularizers in the literature.

Representation Decorrelation: Cogswell (26) proposed a regularizer, namely DeCov, to learn non-redundant representations by minimizing the cross-covariance of hidden activations. Similarly, Gu (54) proposed another regularizer, namely Ensemble-based Decorrelation Method (EDM), by minimizing the covariance between all base learners (hidden activations) during training. Yadav and Agarwal (181) proposed regularizing the training of RNNs by minimizing non-diagonal elements of the correlation matrix computed over the hidden representation, leading to DeCov RNN

loss and DeCov Ensemble loss. Zhu (198) proposed another decorrelation regularizer based on Pearson correlation coefficient matrix working together with group LASSO to learn sparse neural networks. None of these works, however, guarantee that the learned filters should be (nearly) orthogonal.

Orthogonality Regularizers: Harandi and Fernando (56) proposed a generalized BP algorithm to update filters on the Riemannian manifolds as well as introducing a Stiefel layer to learn orthogonal filters. Vorontsov (160) verified the effect of learning orthogonal filters on RNN training that is conducted on the Stiefel manifolds. Huang (68) proposed an orthogonal weight normalization algorithm based on optimization over multiple dependent Stiefel manifolds (OMDSM). Xie (173) proposed a family of orthogonality-promoting regularizer by encouraging the Gram matrix of the functions in the reproducing kernel Hilbert spaces (RKHS) to be close to an identity matrix where the closeness is measured by Bregman matrix divergences. Rodríguez (134) proposed a regularizer called OrthoReg to enforce feature orthogonality locally based on cosine similarities of filters. Bansal (4) proposed another two orthogonality regularizers based on mutual coherence and restricted isometry property over filters, respectively, and evaluated their gain in training deep models. Xie (172) demonstrated that orthonormality among filters helps alleviate the vanishing or exploring gradient issue in training extremely deep networks. Jia (74) proposed the algorithms of Orthogonal Deep Neural Networks (OrthDNNs) to connect with recent interest of spectrally regularized deep learning methods.

Self-Regularization: Xu (176) proposed a self-regularized neural networks (SRNN) by arguing that the sample-wise soft targets of a neural network may have potentials to drag its own neural network out of its local optimum. Martin & Mahoney (109) proposed interpreting deep neural networks (DNN) from the perspective of random matrix theory (RMT) by analyzing the weight matrices in DNN. They claimed that empirical and theoretical results clearly indicate that the DNN training process itself implicitly implements a form of self-regularization, implicitly sculpting a more regularized energy or penalty landscape.

Differently, we propose introducing self-regularization into the design of OR for better un-

derstanding the truly impact of OR in deep learning. In contrast to (109) we discover the self-regularization in convolutional neural networks (CNNs) by considering their angular distributions among learned filters in the context of OR. We propose a novel efficient activation function to compute these angles.

6.2 Self-Regularization as Internal Orthogonality Regularizer

To better present our experimental results, let us first introduce our self-regularization method. Recall that inspired by the angular distributions of pretrained deep models, we aim to learn deep models with mean and variance of their angular distributions close to 90 and 0 as well. Intuitively we could use $\theta = \arccos \frac{\mathbf{w}_n^T \mathbf{w}_n}{\|\mathbf{w}_m\|\|\mathbf{w}_n\|} \in [0, \pi]$ to directly compute the angle between two filters $\mathbf{w}_m$ and $\mathbf{w}_n$, where $(\cdot)^T$ denotes the matrix transpose operator, and $\|\cdot\|$ denotes the ℓ_2 norm of a vector. Empirically we observe similar behavior of this arccos based regularization to that of SRIP-v1 and SRIP-v2 (see Sec. 6.3 for more details). We argue that all the existing orthogonality regularizers are designed *data independently*, and thus lack of the ability of data adaptation in regularization.

In contrast to the literature, we propose introducing implicit *self-regularization* into OR to embed the regularizer into the network architectures directly so that it can be updated *data dependently* during training. To this end, we actively seek for a means that can be used to estimate filter angles based on input data. Only in this way we can naturally incorporate self-regularization with network architectures. Such requirement reminds us of the connection between LSH and angle estimation, leading to the following claim: [ϑ-Space] Without loss of generality, let $\theta \in [0, \pi]$ be the angle between two vectors $\mathbf{w}_m, \mathbf{w}_n \in \mathbb{R}^d$, and $\mathscr{X}$ be the unit ball in the d-dimensional space. We then have

$$\vartheta \stackrel{def}{=} \mathbb{E}_{\mathbf{x} \sim \mathscr{X}} \left[(\mathbf{x}^T \mathbf{w}_m)(\mathbf{x}^T \mathbf{w}_n) \right] = 1 - \frac{2\theta}{\pi}, \tag{6.1}$$

where $\mathbb{E}$ is the expectation operator, sample $\mathbf{x}$ is uniformly sampled from $\mathscr{X}$, $: \mathbb{R} \to \{\pm 1\}$ is the sign function returning 1 for positives, otherwise -1. In fact $(\mathbf{x}^T \mathbf{w})$ defines a random hyperplane

based hash function for LSH (21). Therefore,

$$P_{\mathbf{x}\sim\mathscr{X}}\left[(\mathbf{x}^T\mathbf{w}_m)=(\mathbf{x}^T\mathbf{w}_n)\right]=1-\frac{\theta}{\pi}=\frac{1+\vartheta}{2}, \tag{6.2}$$

which is equivalent to Eq. 6.1. We then complete our proof.

6.2.1 Formulation

Lemma 6.2 opens a door for us to estimate filter angles (θ) using filter responses ($\mathbf{x}^T\mathbf{w}$). Due to linear transformation, both mean and variance in the θ-space, 90 and 0, are converted to 0 in the ϑ-space. Now based on the statistical relation between mean and variance, we can define our self-regularizer, $\mathscr{R}_\vartheta$, as follows:

$$\mathscr{R}_\vartheta \stackrel{def}{=} \sum_i \lambda_1 \mathbb{E}_{\vartheta\sim\Theta_i}(\vartheta)^2+\lambda_2\mathbb{E}_{\vartheta\sim\Theta_i}(\vartheta^2), \tag{6.3}$$

where $\Theta_i, \forall i$ denotes the filter angle pool in the ϑ-space at the i-th hidden layer, and $\lambda_1, \lambda_2 \geq 0$ are two predefined constants. Here we choose the least square loss for its simplicity, and other proper loss functions can be employed as well. In our experiments we choose $\lambda_1 = 100, \lambda_2 = 1$ by cross-validation using grid search. We observe that λ_2 has much larger impact than λ_1 on performance, achieving similar accuracy with $\lambda_1 \in [0, 1000]$ and $\lambda_2 \in [1, 10]$. This makes sense because the mean of an angular distribution in deep learning is close to 90 anyway, but the variance should be small.

6.2.2 Implementation

Computing ϑ in Eq. 6.1 is challenging because of the expectation over all possible samples in $\mathscr{X}$. To approximate ϑ, we introduce the notion of normalized approximate binary activation as follows: [Normalized Approximate Binary Activation[3] (NABA)] Letting $\mathbf{w} \in \mathbb{R}^d$ be a vector and

[3]We tested different sign approximation functions such as softsign, and observed that their performances are very close. Therefore, by referring to (20) in this paper we utilize tanh only.

rt

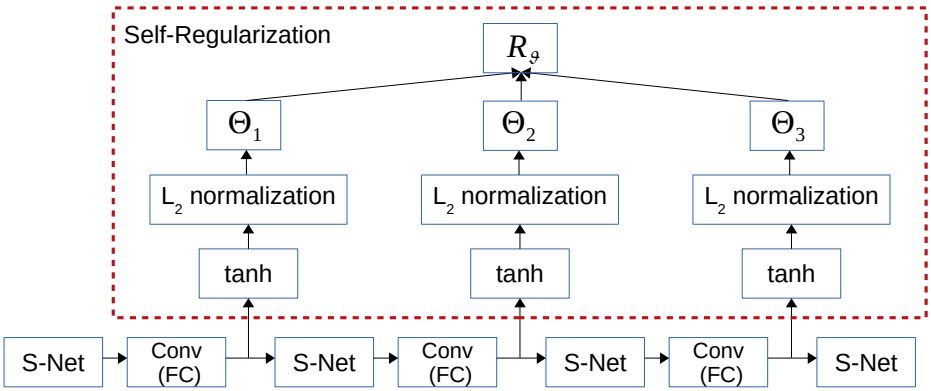

Figure 6.2: An example of integration of our self-regularization as a plug-in on a three hidden-layer DNN as backbone for training. Here "S-Net" denotes a sub-network consisting of non-hidden layers.

$\mathbf{X} \in \mathbb{R}^{d \times D}$ be a projection matrix, then we define an NABA vector, $\mathbf{z} \in \mathbb{R}^D$, for $\mathbf{w}$ as

$$\mathbf{z} = \frac{\tanh\left(\gamma \mathbf{X}^T \mathbf{w}\right)}{\|\tanh(\gamma \mathbf{X}^T \mathbf{w})\|} \Rightarrow \vartheta(\mathbf{w}_m, \mathbf{w}_n) \approx \mathbf{z}_m^T \mathbf{z}_n, \forall m \neq n \tag{6.4}$$

where tanh is an entry-wise function, and $\gamma \geq 0$ is a scalar so that $\lim_{\gamma \to +\infty} \tanh\left(\gamma x\right) = (x), \forall x \in \mathbb{R}$ as used in (20).

Based on the consideration of accuracy and running speed, in our experiments we set $\gamma = 10, D = 16$ for Eq. 6.4. Larger γ brings more difficulty in optimization, as demonstrated in (20). Larger D does approximate the expectation in Eq. 6.1 better, but has marginal effect on training loss and test accuracy, as well as leading to higher computational burden.

Implementing Eq. 6.4 using networks is simple, as illustrated in Fig. 6.2 where now $\mathbf{X}$ denotes the input features to each convolutional (conv) or fully-connected (FC) hidden layer and $\mathbf{w}$ denotes the weights of a filter at the hidden layer. In this way our regularization can be easily integrated with an arbitrary network seamlessly as a plug-in so that the network itself can automatically regularize its weights internally and explicitly, leading to self-regularization.

