

Learning with Limited Data and Supervision

Amir Rahimi

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

November 2021

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted to obtain a degree in any other institution. This thesis is my own work which has been done in collaboration with other researchers.

Amir Rahimi
25 November 2021

To my beloved parents

Acknowledgments

I would like to thank my collaborators who helped through suggestions and discussions while writing the papers included in this thesis. I would like to thank my collaborators (in Alphabetical order) Thalaiyasingam Ajanthan, Byron Boots, Ching-An Cheng, Stephen Gould, Richard Hartley, and Amirreza Shaban. This thesis would not be possible without their contributions. Special thanks to my advisor Prof. Richard Hartley for providing me the opportunity to study under his supervision. I would also thank Prof. Stephen Gould for helping me become a better researcher in the initial stages of my Ph.D. Finally, I would like to thank my family for sending their unconditional love and support from far away.

Abstract

Deep neural networks have been the main driving force of recent successes in machine learning leading to the deployment of these models in a wide range of industries such as healthcare, autonomous driving, and fintech. Despite the great success, these models are known as data-hungry models requiring many labelled training examples and costly computational resources to solve a pre-determined task. Several obstacles limit the applicability of deep learning models in real-world scenarios. First, annotating large-scale training data in tasks such as object localization or segmentation is cumbersome and demands huge time and labor. Second, in real-world scenarios and applications such as field robotics, the models may be required to learn new classes in an ever-changing environment. However, accessing abundant fully labelled training data for novel classes may be infeasible. Therefore, a model needs to adapt to learn novel classes given only a few examples with simple (weak) annotations. Finally, it is known that most modern deep convolutional networks do not have calibrated confidence scores, meaning that the confidence scores they assign to the outcomes do not match the true frequency of those events. These models are of utmost importance to output calibrated prediction scores that the downstream applications can rely upon, especially in safety-critical applications. This thesis focuses on tackling these limitations in deep learning models with applications in Computer Vision. We investigate the task of finding common objects in small image collections and propose an efficient graphical model inference algorithm that utilizes the structure of the problem to reduce the computational time compared to traditional inference algorithms significantly. We also propose a probabilistic approach to solve the few-shot common object localization problem based on a parametric distribution of each class on a unit sphere. We further extend our model to localize objects of novel classes in unseen images. In the next step, we study pairwise similarity knowledge transfer for weakly supervised object localization to reduce the cost of labor and time in annotating large-scale object detection datasets for novel classes. We learn the similarity functions and the assignment of proposals to different novel classes jointly using alternating optimization and show that the assignment problem becomes an integer linear program for a certain type of loss function. Furthermore, we propose an efficient inference algorithm to overcome the difficulty of computing all pairwise similarities. Finally, to overcome pre-trained models' accuracy degradation in learning expressive probability calibration functions using small calibration data, we introduce and formalize the notion of order-preserving functions. We also present two sub-families of order-preserving functions that benefit from parameter sharing across different classes in classification problems.

Contents

Acknowledgments	vii
Abstract	ix
1 Introduction	1
1.1 Task Definitions and Contributions	2
1.1.1 Finding Common Objects Across a Few Image Collections	2
Task definition.	2
Contributions.	3
1.1.2 Few-shot Weakly Supervised Object Detection	3
Task definition.	3
Contributions.	4
1.1.3 Knowledge-transfer for Weakly Supervised Object Localization .	5
Task definition.	5
Contributions.	5
1.1.4 Post-hoc Confidence Calibration of Deep Neural Networks	6
Task definition.	6
Contributions.	6
1.2 Thesis Outline	7
2 Background and Related Work	9
2.1 Supervised Learning	9
2.2 Multiple-Instance Learning	10
2.2.1 MI-SVM and mi-SVM	11
2.3 Few-shot Image Classification	12
2.3.1 Prototypical Networks	13
2.3.2 Relation Networks	13
2.4 Object Detection	14
2.5 Weakly Supervised Object Localization	16
2.5.1 Knowledge-Transfer in MI-SVM WSOD	17
2.6 Confidence Calibration of Neural Networks	18
3 Efficient Inference for Finding Common Objects Across Small Image Col- lections	21
3.1 Introduction	21
3.2 Problem Setup	23
Energy function.	24

3.2.1	Dataset Setup	25
3.3	Potential Functions	25
	Relation module.	25
	Pairwise potentials.	26
	Unary potentials.	26
3.3.1	Inference	27
	Joining.	28
	Pruning.	28
	Complexity.	30
3.4	Experiments	30
3.4.1	Baseline Methods	31
3.4.2	Few-shot Common Object Recognition	31
	Dataset.	31
	Feature extractor.	31
	Sampling collections.	31
	Evaluation metric.	32
	Setting.	32
	Results.	32
	Effect of temperature T on unary potential function.	32
3.4.3	Few-shot Object Co-Localization	34
	Datasets.	34
	Feature extractor.	35
	Implementation.	35
	Evaluation metric.	35
	Results.	36
	Effect of bag size and larger N, \bar{B}	36
3.4.4	Comparison of Energy Minimization Methods	39
3.5	Summary	39
4	Few-shot Weakly-Supervised Object Detection via Directional Statistics	41
4.1	Introduction	41
4.2	Details of Methodology	44
4.2.1	Few-Shot WSOD and COL Tasks Definition	44
4.2.2	Pre-training and Feature Extraction	45
4.2.3	Statistical Model Assumptions	46
4.2.4	COL	47
	4.2.4.1 Expectation-Maximization Derivation	49
	4.2.4.2 Updating κ in M-Step	51
4.2.5	Finding the Common Object in the Query Set	53
4.2.6	WSOD	54
4.3	Experiments	55
4.3.1	Common Object Localization	55
	4.3.1.1 Effect of Updating κ in M-Step	56
	4.3.1.2 Direct Comparison to Greedy Tree	57