Compared with existing OR algorithms, our self-regularization takes the advantage of the dependency between input features and filters so that the weights are learned more specifically to better fit the data. From this perspective, our self-regularization is similar to a family of batch normalization algorithms. As a consequence, we may not need any external regularization to help

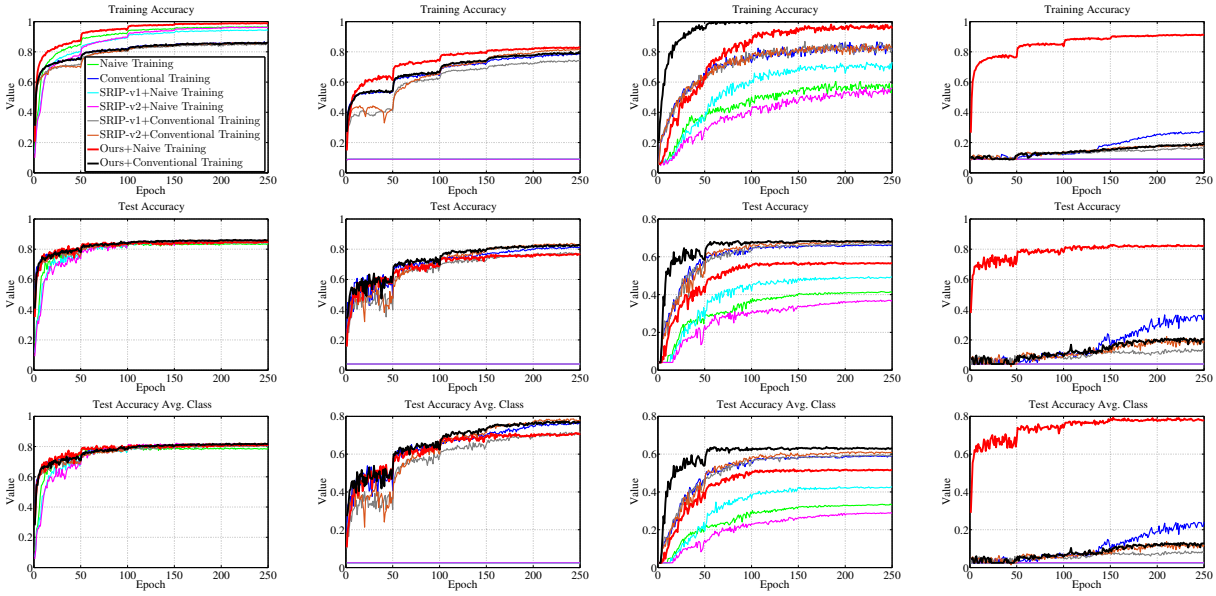

Figure 6.3: Result comparison on ModelNet40: **(left->right)** default setting, $lr = 0.01$, $ts = 440$, and $bs = 2$.

	default setting		$lr = 0.01$		$ts = 440$		$bs = 2$		ave. of 4 settings	
	avg. cls	overall	avg. cls	overall	avg. cls	overall	avg. cls	overall	avg. cls	overall
NT	0.800	0.840	0.025	0.041	0.336	0.416	0.025	0.041	0.297	0.335
SRIP-v1+NT	0.815	0.849	0.025	0.041	0.428	0.495	0.025	0.041	0.340	0.357
SRIP-v2+NT	**0.822**	**0.858**	0.025	0.041	0.290	0.371	0.025	0.041	0.309	0.328
Ours+NT	0.814	0.850	**0.712**	**0.770**	**0.520**	**0.571**	**0.793**	**0.829**	**0.732**	**0.755**
CT	0.819	0.860	0.767	0.818	0.591	0.666	**0.242**	**0.369**	**0.642**	**0.678**
SRIP-v1+CT	0.812	0.856	0.719	0.780	0.598	0.673	0.088	0.143	0.554	0.613
SRIP-v2+CT	**0.824**	**0.863**	**0.787**	**0.839**	0.612	0.682	0.138	0.223	0.590	0.652
Ours+CT	0.821	0.862	0.775	0.832	**0.638**	**0.685**	0.132	0.214	0.620	0.648

Table 6.1: Best test accuracy comparison on ModelNet40.

training.

6.3 Experiments

We study the impact of OR on deep learning using the task of point cloud classification.

Learning Scenarios: In order to identify the gain of OR, we intentionally design some extreme learning scenarios where we only change one hyper-parameter from the default setting while keeping the rest unchanged, and list them as follows:

- *Default Setting:* Under this setting, we train each model using the full training data set and then test it on the full testing data set. We keep all the hyper-parameters unchanged in the public

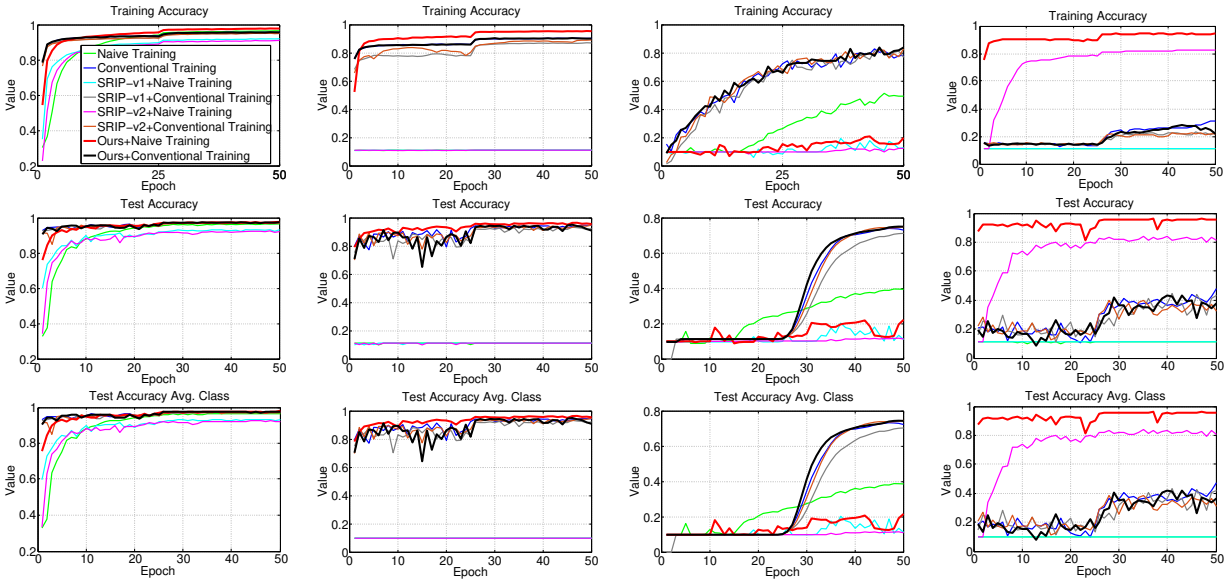

Figure 6.4: Result comparison on MNIST: **(left->right)** default setting, $lr = 0.01$, $ts = 200$, and $bs = 2$.

	default setting		$lr = 0.01$		$ts = 200$		$bs = 2$		ave. of 4 settings	
	avg. cls	overall	avg. cls	overall	avg. cls	overall	avg. cls	overall	avg. cls	overall
NT	0.977	0.977	0.967	0.967	**0.398**	**0.389**	0.114	0.100	0.614	0.608
SRIP-v1+NT	0.934	0.934	0.934	0.933	0.204	0.204	0.114	0.100	0.547	0.543
SRIP-v2+NT	0.928	0.927	0.928	0.927	0.119	0.116	0.928	0.927	0.726	0.724
Ours+NT	**0.978**	**0.978**	**0.978**	**0.978**	0.223	0.217	**0.963**	**0.963**	**0.786**	**0.784**
CT	0.967	0.967	0.976	0.976	0.741	0.734	**0.481**	**0.470**	**0.791**	**0.787**
SRIP-v1+CT	0.975	0.975	0.975	0.975	0.713	0.705	0.222	0.206	0.721	0.715
SRIP-v2+CT	0.974	0.973	0.974	0.973	0.751	**0.748**	0.181	0.166	0.720	0.715
Ours+CT	**0.976**	**0.976**	**0.977**	**0.977**	**0.752**	0.746	0.433	0.421	0.785	0.780

Table 6.2: Best test accuracy comparison on 2D point clouds of MNIST.

code.

- *Large Initial Learning Rate (lr):* Here we only change the initial learning rate to 0.01.
- *Limited Training Samples (ts):* Here we only use small number of training samples uniformly selected from the entire training set at random.
- *Small Batch Size (bs):* Here we only change the batch size to 2.

Baselines: In order to do comparison fairly, we consider the OR algorithms whose code is publicly available and can run in our task. In this sense, we finally have the following baseline algorithms[4]:

- *Naive Training (NT):* In this baseline, we train a network without any regularization.

[4]We fail to integrate OrthoReg (134) with our point cloud classification task, neither using their code nor reimplementing it, because we realize that it is so difficult to modify SGD to achieve reasonable performance or even make it work.

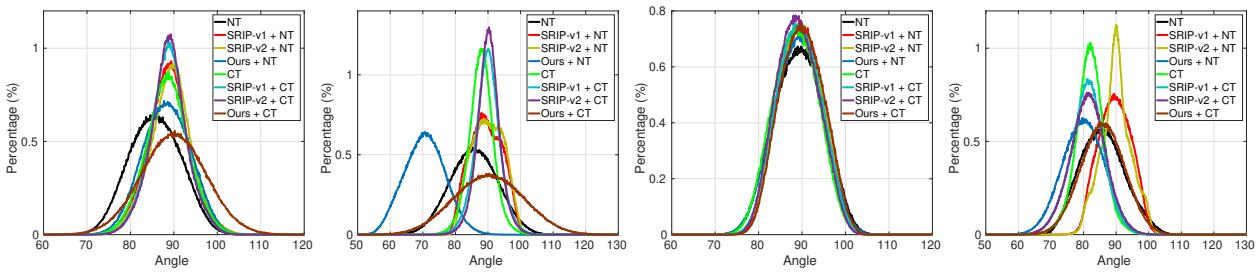

Figure 6.5: Comparison of the learned angular distributions on ModelNet40: **(left->right)** default setting, $lr = 0.01$, $ts = 440$, and $bs = 2$.