4.3.2	Few-shot WSOD	58
4.3.3	Large-Scale WSOD	58
4.3.4	Ablation Study	59
4.3.5	Qualitative Results	61
4.4	Summary	69
5	Pairwise Similarity Knowledge Transfer for Weakly Supervised Object Localization	71
5.1	Introduction	71
5.2	Problem Description and Background	73
5.2.1	Dataset and Notation.	73
5.2.2	Multiple-Instance Learning (MIL).	74
5.2.3	Optimization.	75
5.2.4	Knowledge Transfer.	75
5.3	Proposed Method	75
5.3.1	Re-localization	76
	Inference.	78
	Complexity.	79
5.3.2	Knowledge Transfer	80
5.3.3	Network Architectures	80
	Proposal and feature extraction.	80
	Scoring functions.	81
5.4	Experiments	81
5.4.1	COCO 2017 Dataset	81
5.4.1.1	Initialization Scheme	81
5.4.1.2	Full Pipeline	82
5.4.2	ILSVRC 2013 Detection Dataset	83
5.4.2.1	Baselines and Results	84
5.5	Summary	88
6	Intra Order-Preserving Functions for Calibration of Multi-Class Neural Networks	89
6.1	Introduction	89
6.2	Problem Setup	91
6.2.1	Importance of Inductive Bias	93
6.3	Intra Order-Preserving Functions	93
6.3.1	Setup: Sorting and Ranking	93
6.3.2	Intra Order-Preserving Functions	94
6.3.3	Order-Invariant and Diagonal Sub-families	95
6.3.4	Practical Considerations	96
6.4	Implementation	97
6.5	Experiments	98
	Datasets.	98
	Baselines.	100

6.5.1	Results	101
6.5.2	Ablation Studies and More Experiments	103
6.5.2.1	Is Classwise-ECE a Proper Scoring Rule Calibration Metric?	105
6.5.2.2	Debiased ECE and a Fix to Classwise-ECE	107
6.6	Summary	107
7	Conclusion	109
7.1	Future Directions	111
	Multiple diverse proposals for common objects.	111
	Universal Cross-Transformers for few-shot classification.	111
	Incorporating semantic knowledge.	111
	Other applications of intra order-preserving functions.	111
A	Appendix A	113
A.1	Modeling with Gaussian Distribution	113
A.2	MI-SVM WSOD Baseline	113
B	Appendix B	115
B.1	Missing Proof for Linearity of Labels in Sigmoid Cross-entropy Loss Function	115
C	Appendix C	117
C.1	Missing Proofs for Intra Order-Preserving Functions	117
C.1.0.1	Deferred Proofs of Lemmas	118
C.1.1	Proof of Theorem 2, Order-invariant Functions	119
C.1.1.1	Properties of Order Invariant Functions	119
C.1.1.2	Main Proof	120
C.1.1.3	Deferred Proof of Lemmas	120
C.1.2	Proof of Theorem 3, Diagonal Functions	122
C.2	Continuity and Differentiability of the Proposed Architecture	123
C.3	Reliability Diagrams	124

List of Figures

1.1	Example of annotations for a supervised object detection problem. Providing fully annotated datasets can be cumbersome in such tasks.	2
1.2	An example of the finding common objects problem: Each image corresponds to a bag in this example. Bounding boxes represent the elements of the bags within each image. In this example, “horse” is the target class and is shown by blue bounding boxes. Orange boxes show all other non-target elements. In this problem, the negative bag is optional but can help reduce the ambiguity of the problem. For instance, the “persons” in the positive bags cannot be the target object since there is a person in the negative bag. The finding common objects task is defined as finding a selection, one element from each positive bag, such that all the selected elements are from the same (target) class. So, in this example, we are given the bags with bounding boxes as their elements. We know whether a bag is positive or negative. However, we do not know their elements’ labels. The task is to select one horse bounding box in each positive bag.	3
1.3	An example of the few-shot weakly supervised object detection problem. The input is a support set comprising a few images containing instances of unseen categories with only image-level labels. The task is to find instances of the objects (from the novel categories in the support set) in the query images. . . .	4
1.4	Knowledge transfer for weakly supervised object localization problem. We have access to a fully annotated source dataset (left), and images of novel classes in the target dataset with image-level labels (middle). Our goal is to provide bounding box annotations to the images in the target dataset (right). .	5
2.1	Prototypical Networks classification in the embedding space. Image from [Snell et al., 2017].	14
2.2	Architecture of the relation networks for 1-shot, 5-way classification problem using a query example. Image from [Sung et al., 2018].	14
2.3	Faster-RCNN and RPN architectures. Images from [Ren et al., 2015].	15
3.1	Examples of finding common objects problems we consider in this chapter. . .	22
3.2	Architecture of the relation module used for unary and pairwise potentials. . .	25
3.3	Each bag and its elements are shown by a horizontal green box and red circles, respectively. A subproblem between p -th and q -th bag is presented as $\mathcal{T}_c(p, q)$.	28

3.4	An example of the greedy inference algorithm on a collection with $N = 4$ positive bags. The first “Joining” operation (left) creates solution proposals of size two and the second one (right) creates solution proposals of size four. The “Pruning” operation (middle) prunes the solution proposals with high energy values.	29
3.5	Average runtime vs. accuracy of different inference algorithms on miniImageNet for $N \in \{8, 16\}$, $\bar{B} \in \{0, 10, 20\}$, and $B = 10$. Each setting is shown with a distinct color and different inference algorithms are shown with different type of markers.	34
3.6	Computational time comparison of forward (computing pairwise similarities) and graphical model inference (in sec.) on COCO dataset. Image from [Shaban et al., 2019].	35
3.7	Qualitative results on COCO dataset. Every two rows show a sampled collection. For every collection, the first row shows the positive bags followed by negative bag on the second row. Note that the first image in the first two collections are identical but the target class (“Cake” vs. “Cat”) is different. In the first problem, class “Person” does not appear in the negative images. This could explain why “Unary Only” method detects people in the first problem. The last row shows a failure case of our algorithm. While “Cup” is the target object, our method finds “Plant” in the second image. This might be due to the fact that pot (which has visual similarities to “Cup”) and “Plant” are labelled as one class in the training dataset. Note that “Dog”, “Cake” and “Cup” are samples from unseen classes. Selected regions are tagged with method names. Ground-truth target bounding box is shown in green with tag “GT”. Image from [Shaban et al., 2019].	37
3.8	Qualitative results on ImageNet dataset. In each sampled collection, the first row and the second row show positive and negative images respectively. Similar to Figure 3.7, the greedy method performs better when there are multiple objects in each positive image. Selected regions are tagged with method names. Ground-truth target bounding boxes are shown in green with tag “GT”. Image from [Shaban et al., 2019].	38
4.1	Few-shot WSOD problem. Similar to the few-shot classification problem, the input training set (support set) only contains image labels (car, cow and person are novel classes in this example). The model learns to detect the target objects in the test (query) image. Few-shot WSOD bridges few-shot classification and object detection by learning to detect the novel objects in the query images while only needs image-level labels for the support images. . . .	42