- *Conventional Training (CT):* In this baseline, we train a network with weight decay (5e-4), dropout (keep-ratio 0.7), and batch normalization (BN).
- *Spectral Restricted Isometry Property (SRIP-v1 & SRIP-v2) (4)* [5]*:* This regularizer is defined as $\mathscr{R} = \lambda \cdot \sigma\left(\mathbf{W}^T\mathbf{W} - \mathbf{I}\right)$, where $\mathbf{W}$ is the network weight matrix, $\mathbf{I}$ is an identity matrix, $\sigma(\cdot)$ is the spectral norm, and λ is a constant. Currently this is the state-of-the-art OR in the literature.

In the sequel we will present our results for different types of networks on different data sets.

6.3.1 Multilayer Perceptron (MLP): PointNet

Data Sets: We conduct comparison on two benchmark data sets, ModelNet40 (169) and 2D point clouds of MNIST (89). ModelNet40 (169) has 12,311 CAD models for 40 object categories, and is split into 9,840 for training and 2,468 for testing. We uniformly sample 1024 points from meshes to obtain 3D point clouds, and normalize them into a unit ball. MNIST is a handwritten digit data sets, consisting of 60,000 training images and 10,000 testing images with $28 \times 28 = 784$ gray pixels. We convert each image to 2D point clouds by taking image coordinates of all non-zero pixels.

Networks: We choose PointNet-vanilla (124) (an MLP(64, 64, 64, 128, 1024, 512, 256, 40) without T-nets) as the backbone network and integrate different regularization techniques with it to verify the effects of OR. During training we conduct data augmentation on-the-fly by randomly

[5]From `https://github.com/nbansal90/Can-we-Gain-More-from-Orthogonality/`, we find that there are two implementations of the approach, under the folders "Wide-Resnet" and "SVHN", respectively. We therefore name them as "v1" and "v2".

rotating the points along the up-axis and jittering the coordinates of each point by a Gaussian noise with zero mean and 0.01 std. We use the PyTorch code from `https://github.com/meder411/PointNet-PyTorch` as our testbed and the same training and evaluation protocols as the original PointNet code. We tune each approach to report the best performance.

In order to demonstrate that our findings are common across different MLP, on MNIST we slightly modify the architecture of original PointNet to another MLP(64, 64, 64, 128, 512, 256, 128, 10).

Training Stability: We summarize the training and testing behavior of each approach on ModelNet40 in Fig. 6.3. **(a)** Under the default setting, our regularizer help the naive training converge much faster than the others which are appreciated, while with the conventional training it seems that such effect is neutralized by the other regularization. In testing, the overall behavior of each approach is similar to each other with no significant performance gap. **(b)** Under the setting of $lr = 0.01$, naive training and both SRIPs fail to work, while our regularizer still works reasonably well so that it still converges faster than the others in training and with conventional training, it achieves the best performance. **(c)** Under the setting of $ts = 440$, the performances of different algorithms differ significantly, even though all the competitors work. Our regularizer with conventional training achieves the best performance, converging around 100 epochs which is much faster than the others. Our regularizer outperforms SRIP significantly as well. **(d)** Under the setting of $bs = 2$, only our regularization with naive training works, and surprisingly achieves the very good performance similar to that under the default setting. For conventional training, the regularization dominates the training behavior so strongly that even our regularizer cannot make it work. This is expected as usually conventional regularization techniques cannot work well using very small batch size. In Fig. 6.4 we summarize the training and testing behavior of each approach on MNIST. Similar observations can be made here in both training and testing.

In summary, for MLP we do not observe any significant gain on accuracy using SRIP, statistically speaking on average, but some improvement on convergence in training under the default setting. Our self-regularization, however, improves both naive training and conventional train-

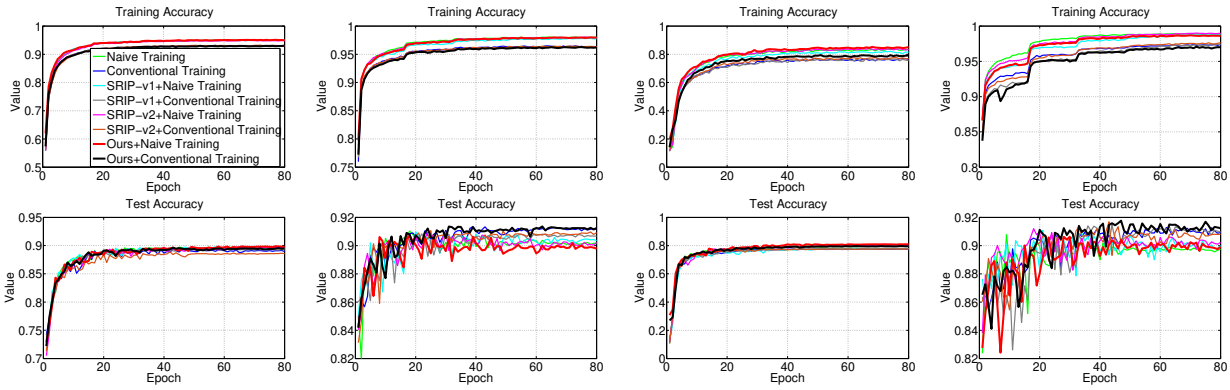

Figure 6.6: Result comparison on ModelNet10: **(left->right)** default setting, $lr = 0.01$, $ts = 400$, and $bs = 2$.

ing significantly on both convergence and accuracy, indicating better training stability from our method.

Accuracy: We summarize in Table 6.1 and Table 6.2 the best accuracy of each approach per setting in Fig. 6.3 and Fig. 6.4, respectively. Compared with other regularization techniques, we conclude that our regularizer can work well not only under well-defined setting but also under extreme learning cases. For instance, the naive training with our self-regularization works as well as, or even better than, the conventional training. These observations imply that our regularizer has much better generalization ability for optimizing deep networks, potentially leading to broader applications.

Angular Distributions: We illustrate the learned angular distributions in Fig. 6.5. First of all, it seems that all the well-trained models under the default setting form Gaussian-like distributions with mean around 90. The variances of these distributions, however, are larger than those in Fig. 6.1. We conjecture that the main reason for this is that in PointNet the input dimension is usually smaller than the number of filters per hidden layer so that it is hard to achieve (near) orthogonality among all the filters. Next, our self-regularization with naive training always learns Gaussian-like distributions, while conventional training sometimes has negative effects on our learned distributions such as shifting. This is probably due to BN as it normalizes the statistics in gradients. Finally, it seems that spike-shape distributions tend to produce bad models. This observation has also been made in the angular distributions with random weights.

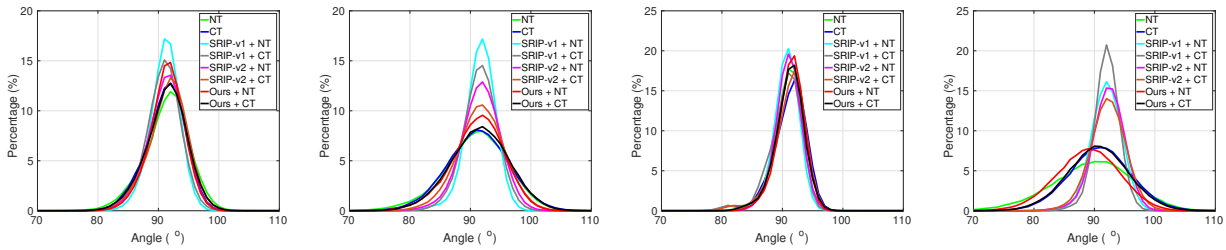

Figure 6.7: Comparison of the learned angular distributions on ModelNet10: **(left->right)** default setting, $lr = 0.01$, $ts = 400$, and $bs = 2$.

	def. s.	$lr(0.01)$	$ts(400)$	$bs(2)$	ave.
NT	0.898	**0.910**	0.797	0.908	0.878
SRIP-v1+NT	0.898	**0.910**	0.781	0.911	0.875
SRIP-v2+NT	0.897	**0.910**	0.800	**0.912**	0.880
Ours+NT	**0.899**	0.907	**0.810**	0.906	**0.881**
CT	0.894	**0.914**	0.776	0.913	0.874
SRIP-v1+CT	0.894	0.909	0.779	0.915	0.874
SRIP-v2+CT	0.889	0.911	0.776	0.917	0.873
Ours+CT	**0.896**	**0.914**	**0.796**	**0.918**	**0.881**

Table 6.3: Best test accuracy comparison on ModelNet10.

6.3.2 Convolutional Neural Networks (CNNs): VoxNet

Data Set: We use Modelnet10 (111) for our comparison. In the data set there are 3D models as well as voxelized versions, which have been augmented by rotating in 12 rotations. We use the provided voxelizations and follow the train/test splits for evaluation.

Networks: We use VoxNet (111) for comparison, which is a network architecture to efficiently dealing with large amount of point cloud data by integrating a volumetric occupancy grid representation with a supervised 3D CNN. We use the PyTorch code from `https://github.com/Durant35/VoxNet` as our testbed, and tune each approach to report the best accuracy averaged per class.

Training Stability & Accuracy: We illustrate the training and testing behavior of each algorithm on ModelNet10 in Fig. 6.6. As we see, unlike for PointNet, there is no significant difference in both training and testing for VoxNet. To further verify the performance, we summarize the best test accuracy in Table 6.3, where our improvement is marginal. Interestingly, we find similar observations in image classification that all the OR algorithms perform equally well, and no big advantage over conventional training with careful tuning. For instance, we list our classification

	lr	acc.
NT	0.001	0.888
SRIP-v1+NT	0.001	0.893
OrthoReg+NT	-	-
Ours+NT	0.001	**0.898**
CT	0.1	0.963
SRIP-v1+CT	0.1	0.962
OrthoReg+CT	0.1	0.962
Ours+CT	0.1	**0.965**

Table 6.4: Test accuracy comparison on CIFAR-10. Here "-" indicates that we cannot make OrthoReg work.

results on CIFAR-10 (82) in Table 6.4.