-
- 4.2 The feature maps are shown as the shape of their tensors. Q , M , and C denote the number of queries, support images, and classes respectively. A pre-trained Faster-RCNN (shown in Figure 4.3) on the base dataset is used to extract P proposals from each input image. The embeddings are grouped based on their corresponding image-level labels and each group is fed into a separate Common-Object Localization (COL) module. **COL** module (shown in detail on the right) receives proposal embeddings of images of a class (M_c is the number of images within class c) and simultaneously estimates the common class direction θ_c and concentration κ_c along with bounding-box level labels \mathbf{w}_c via EM steps. **The Object Detection** module uses the top labels of \mathbf{w}_c to learn an appearance model for each novel class in the support set. This appearance model is then tested on the testing proposals to detect novel objects in the query set. 43
- 4.3 Feature Extraction. We use a pre-trained Faster-RCNN on the base dataset to extract P proposals from each input image. A ℓ_2 normalization layer is employed to project all the features onto the unit hypersphere. 45
- 4.4 Two-dimensional T-SNE projection of proposal features extracted from a pre-trained Faster-RCNN. The proposals are selected using **IoU** threshold of 0.6 with ground-truth boxes. The instances for each class (approximately) form a single cluster. 46
- 4.5 Example of COL across three images. Data points on the unit sphere represent feature proposals extracted from all input images. Features extracted from each image are colored the same (shown in white, gray, black colors). Background score function $u_{\omega}^-(\mathbf{x})$ is also shown on the unit sphere where blue and red indicate the highest and lowest background scores, respectively. The COL unit's goal is to find a common object representation θ (shown by green arrow) which is close to at least a white, gray, and black data point. Note that the area marked with dashed circle is also close to proposals from all three images but direction θ is favored as it has a lower background score. 48
- 4.6 Plot of different estimates of $\hat{\kappa}$ as a function of \bar{r} , for dimension $d = 100$. At this resolution, the exact estimate is indistinguishable from the estimate Equation (4.21). The graph also shows approximations of different orders, such as Equation (4.23) and Equation (4.24), which are accurate for small-to-medium values of \bar{r} , but not for larger values. However, the exact value of $\bar{\kappa}$ is extremely sensitive to small variations in the value of \bar{r} , and it diverges to infinity as \bar{r} approaches 1. For this reason it may not be good practice (as is verified by our experiments) to use the exact estimate of $\bar{\kappa}$ in clustering. . . . 52
- 4.7 mAP(%) vs. number of EM iterations in common object localization task with $K = 5$ on COCO60 and VOC07 datasets. The performance reaches a plateau at step 4. 58

4.8	<i>Bounding box adjustments at each iteration for the common object localization experiment on COCO60 with $K = 5$. Only the top prediction in the query image is shown (in pink color) for each iteration. Ground-truth bounding boxes of the target classes are shown in green. EM refinements improve the target object localization in the query image.</i>	60
4.9	<i>Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object on the 4 different query images on the right side. The algorithm fails to detect person in the first query image but successfully detects other target objects.</i>	62
4.10	<i>Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.</i>	63
4.11	<i>Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.</i>	64
4.12	<i>Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.</i>	65
4.13	<i>Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.</i>	66
4.14	<i>Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.</i>	67
4.15	<i>5-shot common object localization ($N = 1$) on MS COCO. Each row shows one common object localization problem. Ground-truth annotations (shown in green) are just for visualization and are not used in the algorithm. Top query bounding box prediction for each problem is shown in pink.</i>	68
5.1	<i>ICM iteration (left) and initialization (right) graphical models. In both graphs, each node represents a bag (with B proposals) within a dataset with $\mathcal{T}_c = 9$ bags. Left: ICM updates the unary label of the selected node (shown in green). Edges show all the pairwise labels that gets updated in the process. Since the unary labeling of other nodes are fixed each blue edge represents B elements in vector $\hat{\mathbf{r}}_c$. Right: For initialization we divide the dataset into smaller mini-problems (with size $K = 3$ in this example) and solve each of them individually. Each edge represents B^2 pairwise scores that need to be computed.</i>	79

5.2	Success cases on ILSVRC 2013 dataset. Unary method that relies on the objectness function tends to select objects from source classes that have been seen during training. Note that “banana”, “dog”, and “chair” are samples from source classes. Bounding boxes are tagged with method names. Each method is shown with distinct color: “Ours”, “WU”, “Unary”, “WU(Unary)”, “GT” represent our method, warm-up, unary-only, warm-up with unary, and groundtruth respectively.	85
5.3	Extended results of Figure 5.2	86
5.4	Failure cases on ILSVRC 2013 dataset (see Figure 5.2 for details).	87
5.5	Success and failure cases on COCO dataset. First two rows show the success cases of our method while the last row shows the failure cases. For a success case example, the task for the middle image in the second row is to find the cat. However, other methods selected the dog in this case. For a failure case, in the bottom right image, our method selected the object inside the mirror (see Figure 5.2 for details).	87
6.1	Comparing instances of intra order-preserving and order-invariant family defined on the 2-dimensional unit simplex. Points $C_1 = [1, 0, 0]^\top$, $C_2 = [0, 1, 0]^\top$, $C_3 = [0, 0, 1]^\top$ are the simplex corners. Arrows depict how an input is mapped by each function. Unconstrained function freely maps the input probabilities, intra order-preserving function enforces the outputs to stay within the same colored region as the inputs, and order-invariant function further enforces the vector fields to be the same among all the 6 colored regions as reflected in the symmetry in the visualization.	91
6.2	Relationship between different function families. Theorem 1 specifies the intra order-preserving functions \mathbb{A} . Theorem 2 specifies the intra order-preserving and order-invariant functions $\mathbb{A} \cap \mathbb{B}$. Theorem 3 specifies the diagonal intra order-preserving functions \mathbb{D} . By Corollary 1, these functions are also order-invariant and inter order-preserving i.e. $\mathbb{D} \subseteq \mathbb{A} \cap \mathbb{B} \cap \mathbb{C}$	96
6.3	Flow graph of the intra order-preserving function. The vector $\mathbf{x} \in \mathbb{R}^n$ is the input to the graph. Function \mathbf{m} is estimated using a generic multi-layer neural network with non-linear activation for the hidden layers. The input to the network is sorted for learning order-preserving functions. We employ softplus activation function s^+ to impose strict positivity constraints.	98
6.4	Accuracy, ECE, and NLL plots in MS and DIR for ResNet 152 on ImageNet with different regularization weights. In the plots, x-axis shows the log scale regularization and y-axis shows the accuracy, ECE, and NLL of different methods, respectively. The value of the bias regularizer is found by cross validation and kept constant for visualization purpose. Changing the bias regularizer has little effect on the final shape of the plots.	101