In summary, we observe that OR is more useful for MLP than for CNNs to improve the training stability. We believe that one of the key reasons is because the filters in CNNs are much better structured due to the input data such as images and 3D volumes, so that learning orthogonal filters becomes unnecessary. In contrast, the input data for MLP is much less structured individually where orthogonal filters can better cover the feature space. In both cases, however, our self-regularization outperforms the state-of-the-art OR algorithms.

Angular Distributions: We also illustrate the angular distributions of learned models on MondelNet10 in Fig. 6.7. Again all the distributions form Gaussian-like shapes with mean close to 90 and relatively small variance. Together with Fig. 6.5, we conclude that angular distributions may not be a good indicator for selecting good deep models, but their statistics are good for self-regularization.

6.4 Conclusion

In this paper, we manage to identify that the real gain of orthogonality regularization (OR) in deep learning is to better stabilize the training, leading to faster convergence and better generalization. In terms of accuracy, existing OR algorithms perform no better than the conventional training algorithm with weight decay, dropout, and batch normalization, statistically speaking. Instead, we propose a self-regularization method as an architectural plug-in that can be easily integrated with an arbitrary network to learn orthogonal filters. We utilize LSH to compute the filter angles approximately based on the filter responses, and push the mean and variance of such angles towards

90° and 0°, respectively. Empirical results on point cloud classification demonstrate the superiority of self-regularization.

Chapter 7

Miti-Detr: Object Detection based on Transformers with Mitigatory Self-Attention Convergence

Object Detection with Transformers (DETR) and related works reach or even surpass the highly-optimized Faster-RCNN baseline with self-attention network architectures. Inspired by the proof that pure self-attention possesses a strong inductive bias that leads to transformer losing the expressive power with respect to network depth, we propose the transformer architecture with a mitigatory self-attention mechanism by applying possible direct mapping connections in self-attention networks to mitigate the convergence to rank collapse so as to counteract feature expression loss and enhance model performance. We apply this proposal in object detection tasks and provide the model named Miti-Detr. Miti-Detr reserves the inputs of each single attention layer to the outputs of this layer so as the "non-attention" information has participated in any attention propagation. This formed residual self-attention network that helps address two critical issues: (1) stop the self-attention networks from degenerating to rank-1 to the maximized degree; (2) further diversify the path distribution of parameter update so that easier attention learning is expected. Miti-Detr demonstrates significant enhancement of average detection precision and convergence

speed towards existing DETR based models on the challenging COCO object detection dataset. Moreover, the proposed transformer with residual self-attention network could be easily generalized or plugged in other related task models without specific customization.

7.1 Introduction

The attention mechanism has been found effective use in transformer networks (158), not only in the application of long-range sequential knowledge, such as natural language processing (33), speech recognition (103), but also in computer vision tasks (17; 30; 197), where DETR has achieved competitive performance as an end-to-end object detector (17). Attention mechanism, transformer networks and DETR, thus, have become the research focuses, where the inner workings of transformers and attention, the training and optimization challenge of DETR, etc, have been regarded shedding light for the future works.

The attention-mechanism based transformer networks realize the the most generalized deep learning model in terms of computer vision and image tasks. For transformer networks, one pixel in an image cares about all the other pixels in that image so that any single region obtains and integrates relevance with all other regions, which is in comparison with CNN that any one pixel cares about its immediate neighbourhood and then what the neighbourhood as a whole cares about is its immediate neighbourhood. This could be a good explanation why DETR can be on par with state-of-the-art classifier in terms of classification accuracy. It also explains the strong inductive bias of self-attention. While the research in (35) demonstrates that pure self-attention networks (SANs) would lead to the loss of expressive power doubly exponentially with respect to network depth, and the output converges with a cubic rate to a rank one matrix that has identical rows.

Inspired by the analysis that skip connections play a key role in mitigating rank collapse in transformer, we propose the Miti-Detr detector model where residual self-attention network architecture is introduced. Specifically, the inputs of a multi-head self-attention layer are short connected to its outputs, as shown in Figure 7.1. This connection skips the Multi-layer perceptions (MLP), which is usually considered rendering the model less sensitive to its input perturba-

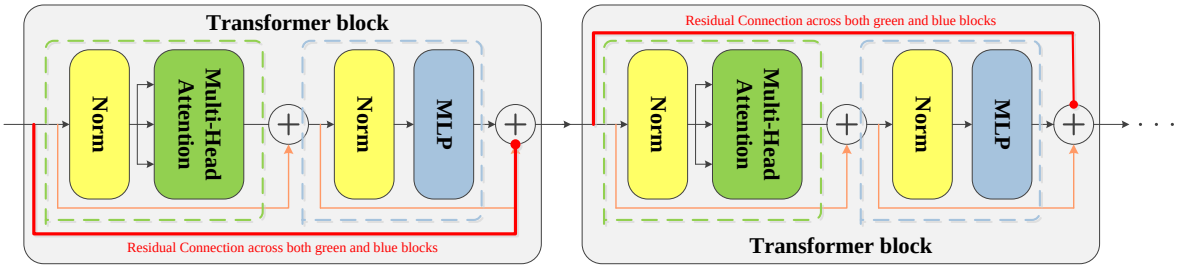

Figure 7.1: Residual Attention Network in Transformer

tions (27). Thus, the relative "non-attention" features are integrated to avoid the outputs dramatically converge to rank one matrix. Generally, this work provides two critical contributions to the current Detr models:

- (1) The training and optimization challenges of DETR could be better solved based on the network model itself, which is data-independent and could be easily applied in related models. It is verified the proposed model could significantly speed up the training convergence so as to avoid the extremely long time training schedule.

- (2) Further enhance the object detection performance of DETR models. From the perspective of models themselves, we address the limitations of DETR models whose transformer networks tend to lose efficient feature expression power by proposing the residual self-attention network. Thus, the proposed Miti-Detr could better stop the outputs from degeneration and gain nearly 3% higher performance than the original DETR models.

We evaluate Miti-Detr on one of the most popular and challenging object detection datasets, COCO, and compare its performance with the traditional DETR-related models. Our experiments show that our model is capable of further enhancing the DETR models' current performance. Specifically, Miti-Detr brings the superiority of DETR detecting large objects into full play, results likely enabled by the protection of global expression power.

7.2 Related Work

This work builds on prior researches of attention mechanism (158), feature mapping and propagation (59) and object detection with transformer (17).

7.2.1 Attention Mechanism

Attention-based architectures have become ubiquitous in machine learning, which brings about the better learning for long-sequence and large-range knowledge (3) (125). They have permeated machine learning applications across data domains, such as natural language processing (33), speech recognition (103) and computer vision (9) (17). Attention mechanisms are neural network layers that aggregate information from the entire input sequence (3). They allow modeling of dependencies without regard to their distance in the input or output sequences (3) (77) and the early such attention mechanisms models mostly are applied in conjunction with the recurrent network (120).

Self-attention is an attention mechanism that relates different positions in a single sequence so as to compute a sequence representation (24) (98). End-to-end memory networks are based on a recurrent attention mechanism rather than sequence aligned recurrence, which shows advantage on simple-language question answering and language modeling tasks (145). Transformer, currently, is the first transduction model which entirely relies on self-attention to compute representations of its input and output without using sequence aligned RNNs or convolution (158). They introduce self-attention layers to Non-Local Neural Networks (164). One advantage of attention-based models is the global computations and superior memory, making them more suitable compared with RNNs on long sequences. Transformers now have shown its replacing role then RNNs in many problems in natural language processing, speech processing and computer vision (121) (147).

Recently, researchers find that pure self-attention networks (SANs), for example, transformers with skip connections and multi-layer perceptrons (MLPs) disabled, lose expressive power doubly exponentially with respect to network depth. They prove that the output converges with a cubic rate to a rank one matrix with identical rows (35). Their analysis verifies that skip connections are

the key in mitigating rank collapse, and MLPs can slow down the convergence by increasing their Lipschitz constant. This research inspires our deep thoughts towards the current object detection models with transformer. We try to excavate the inner specialty of transformer so as to solve the existing problem, especially in object detection applications.

7.2.2 Object Detection with Transformer

Previously, the mainstream object detection models make predictions relative to some initial guesses. Two-stage detectors (133) predict bounding boxes relative to proposals, and single-stage methods make predictions based on anchors (105) (104) (99) or other possible object key points (152) (195). While the performance of these models heavily depends on how the initial guesses are set (). There are Anchor-free detection technologies that assign positive and negative samples to feature maps by a grid of object centers (86).

DETR is recently proposed that successfully apply transformer in object detection that is conceptually simpler without handcrafted process by direct set prediction (17). DETR utilizes a simple architecture, by combining convolutional neural networks (CNNs) and Transformer encoder-decoders. Deformable DETR is proposed to improve the problems of slow-convergence and limited feature spatial resolution by making its attention modules only attend to a small set of key points around a reference (197). UP-DETR is inspired by the pre-training transformers in natural language processing and proposes the random query patch detection to unsupervisedly pre-train the transformer of DETR (30), which boosts the performance significantly. While these two models both solve the problem from the data end, one by ImageNet pre-training, the other through multi-scale feature representation. Compared with the previous works, Miti-DETR tries to solve the existing problems from the inner property of transformer so that it is data-independent and more practical.

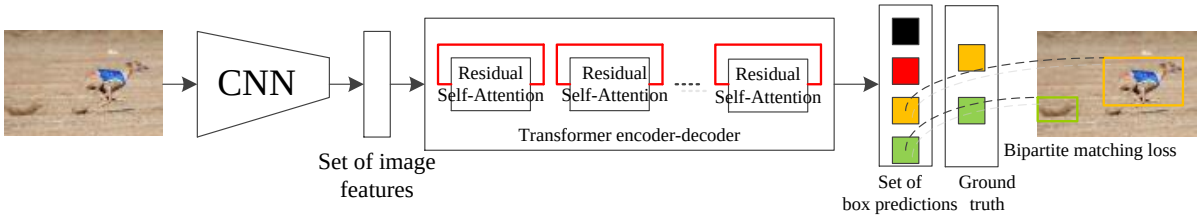

Figure 7.2: Architecture Streamline of Miti-DETR

7.3 Miti-DETR

The proposed Miti-DETR model maintains the pretty simple architecture as the original DETR, depicted in Figure 7.2. It contains three main components: a CNN working as the feature extractor; the transformer encoder and decoder with the proposed residual self-attention network; and the final feed-forward networks (FFN) working for the object detection prediction. We still adopt the effective bipartite matching loss for direct prediction (17). In this section, we mainly discuss why we need to introduce the proposed residual connection, and how to build up the corresponding transformer architecture, and finally, we show the proof why this new architecture could bring about mitigatory self-attention convergence.