6.5	Performance evaluations of ResNet 152 (Top Row) and PNASNet5 large (Bottom Row) on ImageNet dataset. (Left) Reliability diagrams. As suggested by [Maddox et al., 2019] we show the difference between the estimated confidence and accuracy over $M = 15$ bins. The dashed grey lines represent the perfectly calibrated network at $y = 0$. Points above (below) the grey line show overconfident (underconfident) predictions in a bin. (Middle) Weighted reliability diagrams where bin values are weighted by data frequency distribution. Since the uncalibrated network has different distances to the perfect calibration in different bins, scaling by a single temperature will lead to a mix of underconfident and overconfident regions. Our order-preserving functions, on the other hand, have more flexibility to reduce the calibration error. (Right) Transformation learned by DIAG function compared to temperature scaling and uncalibrated model (identity map).	102
6.6	Accuracy, NLL, and ECE vs. calibration set size for CIFAR, CARS, BIRDS datasets. For each experiment, we use from 10% to 100% of the calibration set to train pos-hoc calibration functions and plot their accuracy, NLL, and ECE. Compared to DIR and MS, performance of the intra order-preserving methods (TS, DIAG, OI, and OP) degrades less with reducing the calibration set size. .	104
C.1	Reliability diagrams and learned diagonal functions. See Figure 6.5 for the explanation of each diagram and axis.	126
C.2	Reliability diagrams and learned diagonal functions. See Figure 6.5 for the explanation of each diagram and axis.	127

List of Tables

3.1	Statistics of the standard split of the miniImageNet dataset [Ravi and Larochelle, 2017] used for the few-shot common object recognition task.	31
3.2	Results on the miniImageNet dataset using different positive bags N , total number of negative elements \bar{B} , and bag sizes $B = 5$ and $B = 10$. The baseline method uses cosine similarity on the bag elements as relation module*.	33
3.3	Comparison of our method with other MIL methods (top), and graphical model inference methods (middle). The effect of unary and pairwise potentials are shown in the bottom part (bottom). The common object is found across 8 positive and 8 negative images in these examples*.	33
3.4	Comparison of different unary potential functions on miniImageNet dataset with $N = 8$, $B = 5$ and $\bar{B} = 10$	34
3.5	Different statistics of the datasets used for the few-shot object co-localization task. . .	34
3.6	Run time and accuracy by varying the bag size and number of positive bags on COCO dataset.	36
3.7	Average energy values for different graphical model inference methods on the mini-ImageNet dataset*.	39
3.8	Average energy values for the few-shot object co-localization task*.	39
4.1	CorLoc (top) and mAP (bottom) performance of different few-shot common object localization methods on VOC07 test set. All of the models are trained on COCO60 and evaluated on a test query with $K = 5$ images in the support set. The best and second best performing methods are shown in bold and gray backgrounds respectively. *MI-SVM receives K extra negative images.	55
4.2	CorLoc(%) and mAP(%) results of different methods for the task of common object localization on novel object classes on the COCO60 dataset with support set size $K = 5$ and $K = 10$. *MI-SVM receives K extra negative images.	56
4.3	CorLoc(%) and mAP(%) results with κ estimations for the task of COL on novel object classes on the COCO60 dataset with support set size $K = 5$ and $K = 10$	57
4.4	Class-agnostic CorLoc(%) with 95% confidence interval of the method in Chapter 3 compared to our method. All methods use $K = 8$ positive images for finding the common object.	57
4.5	mAP(%) of different few-shot WSOD methods on COCO60 and PASCAL VOC datasets.	58
4.6	Large-Scale WSOD on ImageNet Detection.	59
4.7	Ablation study on COCO60 dataset. #1-6 show the importance of initialization, iterative EM updates, and learning the background model. #7-9 compare different statistical models in the EM algorithm.	61

5.1	<i>Performance and time comparison of different initialization algorithms. Our method exhibits the highest initialization performance for $K = 64$, however, we get similar performance for $4 \leq K \leq 64$ after applying ICM.</i>	82
5.2	<i>Performance of different methods on ILSVRC 2013. Proposal generators and their backbone models are shown in the second and third column. Total time is shown in "Training+Inference" format. CorLoc is reported on the target set. The last column shows the performance of an object detector trained on the target set and evaluated on the target test set. *The first 3 methods use RCNN detector with AlexNet backbone while other methods utilize Faster-RCNN detector with Inception-Resnet backbone.</i>	83
6.1	<i>Statistics of the Evaluation Datasets.</i>	99
6.2	<i>Hyperparameters learned by cross validation. For DIAG, OI, OP, and UNCONSTRAINED we show the network architectures learned by cross validation. The number of units in each layer are represented by a sequence of numbers, e.g. (10,20,30,40) represents a network with 10 input units, 20 and 30 units in the first and second hidden layers, respectively, and 40 output units. We perform multi-fold cross-validation and select the architecture with lowest NLL on validation set.</i>	100
6.3	<i>ECE (with $M = 15$ bins) on various image classification datasets and models with different calibration methods. The subscript numbers represent the rank of the corresponding method on the given model/dataset. The accuracy of the uncalibrated model is shown in parentheses. The number in parentheses in DIR, MS, and UNCONSTRAINED methods show the change in accuracy for each method.</i>	101
6.4	<i>Average relative error. Each entry shows the relative error compared to the uncalibrated model averaged over all the datasets. The subscripts represent the rank of the corresponding method on the given metric. See the Appendix for per dataset performance comparisons.</i>	103
6.5	<i>Scores and rankings of different methods for Brier.</i>	103
6.6	<i>NLL.</i>	105
6.7	<i>Classwise ECE.</i>	106
6.8	<i>Temperature scaling effect on Classwise-ECE. A large temperature value improves the Classwise-ECE in most of the cases. The subscript numbers represent the rank compared to the values in Table 6.7. We remark that the purpose of this experiment is not to improve the performance but rather highlight the need for studying Classwise-ECE metric in the future works.</i>	107
6.9	<i>Debiased ECE [Kumar et al., 2019].</i>	108
6.10	<i>Marginal Calibration Error [Kumar et al., 2019].</i>	108