7.3.1 Attention Network Loses Rank

The attention mechanism has become ubiquitous by its outstanding performance of learning long-range knowledge both in timing and spatial sequence (3) (158). However, it has been certified that the pure-attention networks (SANs), with the skip connections and multi-layer perceptions (MLPs) disabled, tend to lose expressive power with respect to network depth, leading to the output converging with cubic rate to a rank one matrix that has identical rows (35). This theorem can be expressed as below.

Theorem 2 *For any single-head SAN consisting of L layers where* $\|\boldsymbol{W}_{QK}^{l}\|_1\|\boldsymbol{W}_{V}^{l}\|_{1,\infty} \leq \beta$*, we have that*

$$\|res(SAN(\boldsymbol{X}))\|_{1,\infty} \leq \left(\frac{4\beta}{\sqrt{d_{qk}}}\right)^{\frac{3^L-1}{2}} \|res(\boldsymbol{X})\|_{1,\infty}^{3^L} \tag{7.1}$$

which amounts to a doubly exponential rate of convergence.

In Theorem 2, the residual item works as,

$$res(\boldsymbol{X}) = \boldsymbol{X} - 1\mathbf{x}^T, where\ \mathbf{x} = argmin_{\mathbf{x}}\|\boldsymbol{X} - 1\mathbf{x}^T\|, \tag{7.2}$$

$\boldsymbol{X}$ is the $n \times d_{in}$ input matrix consisting of n tokens. $\boldsymbol{W}_{QK}^{l}$ and $\boldsymbol{W}_{V}^{l}$ are the corresponding value weight matrixes. It is noted that the bound in Equation 7.3 guarantees $\|res(SAN(\boldsymbol{X}))\|_{1,\infty}$ convergence for all the small residual's inputs whenever $4\beta \leq \sqrt{d_{qk}}$. The detailed proofs can be found in (35).

Theorem 3 *Consider a depth L and width H self-attention network. Supposing* $\|\boldsymbol{W}_{QK,h}^{l}\|_1\|\boldsymbol{W}_{h}^{l}\|_{1,\infty} \leq \beta$ *for all heads* $h \in [H]$ *and layers* $l \in [L]$*, then we have*

$$\|res(SAN(\boldsymbol{X}))\|_{1,\infty} \leq \left(\frac{4\beta H}{\sqrt{d_{qk}}}\right)^{\frac{3^L-1}{2}} \|res(\boldsymbol{X})\|_{1,\infty}^{3^L} \tag{7.3}$$

which amounts to a doubly exponential rate of convergence.

From Theorem 3, it conforms the same convergence condition with Theorem 2 for multi-head self-attention network. The bound guarantees convergence of $SAN(\boldsymbol{X})$ to rank one when $4\beta H < \sqrt{d_q k}$.

7.3.2 Skip Connection and MLP Counteract Degeneration

There is a natural but pertinent question then: if the (pure) attention network degenerate to a rank one matrix doubly exponentially with depth, why do attention-based transformer networks work in practice? It is verified that the presence of skip connections is crucial that stops the SAN from

degenerating to rank one and the Multi-layer perceptrons (MLP) help control the convergence rate (35).

The lower bound on the residual to conform with the practice that SANs with skip connections neutralize the rank collapse is presented as below, which could be referred to with proof in (35).

Claim 1 *Given a depth L and width H self-attention network with skip connections. There exists infinitely many parameterizations for which* $\|res(\boldsymbol{X}^L)\| \geq \|res(\boldsymbol{X})\|$. *This holds even for* $L \to \infty$ *and* β *arbitrary small.*

Considering the MLP, we use f_l to denote the MLP as well as the output bias. Its corresponding output expression can be written as

$$\boldsymbol{X}^{l+1} = f_l\left(\sum_{h\in[H]} \boldsymbol{P}_h \boldsymbol{X}^l \boldsymbol{W}_h\right), \tag{7.4}$$

where P_h is the $n \times n$ row-stochastic matrix. W_h is the weights matrix.

We use $\lambda_{l,1,\infty}$ to denote the Lipschitz constant of f_l with respect to $l_{1,\infty}$ norm. The upper bound for the residual is derived in the following (35):

Corollary 1 *Consider a SAN with MLP of depth L and width H. Supposing that* $\|\boldsymbol{W}^l_{QK,h}\|_1 \|\boldsymbol{W}^l_h\|_{1,\infty} \leq \beta$ *for all* $h \in [H]$ *and* $l \in [L]$ *and fix* $\lambda_{l,1,\infty} \leq \lambda$, *then we have*

$$\|res(\boldsymbol{X}^L)\|_{1,\infty} \leq \left(\frac{4\beta H\lambda}{\sqrt{d_{qk}}}\right)^{\frac{3^L-1}{2}} \|res(\boldsymbol{X})\|^{3^L}_{1,\infty} \tag{7.5}$$

which amounts to a doubly exponential rate of convergence.

As seen in Corollary 1, the convergence rate can be controlled by the Lipschitz constants $\lambda_{f,1,\infty}$ of the MLPs, which indicates that more powerful MLPs bring about slower convergence. This reveals the tug-of-war between self-attention layers and the MLPs whose nonlinearity contributes to increasing the rank (35).

7.3.3 Residual Self-attention Network

By the observation towards the current transformer network, we find that the skip connections are across "independent modules". Here the "independent modules" refer to the multi-head self-attention network and the feed-forward fully connection network in transformer encoder and decoder layer architectures. Based on the analysis towards the functions of SANs, skip-connection and MLPs, we can define the features' attention levels and provide the sequence. Considering one single transformer layer, we use $\boldsymbol{X}^l$ to denote its inputs, $SAN(\boldsymbol{X}^l)$ to denote the outputs of SAN network, and $\boldsymbol{X}^{l+1}$ as the outputs after MLPs. If g is defined as the mapping function of features' attention level, we can derive the following conclusion,

$$g(SAN(\boldsymbol{X}^l)) \geq g(\boldsymbol{X}^{l+1}) \geq g(\boldsymbol{X}^l) \tag{7.6}$$

Combined with the advantage brought by skip connections, we believe that the skip connections that dramatically diversify the path distribution are the structural factor that explains the degeneration counteraction. This structural factor brings in the diversity of attention levels. Essentially, we believe that it is the diversity of features' attention levels that mitigate the strong inductive bias towards "token uniformity" of the self-attention networks. Inspired by this idea, we propose the residual self-attention network, in which we reserve the inputs of each single attention layer to the outputs of this layer so that the "non-attention" information could participate in the feature attention propagation. Thus, the diversity of feature attention levels is maximized. The corresponding architecture is depicted in Figure 7.1. In the sequence, we provide proof for the convergence property of the new self-attention network with the proposed residual connection.

The output of the l_{th} layer attention network, in this case, can be expressed as

$$\boldsymbol{X}^{l+1} = f_l\left(\sum_{h\in[H]} \boldsymbol{P}_h \boldsymbol{X}^l \boldsymbol{W}_h\right) + \boldsymbol{X}^l, \tag{7.7}$$

then the l_{th} layer output not considering the residual connection is,

$$\boldsymbol{X}^{l+1} - \boldsymbol{X}^{l} = f_l \left(\sum_{h \in [H]} \boldsymbol{P}_h \boldsymbol{X}^{l} \boldsymbol{W}_h \right). \tag{7.8}$$

We follow the definition of the residual in Theorem 2,

$$res(\boldsymbol{X}^{l+1} - \boldsymbol{X}^{l}) = (\boldsymbol{X}^{l+1} - \boldsymbol{X}^{l}) - 1\mathbf{x}^{T}, where\ \mathbf{x} = argmin_{\mathbf{x}} \| (\boldsymbol{X}^{l+1} - \boldsymbol{X}^{l}) - 1\mathbf{x}^{T} \|, \tag{7.9}$$

while the proposed residual connection skips both the multi-head attention layer and the MLPs, the actual output residual should be,

$$res(\boldsymbol{X}^{l+1}) = \boldsymbol{X}^{l+1} - 1\mathbf{x}^{T}, where\ \mathbf{x} = argmin_{\mathbf{x}} \| (\boldsymbol{X}^{l+1} - \boldsymbol{X}^{l}) - 1\mathbf{x}^{T} \|, \tag{7.10}$$

thus, $res(\boldsymbol{X}^{l+1})$ can also be expressed in the following equation

$$res(\boldsymbol{X}^{l+1}) = \boldsymbol{X}^{l} + \boldsymbol{\varepsilon}, where\ \boldsymbol{\varepsilon} < \boldsymbol{X}^{l} and \boldsymbol{\varepsilon} = argmin_{\boldsymbol{\varepsilon}} \| \boldsymbol{X}^{l+1} - \boldsymbol{X}^{l} - \boldsymbol{\varepsilon} \|. \tag{7.11}$$

With the introduced residual connection, $res(\boldsymbol{X}^{l+1})$ anchors at the inputs $\boldsymbol{X}^{l}$ of the current layer. Its fluctuation depends on the perturbance $\boldsymbol{\varepsilon}$ of the inputs and the original convergence performance of the attention network. We compare the above three equations, $res(\boldsymbol{X}^{l+1} - \boldsymbol{X}^{l})$ still satisfies the Corollary 1 by a convergence upper bound, and

$$\| res(\boldsymbol{X}^{l+1}) \| > \| res(\boldsymbol{X}^{l+1} - \boldsymbol{X}^{l}) \|. \tag{7.12}$$

Thus, the proposed residual self-attention network brings about an anchor for the convergence of each attention layer, which propagates through all attention layers in the transformer architecture and helps render the model keep sensitive to the input perturbation, which further slows down the convergence and counteracts the rank collapse. This research conclusion is verified by concrete learning performance in the experiment section.