Introduction

A key aspect of human intelligence is learning new concepts from a limited number of examples with minimal supervision. As an example, consider the problem of object detection, where the goal is localizing objects within an image. Suppose we show a few pictures of a previously unknown object, e.g., “zebra,” to a child and tell that all of them contain instances of the previously unknown object category. In that case, the child can learn the new object category and locate instances from that object category in new images. Note that we only provide weak supervision to solve the new task, i.e., showing images with the unknown object rather than providing their bounding boxes to solve the problem. The child can solve the new task since they have been provided with many examples from similar objects, e.g., horses, during their life and can *transfer* that knowledge to the new task.

With the advent of deep learning methods, mixed with large amounts of data and fast computational resources in recent years, there have been great successes in solving tasks such as language translation, image classification, and many more. Deep learning methods supersede the burden of feature engineering by automatically learning task-relevant features as part of the task objective. However, most of these methods require a large amount of labeled data and computational time to get near-human performance limiting their applicability in real-world scenarios. The reason is that hand-labeling data often requires a significant amount of time and investment. Additionally, the trained models may require solving a similar learning task but for novel categories in a limited time when deployed in applications. However, a large amount of training data and expensive computational resources are not accessible to the clients. Moreover, the clients cannot wait for days of training to have their tasks solved.

Labeling datasets requires significant effort for tasks such as object detection or segmentation because they usually require object/pixel-level annotations as shown in Figure 1.1. Weakly supervised methods have been leveraged in such scenarios to minimize labor costs. These methods utilize weaker forms of annotations such as image-level tags to solve the task at hand. Nevertheless, they usually are not transferable to unknown categories and require many training examples to learn from weak labels. Meta-learning and few-shot learning techniques, on the other hand, reduce requiring large training instances in low data regimes. Nonetheless, these methods require full supervision for new classes.



Figure 1.1: *Example of annotations for a supervised object detection problem. Providing fully annotated datasets can be cumbersome in such tasks.*

In this research, we study learning and inference algorithms that shift the burden of tedious hand-labeled data (as deep networks supersede feature engineering) and facilitate learning new tasks with limited data or supervision. Our goal is to transfer knowledge from a fully supervised dataset by training a model that can learn to solve a new unknown learning task by having a (possibly small) number of examples that themselves have weak labels. Our applications of interest for these tasks are objection localization and object detection. Furthermore, we explore the applicability of deep learning models in safety-critical applications where the downstream decision-making system can rely on the predicted probabilities output from these models.

1.1 Task Definitions and Contributions

1.1.1 Finding Common Objects Across a Few Image Collections

Task definition. In this problem, the input is a small number of bags where each bag contains multiple elements. An element either belongs to one of the pre-defined foreground classes or a background class. Given a target class, a bag is considered positive if at least one of its elements belongs to the target class; otherwise, it is considered as a negative bag. The algorithm only knows whether a bag is positive or negative concerning an unknown target class without knowing the elements' labels. We say the bag is weakly labeled in this case. The goal of the algorithm is to select one element from each positive bag such that all the selected elements are from the same target class (see Figure 1.2). We assume an annotated source dataset with element-wise annotations comprising multiple known object categories is available during training. It is assumed that the target class is from a novel category (not seen during training) at test time. Although this task has many applications in Computer Vision such as action localization [Yang et al., 2020], object co-segmentation [Vicente et al., 2011], and etc., we consider the problems of few-shot common object recognition where each bag element is an image of a single category and few-shot object co-localization where a bag is an image, and its elements correspond to object proposals generated from that image.



Figure 1.2: An example of the finding common objects problem: Each image corresponds to a bag in this example. Bounding boxes represent the elements of the bags within each image. In this example, “horse” is the target class and is shown by blue bounding boxes. Orange boxes show all other non-target elements. In this problem, the negative bag is optional but can help reduce the ambiguity of the problem. For instance, the “persons” in the positive bags cannot be the target object since there is a person in the negative bag. The finding common objects task is defined as finding a selection, one element from each positive bag, such that all the selected elements are from the same (target) class. So, in this example, we are given the bags with bounding boxes as their elements. We know whether a bag is positive or negative. However, we do not know their elements’ labels. The task is to select one horse bounding box in each positive bag.

Contributions. We formalize the problems of finding instances of common object categories across small collections and provide a benchmark for evaluating the few-shot common object recognition and few shot object co-localization tasks. The problem is formulated as an energy minimization problem with learned unary and pairwise potential functions. We propose a specialized algorithm for the structured inference problem that achieves comparable performance to state-of-the-art inference methods while requiring less computational time. The project was a collaboration between the author and Professor Byron Boot’s group at the Georgia Institute of Technology (GATech). The author’s contributions include (i) formalization of the problem, (ii) a novel greedy merge-and-prune inference heuristic to find the minimum cost labeling, and (iii) developing an attention mechanism for learning the unary potentials. Apart from these contributions, the author contributed to the development of all parts of this project. The greedy algorithm uses the fact that the common object in a set of bags is also a common object in any subset of those bags. We eliminate the requirement of computing pairwise similarities between all of the elements in all bags using this simple heuristic. While accurate, it reduced the computational time of the inference by 85% of the other known graph inference methods.

1.1.2 Few-shot Weakly Supervised Object Detection

Task definition. In this task, the input is a support set containing $N \times K$ images with image-level labels where N is the number of previously unseen object classes, and K is the number of images per class. Each image may contain multiple objects,

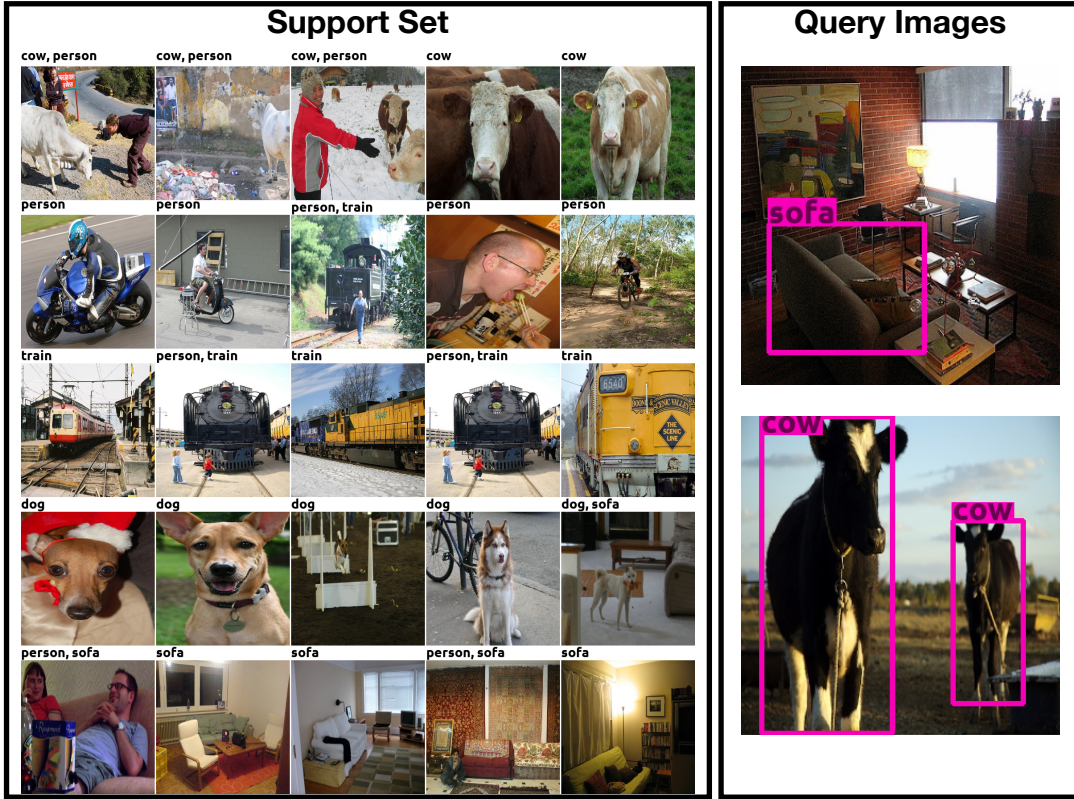


Figure 1.3: An example of the few-shot weakly supervised object detection problem. The input is a support set comprising a few images containing instances of unseen categories with only image-level labels. The task is to find instances of the objects (from the novel categories in the support set) in the query images.

and the number of classes and samples from each class is typically less than 20. The task is to find all the instances from any of the novel classes in a query image. Similar to the finding common objects task, we assume having access to a source dataset with bounding box annotations. Figure 1.3 illustrates an example of the few-shot weakly supervised object detection task.

Contributions. For the few-shot weakly supervised object detection task, our contributions are as follows: First, based on the recent observations in few-shot classification and few-shot object detection [Wang et al., 2020; Tian et al., 2020], we observe that learning a good embedding works surprisingly well for this task. We eliminate the sophisticated episodic training of relation modules and the greedy inference method introduced in our previous contribution without reducing the accuracy. We propose a probabilistic few-shot object co-localization approach that operates directly on the features extracted from a pre-trained (on the source dataset) object detector without introducing any new parameter. We assume instances of each class are distributed around their center and utilize von Mises Fisher (vMF) for our probabilistic distribution. We remark that our probabilistic model can better capture semantic in-



Figure 1.4: *Knowledge transfer for weakly supervised object localization problem. We have access to a fully annotated source dataset (left), and images of novel classes in the target dataset with image-level labels (middle). Our goal is to provide bounding box annotations to the images in the target dataset (right).*

formation than the widely used Gaussian distribution. We employ the expectation-maximization algorithm to find the parameters of the vMF distribution for each class. Second, we propose a detection module based on the cosine distance to extend the object co-localization problem to the few-shot weakly supervised object detection (FSWSOD). We empirically show that our method, despite the simplicity, outperforms other strong baselines for both object co-localization and FSWSOD tasks. The author is thankful to Amirreza Shaban, who originally came up with the idea of using expectation-maximization for learning the class parameters and proposal assignment in weakly supervised settings for Gaussian distributions, which further extended to vMF distribution by the author of this thesis.

1.1.3 Knowledge-transfer for Weakly Supervised Object Localization

Task definition. This task is similar to the few-shot object co-localization task. The difference is that in knowledge-transfer for weakly supervised object localization (WSOL), the task is to localize objects of novel categories in a large-scale weakly supervised target dataset with the help of a fully annotated source dataset of known classes. An illustration for this problem is presented in Figure 1.4.

Contributions. In the few-shot object co-localization task, we learn pairwise and unary scoring functions from the source dataset and apply them at test time on a small number of bags. Now, what if we have a large-scale target dataset instead of few weakly labeled bags? In contrast to the few-shot case, we can now adapt our pairwise and unary scoring functions using the target dataset to improve our localization performance further. It is much harder to perform adaptation in the few-shot setting because the model will overfit to the input data since we have a small number of examples. Multiple-Instance Learning (MIL) methods are typically utilized for model adaptation on the target dataset in the large-scale setting. These methods

perform alternating optimization to gradually learn a classwise objectness (unary) function and optimal proposal selection in re-training and re-localization steps. A class-agnostic objectness score learned from the source dataset guides the optimization. We argue that only using objectness is a weak form of knowledge-transfer and augment the traditional MIL loss function by adding pairwise scoring terms. Similar to MIL methods, we learn the scoring and selection jointly using alternating optimization. We show that by carefully choosing the loss functions, our re-localization becomes an integer linear program. A graphical model can represent this problem. However, most state-of-the-art graph inference methods are not applicable because of the quadratic increase in pairwise labels. We found that coordinate descent optimization methods like Iterated Conditional Modes (ICM) [Besag, 1986] can solve the problem efficiently. Nevertheless, methods like ICM are prone to local minima and have poor performance if they are not initialized properly. We provide high-quality initialization for the ICM algorithm by dividing the target dataset into multiple smaller subproblems and optimize the re-localization loss for each subproblem separately. Our approach makes optimization in large-scale settings possible. With extensive experiments, we show the effectiveness of our proposed method. The author collaborated with Amirreza Shaban, who helped the author with experiments and realized that the sigmoid cross-entropy is linear with respect to the labelling.