7.3.4 Transformer with Mitigatory Convergence in Object Detection

Based on the above analysis, the proposed residual attention architecture enables the transformer to have mitigatory convergence performance theoretically. We follow the same architecture streamline as the DETR model but apply our proposed transformer network as the corresponding encoder-decoder transformer. Specifically, Midi-DETR consists of a convolutional CNN backbone, self-encoder and decoder attention network (17) and prediction feed-forward network (FFNs). In this work, we design the residual self-attention network in every transformer encoder and decoder layer, as the illustration shows in the transformer block in Figure 2. The residual connection bridges the inputs and outputs of a single transformer layer in a short connection way. This path bypasses the composite module of multi-head self-attention network and the feed-forward fully connection network, forming the new "non-attention" feature propagation. Then we concatenate the original inputs of the current layer and the outputs from the MLPs and normalization layer at the top of the transformer layer. Thus, this proposed new transformer network works as an independent module that can be implemented in any deep learning framework that provides a common CNN backbone and a transformer architecture implementation.

7.4 Experiments

We show that Miti-DETR achieves competitive results compared to the original DETR in quantitative evaluation on COCO (97). Then we provide a detailed analysis towards the training and learning progress, with insights and qualitative results. Then, we provide the detection accuracy results of Miti-DETR on COCO and shows its leading performance at multiple measuring criterion. To show the advantage of speeding up convergence and effective optimization, we will compare Miti-DETR and UP-DETR (30) and present the detection results and analysis as well in the future work. The corresponding experimental settings and implementation details come below.

7.4.1 Implementation Details

We train the related models in the work with AdamW (101). The transformer weights are initialized with Xavier init (49) and the backbone is the ImageNet-pretrained ResNet model (59). In this work, we report the results with the backbone of ResNet-50, which is a relatively basal option. All the other hyperparameters in this experiment strictly follow the setting of DETR (17).

We use the training schedule of 300 epochs with a learning rate drop of 10 after 200 epochs. All the training images are passed over the model for a single epoch. We train the related models on 4 P100 GPUs. We apply the pretrained backbone network of DETR R50 provided by DETR official code at https://github.com/facebookresearch/detr.

7.4.2 Dataset

We conduct the experiments on COCO2017 detection dataset (97), containing 118K training images and 5K validation images. Averagely, there are 7 instances, up to 63 instances in a single image in the training set, including small and large objects on the same images (17). We report Average Precision (AP) as bounding box AP, under the integral metric with multiple thresholds. For the comparison with state-of-the-art models, we report the validation AP from the highest epoch performance.

7.4.3 Training and Learning

Typically, transformers are trained with Adam or Adagrad optimizers with very long training schedules and dropout, which is true for DETR models as well. Despite this disadvantage, Miti-DETR is designed to make a faster transformer-based detector. We report the training loss curves and test error curves of all the experimental models during the learning process.

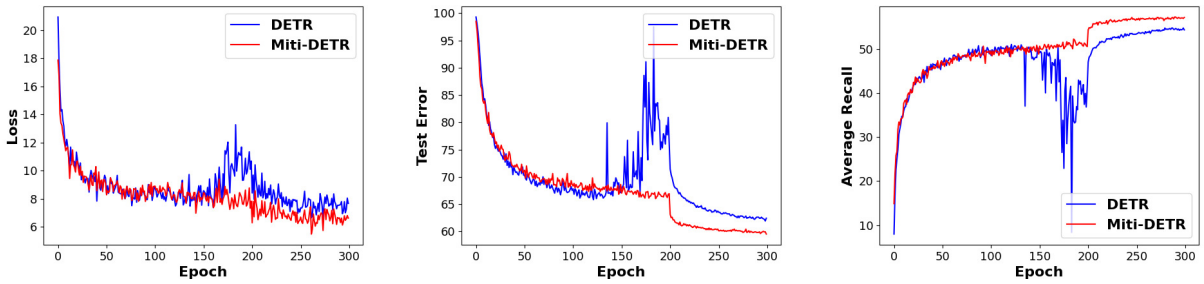

Figure 7.3: Comparison of the learning process distributions on COCO: **(left->right)** Learning Loss, Test Error, and Average Recall.

7.4.4 Detection Results

Set up. The models in the experiment settings are fine-tuned on COCO train2017 (approximate of 118k images) and evaluated on val2017. A comprehensive comparison, including AP, AP_{50}, AP_{75}, AP_S, AP_M and AP_L, is reported. Moreover, we also show the curves of training loss, test error, and average recall in the learning process and make a comparison among the related models.

Results. In Figure 7.3, Miti-DETR outperforms DETR for the entire learning process in terms of the convergence speed, showing a clear advantage, especially after the 150 epoch schedule. The statistic of test error in the middle figure in Figure 7.3 is the remaining value after deducting Average Precision (AP) from 1 at each epoch. Moreover, DETR shows unstable learning state, even divergence, between epoch 150 and epoch 200, the end of the first learning rate schedule. While Miti-DETR appears very stable convergence procedure during the entire process and all of the loss, test error and recall curves show a consistent trend. This is in accord with the theory that the original transformer tends to rank collapse. Although DETR returns to the normal track of fast convergence after the 200 epoch schedule, the unstable state could be the reason that lowers down the final convergence performance. After reducing the learning rate at epoch 200, the Miti-DETR averagely keeps the 0.9 test error of that of DETR. A similar situation applies to the average recall curves. This experiment suggests that the proposed residual self-attention network could effectively improve the convergence problem of DETR, where the unstable learning procedure caused by the rank-1 trend is avoided by the residual connection across composite network modules in transformer.

Model	Backbone	Epoch	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR	R50	300	37.6	57.8	39.3	18.0	40.6	55.7
Miti-DETR	R50	300	40.5	60.4	42.7	19.7	43.9	59.3

Table 7.1: Detection accuracy on COCO.

Table 7.1 shows the AP statistic results of both DETR and the proposed Miti-DETR. As shown in this table, Miti-DETR generally leads DETR by 3% in terms of AP. For AP_{75}, the highest threshold, Miti-DETR even surpasses DETR by nearly 4%. This indicates that Miti-DETR generally prominently improves the detection quality of DETR. More detected bounding boxes have higher IoU with the ground truth. Considering the size property of objects, Miti-DETR shows all-round advantage over DETR, ranging from small, medium, and large thresholds. Although Miti-DETR has no more than 2% higher than DETR in terms of AP_S, this is still a big enhancement considering small objects are hard to handle for current object detectors, and it's an existing problem of DETR as well. Combined with Figure 7.3, it is worth noting the stable learning process effectively contributes to the final detection accuracy of DETR and the Miti-DETR shows a promising research direction for future research based on transformer network itself.

7.5 Conclusion

We presented Miti-DETR, a DETR object detector based on Transformer with mitigatory self-attention convergence. The model outperforms DETR by a big advantage. The proposed core technique, residual self-attention network is verified capable of preventing the attention network from losing rank based on the transformer network itself. Miti-DETR is straightforward to implement and has a flexible structure where the residual self-attention network could be extensible to other attention-mechanism models or tasks. In addition, it achieves significantly better performance on small objects than DETR, likely thanks to the effective processing towards global information and the stable convergence procedure. We hope this work could be inspiring for further research towards the working principles of attention mechanism and transformer network, especially for the effective and efficient processing towards the global attention. Accordingly,

the current models could be more productive with the training process and a more comprehensive method of combining global information and local features based on the transformer network itself is expected.

Chapter 8

Conclusion and Future Work

For the research towards deep learning based object detection and classification, we have been working on better handling hard scenes, easier detecting small objects, more straightforward generalization and further intelligence. I have been working on improving the above problems from the perspectives of feature learning, semantic understanding, cognitive reasoning, regularization. We believe this research is based on network working theory itself, which would obtain maximum adaptability. Thus, my research starts from feature learning, especially for the extraction and representation of contextual information and abstract semantic. So we design and propose the deep feature learning module which explores the relationship semantic between objects and scenes so that the detectors obtain better processing over small objects. From the perspective of feature learning, I believe the essence is to maximize the combination of abstract and detailed features.

Since the CNN networks are good at extracting details, I focus on the attention mechanism and transformer network. They realize the most generalized compute model in terms of deep learning literature, which attends global information effectively. We propose the Miti-DETR and the residual transformer network and solved the unstable learning procedure and rank-1 convergence problems of current DETR models. Considering DETR related models currently have the most reasonable network architecture, with CNN as the backbone and Transformer as the global correlation module, I will continue my current work in searching for the solutions to the training

struggle of transformer and excavating the attention mechanism for lighter correlation computation and exploring techniques for better detection towards small objects. In addition to feature learning, algorithm design, another key point in my research is cognitive learning. We propose the first-step work by presenting the semantic clustering-based deduction learning, where the classification model is guided to notice and find the high-level semantic relations in semantic space, enabling the model to deduce the semantic knowledge at the cognitive level. I believe cognitive learning will be a developing trend in the deep learning literature and I will focus on this field in my future work.

References

[1] Akata, Z., Perronnin, F., Harchaoui, Z., & Schmid, C. (2015). Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7), 1425–1438.

[2] Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[3] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[4] Bansal, N., Chen, X., & Wang, Z. (2018). Can We Gain More from Orthogonality Regularizations in Training Deep CNNs? In *NIPS*.

[5] Bansal, N., Chen, X., & Wang, Z. (2018). Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems* (pp. 4261–4271).

[6] Baroni, M. & Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4), 673–721.

[7] Bartlett, P. L., Jordan, M. I., & McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473), 138–156.

[8] Bell, S., Lawrence Zitnick, C., Bala, K., & Girshick, R. (2016). Inside-outside net: Detecting

objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2874–2883).

[9] Bello, I., Zoph, B., Vaswani, A., Shlens, J., & Le, Q. V. (2019). Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 3286–3295).

[10] Bellver, M., Giro-i Nieto, X., Marques, F., & Torres, J. (2016). Hierarchical object detection with deep reinforcement learning. In *Deep Reinforcement Learning Workshop, NIPS*.