1.1.4 Post-hoc Confidence Calibration of Deep Neural Networks

Task definition. Deep neural networks are known to have over-confident predictions. In the context of image classification, the output confidence scores they assign to different classes do not match the empirical frequency of predicting those classes correctly. The mismatch between the predicted confidence score and the empirical frequency restricts their applicability for downstream applications where the trained models might be adapted based on their confidence scores. In post-hoc confidence calibration of deep networks, the task is to transform the output confidence scores of a trained model using a hold-out calibration data of small size such that the transformed confidence scores better match the empirical frequency of predicting the events of interest correctly. Similar to the previously defined tasks, the post-hoc confidence calibration methods objective is to learn generalizable transformations of confidence scores (or logits) given a limited number of calibration examples.

Contributions. In post-hoc calibration, the calibration dataset usually has a small size. So, learning an overly general calibration function can overfit to the calibration data and reduce the accuracy of the base model. This fact motivates us to define intra order-preserving functions that act as a regularization in the space of calibration functions. An intra order-preserving function is a vector-valued function where the output has the same ordering as the input. We provide necessary and sufficient conditions for defining intra order-preserving functions and show how to model them using basic blocks used in any deep learning framework. We also define two other subfamilies, namely order invariant and diagonal intra order-preserving

functions, that benefit from the shared characteristics between different categories. Throughout comprehensive experiments, we show the advantage of the proposed order-preserving regularisations. While the author was the lead contributor to this project, he is thankful to Ching-an Cheng and Amirreza Shaban for their help in providing the proofs presented in Appendix C.

1.2 Thesis Outline

The remainder of this thesis is comprised of 6 chapters. In Chapter 2 we briefly introduce the introductory background materials and definitions which help understand the rest of the thesis. We also concisely review the prior arts relevant to the remainder of this manuscript. We formalize the problem of finding common objects across a few image collections in Chapter 3 and describe the mechanism for learning the potential functions. Subsequently, we introduce our efficient greedy inference algorithm for the problem. This work was presented in [Shaban et al., 2019]. In Chapter 4, we first describe our probabilistic common object localization method. Then, we extend it to localize novel objects in a query image for the task of few-shot weakly supervised object detection. This work will be presented in [Shaban et al., 2022]. We study weakly supervised object localization in the context of knowledge transfer in Chapter 5. We unify the re-localization and re-training losses for this task and show that the re-localization problem becomes equivalent to a graph labelling problem. Furthermore, we present an efficient inference algorithm suitable for large-scale problems. This work was presented in [Rahimi et al., 2020a]. In Chapter 6, we investigate deep neural network confidence calibration and introduce the family of intra order-preserving functions. In addition, we present two other sub-families related to diagonal and order-invariant functions for learning better calibration functions in low-data regimes. This work was presented in [Rahimi et al., 2020b]. Lastly, Chapter 7 concludes and explores future directions to extend the works presented in this thesis.

Background and Related Work

In this chapter, we introduce basic concepts and building blocks that have been used throughout this thesis. We start with defining the problem of supervised learning. Particularly, we introduce the classification problem and the typical loss function used in this problem. Next, we define the multiple-instance learning problem that uses a weaker form of supervision compared to the supervised classification problem. We also present the two most widely used multiple-instance learning algorithms based on support vector machines. We study the problem of few-shot image classification in which we only have access to a few labeled examples for each class. Then, the Faster-RCNN architecture for object detection is presented. We also review recent methods in weakly supervised object localization. Finally, we review the confidence calibration of the neural networks problem.

2.1 Supervised Learning

The supervised learning problem is typically formulated as searching for a parametric function $f_\theta : \mathcal{D}_X \rightarrow \mathcal{D}_Y$, with parameters $\theta \in \Theta$ such that the expectation of a given loss function $\mathcal{L}(\mathbf{x}, y, f_\theta) : \mathcal{D}_X \times \mathcal{D}_Y \times \Theta \rightarrow \mathbb{R}$ for input samples $\mathbf{x} \in \mathcal{D}_X$, and target values $y \in \mathcal{D}_Y$ is minimized:

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathbb{E}_{(\mathbf{x}, y) \sim P} \mathcal{L}(\mathbf{x}, y, f_\theta) = \int \mathcal{L}(\mathbf{x}, y, f_\theta) dP(\mathbf{x}, y) . \quad (2.1)$$

In this formulation, we assume $P(\mathbf{x}, y)$ is a joint probability distribution over our input random variable X and target random variable Y , and Θ is the space of all parameters we seek to minimize our objective. Random variables X and Y take values in some domains \mathcal{D}_X and \mathcal{D}_Y respectively. In classification problems, the label y takes values from a set of c predefined classes $y \in \llbracket c \rrbracket$, where we denote this set by $\llbracket c \rrbracket := \{1, 2, \dots, c\}$.

The expectation in Equation (2.1) cannot be computed because the distribution $P(\mathbf{x}, y)$ is unknown. But, we assume having access to a training set $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with instances drawn i.i.d from $P(\mathbf{x}, y)$. A general approach to finding the function

f is to minimize the empirical risk instead:

$$\hat{\theta} = \operatorname{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(\mathbf{x}_i, y_i, f_{\theta}) . \quad (2.2)$$

The goal is to learn the associations between \mathbf{x}_i and y_i from the training data and (possibly) measure the performance of the learned function on a dataset of unseen samples known as the test dataset $\mathcal{D}_{\text{test}}$.