[11] Bhagavatula, C., Zhu, C., Luu, K., & Savvides, M. (2017). Faster than real-time facial alignment: A 3d spatial transformer network approach in unconstrained poses. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 3980–3989).

[12] Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision* (pp. 5561–5569).

[Bottou] Bottou, L. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9), 142.

[14] Bottou, L. (2004). Stochastic learning. *Advanced lectures on machine learning*, (pp. 146–168).

[15] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177–186). Springer.

[16] Cai, Z., Fan, Q., Feris, R. S., & Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision* (pp. 354–370).: Springer.

[17] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European Conference on Computer Vision* (pp. 213–229).: Springer.

[18] Cen, F. & Wang, G. (2019). Boosting occluded image classification via subspace decomposition-based estimation of deep features. *IEEE transactions on cybernetics*.

[19] Cen, F., Zhao, X., Li, W., & Wang, G. (2020). Deep feature augmentation for occluded image classification. *Pattern Recognition*, 111, 107737.

[20] Chang, X., Xiang, T., & Hospedales, T. M. (2017). Deep multi-view learning with stochastic decorrelation loss. *arXiv preprint arXiv:1707.09669*.

[21] Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing* (pp. 380–388).: ACM.

[22] Chen, Y., Kalantidis, Y., Li, J., Yan, S., & Feng, J. (2018). Aˆ 2-nets: Double attention networks. *Advances in Neural Information Processing Systems*, (pp. 350–359).

[23] Cheng, B., Wei, Y., Shi, H., Feris, R., Xiong, J., & Huang, T. (2018). Revisiting rcnn: On awakening the classification power of faster rcnn. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 453–468).

[24] Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

[25] Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.

[26] Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., & Batra, D. (2015). Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*.

[27] Cranko, Z., Kornblith, S., Shi, Z., & Nock, R. (2018). Lipschitz networks and distributional robustness. *arXiv preprint arXiv:1809.01129*.

[28] Cui, P., Liu, S., & Zhu, W. (2018). General knowledge embedded image representation learning. *IEEE Transactions on Multimedia*, 20(1), 198–207.

[29] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, (pp. 379–387).

[30] Dai, Z., Cai, B., Lin, Y., & Chen, J. (2021). Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1601–1610).

[31] Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1 (pp. 886–893).: IEEE.

[32] Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., & Adam, H. (2014). Large-scale object classification using label relation graphs. In *European conference on computer vision* (pp. 48–64).: Springer.

[33] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[34] Ding, Y., Wang, L., Fan, D., & Gong, B. (2018). A semi-supervised two-stage approach to learning from noisy labels. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1215–1224).: IEEE.

[35] Dong, Y., Cordonnier, J.-B., & Loukas, A. (2021). Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*.

[36] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[37] Dvornik, N., Shmelkov, K., Mairal, J., & Schmid, C. (2017). Blitznet: A real-time deep network for scene understanding. *Proceedings of the IEEE international conference on computer vision*, (pp. 4154–4162).

[38] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2007). The pascal visual object classes challenge 2007 (voc2007) results.

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

[40] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9), 1627–1645.

[41] Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A., & Berg, A. C. (2017). Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*.

[42] Fukushima, K. & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets* (pp. 267–285). Springer.

[43] Gao, J., Yang, J., Wang, G., & Li, M. (2016). A novel feature extraction method for scene recognition based on centered convolutional restricted boltzmann machines. *Neurocomputing*, 214, 708–717.

[44] Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*.

[45] Ghosh, A., Kumar, H., & Sastry, P. (2017). Robust loss functions under label noise for deep neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[46] Gidaris, S. & Komodakis, N. (2015). Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1134–1142).

[47] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1440–1448).

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

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

[50] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323).

[51] Gong, C., Zhang, H., Yang, J., & Tao, D. (2017). Learning with inadequate and incorrect supervision. In *2017 IEEE International Conference on Data Mining (ICDM)* (pp. 889–894).: IEEE.

[52] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

[53] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., & Wang, G. (2015). Recent advances in convolutional neural networks. *arXiv preprint arXiv:1512.07108*.

[54] Gu, S., Hou, Y., Zhang, L., & Zhang, Y. (2018). Regularizing deep neural networks with an ensemble-based decorrelation method. In *IJCAI* (pp. 2177–2183).

[55] Haeusser, P., Mordvintsev, A., & Cremers, D. (2017). Learning by association–a versatile semi-supervised training method for neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 89–98).

[56] Harandi, M. & Fernando, B. (2016). Generalized backpropagation,\'{E} tude de cas: Orthogonality. *arXiv preprint arXiv:1611.05927*.

[57] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017a). Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, (pp. 2961–2969).

[58] He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision* (pp. 346–361).: Springer.

[59] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770–778).

[60] He, L., Wang, G., & Hu, Z. (2018). Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 27(9), 4676–4689.

[61] He, Y., Ma, X., Luo, X., Li, J., Zhao, M., An, B., & Guan, X. (2017b). Vehicle traffic driven camera placement for better metropolis security surveillance. *arXiv preprint arXiv:1705.08508*.

[62] Hendrycks, D., Mazeika, M., Wilson, D., & Gimpel, K. (2018). Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in neural information processing systems* (pp. 10456–10465).

[63] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012a). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6), 82–97.

[64] Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7), 1527–1554.

[65] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[66] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017a). Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 4700–4708).

[67] Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep networks with stochastic depth. *European conference on computer vision*, (pp. 646–661).

[68] Huang, L., Liu, X., Lang, B., Yu, A. W., Wang, Y., & Li, B. (2017b). Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. *arXiv preprint arXiv:1709.06079*.

[69] Huang, Z., Huang, L., Gong, Y., Huang, C., & Wang, X. (2019). Mask scoring r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6409–6418).

[70] Hubel, D. H. & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1), 215–243.

[71] Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[72] Ishida, T., Niu, G., Hu, W., & Sugiyama, M. (2017). Learning from complementary labels. In *Advances in neural information processing systems* (pp. 5639–5649).

[73] Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al. (2009). What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision* (pp. 2146–2153).: IEEE.

[74] Jia, K., Li, S., Wen, Y., Liu, T., & Tao, D. (2019). Orthogonal deep neural networks. *arXiv preprint arXiv:1905.05929*.

[75] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *Proceedings of the 22nd ACM international conference on Multimedia*, (pp. 675–678).

[76] Jiang, B., Luo, R., Mao, J., Xiao, T., & Jiang, Y. (2018). Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 784–799).

[77] Kim, Y., Denton, C., Hoang, L., & Rush, A. M. (2017). Structured attention networks. *arXiv preprint arXiv:1702.00887*.

[78] Kim, Y., Yim, J., Yun, J., & Kim, J. (2019). Nlnl: Negative learning for noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 101–110).

[79] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[80] Kong, T., Yao, A., Chen, Y., & Sun, F. (2016). Hypernet: towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 845–853).

[81] Krizhevsky, A. & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[82] Krizhevsky, A., Nair, V., & Hinton, G. (2014). The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*.

[83] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

[84] Kukačka, J., Golkov, V., & Cremers, D. (2017). Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*.

[85] Kuo, C.-C. J., Zhang, M., Li, S., Duan, J., & Chen, Y. (2018). Interpretable Convolutional Neural Networks via Feedforward Design. *arXiv preprint arXiv:1810.02786*.

[86] Law, H. & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 734–750).

[87] Law, M. T., Yu, Y., Urtasun, R., Zemel, R. S., & Xing, E. P. (2017). Efficient multiple instance metric learning using weakly supervised data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 576–584).

[88] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4), 541–551.

[89] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[90] Lee, K.-H., He, X., Zhang, L., & Yang, L. (2018). Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5447–5456).

[91] Lei Ba, J., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[92] Li, K., Ma, W., Sajid, U., Wu, Y., & Wang, G. (2019). Object detection with convolutional neural networks. *arXiv preprint arXiv:1912.01844*.

[93] Li, K., Wang, N. Y., Yang, Y., & Wang, G. (2021). Sgnet: A super-class guided network for image classification and object detection. *arXiv preprint arXiv:2104.12898*.

[94] Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. (2018). Detnet: Design backbone for object detection. *Proceedings of the European Conference on Computer Vision (ECCV)*, (pp. 334–350).

[95] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117–2125).

[96] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

[97] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European Conference on Computer Vision* (pp. 740–755).: Springer.

[98] Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017c). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

[99] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European Conference on Computer Vision* (pp. 21–37).: Springer.

[100] Liu, W., Rabinovich, A., & Berg, A. C. (2015). Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*.

[101] Loshchilov, I. & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[102] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2 (pp. 1150–1157).: Ieee.

[103] Luo, H., Zhang, S., Lei, M., & Xie, L. (2021). Simplified self-attention for transformer-based end-to-end speech recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)* (pp. 75–81).: IEEE.

[104] Ma, W., Li, K., & Wang, G. (2020a). Location-aware box reasoning for anchor-based single-shot object detection. *IEEE Access*, 8, 129300–129309.

[105] Ma, W., Wu, Y., Cen, F., & Wang, G. (2020b). Mdfn: Multi-scale deep feature learning network for object detection. *Pattern Recognition*, 100, 107149.

[106] Ma, W., Wu, Y., Wang, Z., & Wang, G. (2018). Mdcn: Multi-scale, deep inception convolutional neural networks for efficient object detection. *2018 24th International Conference on Pattern Recognition (ICPR)*, (pp. 2510–2515).

[107] Mahendran, A. & Vedaldi, A. (2015). Understanding deep image representations by inverting them. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 5188–5196).

[108] Marcus, G. (2020). The next decade in ai: Four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

[109] Martin, C. H. & Mahoney, M. W. (2018a). Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *arXiv preprint arXiv:1810.01075*.

[110] Martin, C. H. & Mahoney, M. W. (2018b). Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *arXiv preprint arXiv:1810.01075*.

[111] Maturana, D. & Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS* (pp. 922–928).: IEEE.

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

[113] Misra, I., Lawrence Zitnick, C., Mitchell, M., & Girshick, R. (2016). Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2930–2939).

[114] Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807–814).

[115] Najibi, M., Rastegari, M., & Davis, L. S. (2016). G-cnn: an iterative grid based object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2369–2377).

[116] Neumann, L., Zisserman, A., & Vedaldi, A. (2018). Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection.

[117] Nøkland, A. & Eidnes, L. H. (2019). Training neural networks with local error signals. In *International Conference on Machine Learning* (pp. 4839–4850).: PMLR.

[118] Ouyang, W., Wang, K., Zhu, X., & Wang, X. (2017). Chained cascade network for object detection. *Proceedings of the IEEE International Conference on Computer Vision*, (pp. 1938–1946).

[119] Papageorgiou, C. P., Oren, M., & Poggio, T. (1998). A general framework for object detection. In *Computer vision, 1998. sixth international conference on* (pp. 555–562).: IEEE.

[120] Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

[121] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., & Tran, D. (2018). Image transformer. In *International Conference on Machine Learning* (pp. 4055–4064).: PMLR.

[122] Paulus, R., Xiong, C., & Socher, R. (2017). A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

[123] Pinheiro, P. O., Lin, T.-Y., Collobert, R., & Dollár, P. (2016). Learning to refine object segments. In *European Conference on Computer Vision* (pp. 75–91).: Springer.

[124] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*.

[125] Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., & Shlens, J. (2019). Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*.

[126] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 2001–2010).

[127] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*.

[128] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788).

[129] Redmon, J. & Farhadi, A. (2017). Yolo9000: better, faster, stronger. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 7263–7271).

[130] Ren, J., Chen, X., Liu, J., Sun, W., Pang, J., Yan, Q., Tai, Y.-W., & Xu, L. (2017a). Accurate single stage detector using recurrent rolling convolution. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 5420–5428).

[131] Ren, L., Lu, J., Feng, J., & Zhou, J. (2017b). Multi-modal uniform deep learning for rgb-d person re-identification. *Pattern Recognition*, 72, 446–457.

[132] Ren, M., Zeng, W., Yang, B., & Urtasun, R. (2018). Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning* (pp. 4334–4343).: PMLR.

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

[134] Rodríguez, P., Gonzalez, J., Cucurull, G., Gonfaus, J. M., & Roca, X. (2016). Regularizing cnns with locally constrained decorrelations. *arXiv preprint arXiv:1611.01967*.

[135] Roy, D., Panda, P., & Roy, K. (2020). Tree-cnn: a hierarchical deep convolutional neural network for incremental learning. *Neural Networks*, 121, 148–160.

[136] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3), 1.

[137] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3), 211–252.

[138] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

[139] Sajid, U., Sajid, H., Wang, H., & Wang, G. (2020). Zoomcount: A zooming mechanism for crowd counting in static images. *IEEE Transactions on Circuits and Systems for Video Technology*.

[140] Salimans, T. & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS* (pp. 901–909).

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

[142] Shrivastava, A., Gupta, A., & Girshick, R. (2016). Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 761–769).

[143] Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep insides convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

[144] Stewart, R., Andriluka, M., & Ng, A. Y. (2016). End-to-end people detection in crowded scenes. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 2325–2333).

[145] Sukhbaatar, S., Szlam, A., Weston, J., & Fergus, R. (2015). End-to-end memory networks. *arXiv preprint arXiv:1503.08895*.

[146] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

[147] Synnaeve, G., Xu, Q., Kahn, J., Likhomanenko, T., Grave, E., Pratap, V., Sriram, A., Liptchinsky, V., & Collobert, R. (2019). End-to-end asr: from supervised to semi-supervised learning with modern architectures. *arXiv preprint arXiv:1911.08460*.

[148] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. *Thirty-First AAAI Conference on Artifical Intelligence*.

[149] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9).

[150] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 2818–2826).

[151] Tanaka, D., Ikami, D., Yamasaki, T., & Aizawa, K. (2018). Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5552–5560).

[152] Tian, Z., Shen, C., Chen, H., & He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9627–9636).

[153] Tychsen-Smith, L. & Petersson, L. (2017). Denet: Scalable real-time object detection with directed sparse sampling. In *Proceedings of the IEEE international conference on computer vision* (pp. 428–436).

[154] Tychsen-Smith, L. & Petersson, L. (2018). Improving object localization with fitness nms and bounded iou loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6877–6885).

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

[156] van Laarhoven, T. (2017). L2 regularization versus batch and weight normalization.

[157] Vapnik, V. & Vashist, A. (2009). A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6), 544–557.

[158] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

[159] Veit, A., Wilber, M., & Belongie, S. (2016). Residual networks are exponential ensembles of relatively shallow. *arXiv preprint arXiv:1605.06431*, 1(2), 3.

[160] Vorontsov, E., Trabelsi, C., Kadoury, S., & Pal, C. (2017). On orthogonality and learning recurrent networks with long term dependencies. In *ICML*.

[161] Wang, J., Tao, X., Xu, M., Duan, Y., & Lu, J. (2018a). Hierarchical objectness network for region proposal generation and object detection. *Pattern Recognition*, 83, 260–272.

[162] Wang, J. & Yang, Y. (2018). A context-sensitive deep learning approach for microcalcification detection in mammograms. *Pattern recognition*, 78, 12–22.

[163] Wang, S., Cheng, J., Liu, H., Wang, F., & Zhou, H. (2018b). Pedestrian detection via body-part semantic and contextual information with dnn. *IEEE Transactions on Multimedia*, 20, 3148–3159.

[164] Wang, X., Girshick, R., Gupta, A., & He, K. (2018c). Non-local neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 7794–7803).

[165] Wu, B., Iandola, F., Jin, P. H., & Keutzer, K. (2017a). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, (pp. 129–137).

[166] Wu, S., Li, X., & Wang, X. (2020). Iou-aware single-stage object detector for accurate localization. *Image and Vision Computing*, (pp. 103911).

[167] Wu, Y., Sui, Y., & Wang, G. (2017b). Vision-based real-time aerial object localization and tracking for uav sensing system. *arXiv preprint arXiv:1703.06527*.

[168] Wu, Y., Zhang, Z., & Wang, G. (2019). Unsupervised deep feature transfer for low resolution image classification. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*.

[169] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *CVPR*.

[170] Xiang, Y., Choi, W., Lin, Y., & Savarese, S. (2017). Subcategory-aware convolutional neural networks for object proposals and detection. *2017 IEEE winter conference on applications of computer vision (WACV)*, (pp. 924–933).

[171] Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

[172] Xie, D., Xiong, J., & Pu, S. (2017a). All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *CVPR* (pp. 6176–6185).

[173] Xie, P., Poczos, B., & Xing, E. P. (2017b). Near-orthogonality regularization in kernel methods. In *UAI*.

[174] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017c). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1492–1500).

[175] Xing, E. P., Jordan, M. I., Russell, S. J., & Ng, A. Y. (2003). Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems* (pp. 521–528).

[176] Xu, C., Yang, J., Gao, J., Lai, H., & Yan, S. (2018). Srnn: Self-regularized neural network. *Neurocomputing*, 273, 260–270.

[177] Xu, W., Keshmiri, S., & Wang, G. (2019a). Adversarially approximated autoencoder for image generation and manipulation. *IEEE Transactions on Multimedia*, 21(9), 2387–2396.

[178] Xu, W., Shawn, K., & Wang, G. (2019b). Toward learning a unified many-to-many mapping for diverse image translation. *Pattern Recognition*, 93, 570–580.

[179] Xu, W., Wang, G., Sullivan, A., & Zhang, Z. (2020a). Towards learning affine-invariant representations via data-efficient cnns. In *The IEEE Winter Conference on Applications of Computer Vision* (pp. 904–913).

[180] Xu, W., Wu, Y., Ma, W., & Wang, G. (2020b). Adaptively denoising proposal collection for weakly supervised object localization. *Neural Processing Letters*, 51(1), 993–1006.

[181] Yadav, M. & Agarwal, S. (2017). Regularization and learning an ensemble of rnns by decorrelating representations. In *The AAAI-17 Workshop on Crowdsourcing, Deep Learning, and Artificial Intelligence Agents WS-17-07*.

[182] Yang, F., Choi, W., & Lin, Y. (2016). Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2129–2137).

[183] Yang, J., Zhang, S., Wang, G., & Li, M. (2015). Scene and place recognition using a hierarchical latent topic model. *Neurocomputing*, 148, 578–586.

[184] Yu, X., Liu, T., Gong, M., & Tao, D. (2018). Learning with biased complementary labels. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 68–83).

[185] Zagoruyko, S. & Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

[186] Zagoruyko, S., Lerer, A., Lin, T.-Y., Pinheiro, P. O., Gross, S., Chintala, S., & Dollár, P. (2016). A multipath network for object detection. *arXiv preprint arXiv:1604.02135*.

[187] Zeiler, M. D. & Fergus, R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

[188] Zhang, S., Jiang, H., & Dai, L. (2016). Hybrid orthogonal projection and estimation (hope): A new framework to learn neural networks. *JMLR*, 17(1), 1286–1318.

[189] Zhang, S., Wen, L., Bian, X., Lei, Z., & Li, S. Z. (2018a). Single-shot refinement neural network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4203–4212).

[190] Zhang, X., Wan, F., Liu, C., Ji, R., & Ye, Q. (2019). Freeanchor: Learning to match anchors for visual object detection. In *Advances in Neural Information Processing Systems* (pp. 147–155).

[191] Zhang, Z. & Brand, M. (2017). Convergent block coordinate descent for training tikhonov regularized deep neural networks. In *NIPS* (pp. 1721–1730).

[192] Zhang, Z., Ma, W., Wu, Y., & Wang, G. (2020). Self-orthogonality module: A network architecture plug-in for learning orthogonal filters. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1050–1059).

[193] Zhang, Z. & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems* (pp. 8778–8788).

[194] Zhang, Z., Wu, Y., & Wang, G. (2018b). Bpgard: Towards global optimality in deep learning via branch and pruning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 3301–3309).

[195] Zhou, X., Wang, D., & Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*.

[196] Zhu, C., He, Y., & Savvides, M. (2019). Feature selective anchor-free module for single-shot object detection. *arXiv preprint arXiv:1903.00621*.

[197] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.

[198] Zhu, X., Zhou, W., & Li, H. (2018). Improving deep neural network sparsity through decorrelation regularization. In *IJCAI* (pp. 3264–3270).