In this thesis, the function f_{θ} is parametrized by a neural network with m parameters $\theta \in \Theta$ and is denoted by $f(\mathbf{x}; \theta)$, with $\Theta \subseteq \mathbb{R}^m$ representing the space of all possible parameters. In classification problems, it is assumed that f_{θ} outputs a categorical distribution over the possible target classes. In this case, the output domain \mathcal{D}_Y is denoted by a $c - 1$ dimensional unit simplex denoted by Δ^c . A c -dimensional vector $\mathbf{q} = [q_1, \dots, q_c]^{\top} \in \Delta^c$ on the simplex $\Delta^c \subset \mathbb{R}^c$ satisfies the following two conditions: (i) $q_i \geq 0, \forall i \in \llbracket c \rrbracket$, and (ii) $\sum_i q_i = 1$. A target class $y \in \llbracket c \rrbracket$ corresponds to a one-hot vector residing on a vertex of the unit simplex. In this case, using

$$\mathcal{L}(\mathbf{x}_i, y_i, f_{\theta}) = -\log(f_{y_i}(\mathbf{x}_i; \theta)) , \quad (2.3)$$

with f_{y_i} denoting the y_i -th element of f , corresponds to the well-known *negative log-likelihood* (NLL) or the cross-entropy loss function. Thus, the minimization in Equation (2.2) is equivalent to maximum likelihood estimation of the parameters θ :

$$\theta_{\text{MLE}}^* = \operatorname{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n -\log(f_{y_i}(\mathbf{x}_i; \theta)) . \quad (2.4)$$

In deep learning-based classification methods the categorical distribution over the possible target classes, also known as confidence scores, are obtained by applying a softmax function at the final layer of the network. The softmax function $\sigma_{\text{SM}} : \mathbb{R}^c \rightarrow \Delta^c$ on a vector $\mathbf{z} = [z_1, \dots, z_c]^{\top} \in \mathbb{R}^c$ is defined as

$$q_i = \sigma_{\text{SM}}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^c \exp(z_j)} , \quad (2.5)$$

where $\sigma_{\text{SM}}(\mathbf{z})_i$ is its i -th element. The network outputs before the softmax layer (e.g., \mathbf{z} in Equation (2.5)) are called the network's predicted *logits*.

2.2 Multiple-Instance Learning

Multiple-instance learning (MIL) assumes a weaker form of supervision than the standard supervised learning by assigning a target value to a set of elements, known as a bag, instead of having a target value for each element. We assume the set of indices $\llbracket n \rrbracket$ is partitioned into N subsets of indices sets $\mathcal{I} = \{I_1, \dots, I_N\}$ such that $\bigcup_{I \in \mathcal{I}} I = \llbracket n \rrbracket, I \subset \llbracket n \rrbracket$. For an index set $I \in \mathcal{I}$, a bag \mathcal{B}_I is composed of multiple elements $\mathcal{B}_I = \{\mathbf{e}_i\}_{i \in I}$. In the simplest case, each element $\mathbf{e}_i \in \mathbb{R}^d$ is assumed to have

a hidden binary label $y_i = y(\mathbf{e}_i) \in \{0, 1\}$, and a bag's binary label $\mathcal{Y}_I = \mathcal{Y}(\mathcal{B}_I) \in \{0, 1\}$ is inherited from the elements in the bag. In the positive case with $\mathcal{Y}_I = 1$, at least one of the elements in the bag has a positive label, i.e., $\exists i \in I$ s.t. $y_i = 1$. When the label of a bag is negative, i.e., $\mathcal{Y}_I = 0$, we conclude that all the elements in the bag have negative labels, i.e., $\forall i \in I, y_i = 0$. Shortly, the relation between the bag and element labels can be represented by $\mathcal{Y}_I = \max_{i \in I} y_i$.

In the MIL framework, the training dataset is a set of bags with their corresponding label $\mathcal{D}_{\text{train}}^{\text{MIL}} = \{(\mathcal{B}_I, \mathcal{Y}_I)\}_{I=1}^N$, and the task is to either predict the label of a test bag or find the most positive element in test or training bags. Note that the element-wise labels are unknown to the learning methods in the MIL framework and are used only for evaluation purposes.

2.2.1 MI-SVM and mi-SVM

The most widely used MIL methods in the computer vision community are mi-SVM and MI-SVM [Andrews et al., 2003]¹. In these methods, a Support Vector Machine (SVM) is used to classify the elements in an iterative process. The mi-SVM formulation is a natural extension of soft-margin SVM as a mixed integer programming problem and can be written as

$$\begin{aligned} \text{mi-SVM} \quad & \min_{\{y_i\}} \min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & \forall i : (2y_i - 1)(\langle \mathbf{w}, \mathbf{e}_i \rangle + b) \geq 1 - \xi_i, \xi_i \geq 0, y_i \in \{0, 1\}, \\ & \sum_{i \in I} y_i \geq 1 \quad \forall I \text{ s.t. } \mathcal{Y}_I = 1, \\ & y_i = 0 \quad \forall I \text{ s.t. } \mathcal{Y}_I = 0, \end{aligned} \quad (2.6)$$

where the parameter $C \geq 0$ determines the importance of mis-classified examples (inversely proportional to the strength of regularization.) Note that the optimization is performed jointly on the soft-margin parameters and latent element labels y_i in contrast with the standard SVM, where the integer labels are input to the optimization problem.

In the MI-SVM formulation, the concept of soft-margin on elements is generalized to a soft-margin defined on bags. In this case, the optimization is written as

$$\begin{aligned} \text{MI-SVM} \quad & \min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_I \xi_I \\ \text{s.t.} \quad & \forall I : (2\mathcal{Y}_I - 1) \max_{i \in I} (\langle \mathbf{w}, \mathbf{e}_i \rangle + b) \geq 1 - \xi_I, \xi_I \geq 0. \end{aligned} \quad (2.7)$$

Notice that only the most positive element per positive bag and the least negative one in each negative bag is considered in the minimization in Equation (2.7), while

¹Both “mi” and “MI” are abbreviations for multiple-instance. The lower case “mi” reflects that the SVM loss is defined on instances (elements) of bags, and the upper case “MI” reflects the fact that the loss is defined on bags.

Algorithm 1: mi-SVM Algorithm [Andrews et al., 2003]

Input: $\mathcal{D}_{\text{train}}^{\text{MIL}} = \{(\mathcal{B}_I, \mathcal{Y}_I)\}_{I=1}^N$
Output: (\mathbf{w}, b)
Initialize $y_i^0 = \mathcal{Y}_I$ for $i \in I, t = 0$
do
 Solve inner minimization for Equation (2.6) given the current labels to get SVM solutions \mathbf{w}, b
 Compute outputs $f_i = \langle \mathbf{w}, \mathbf{e}_i \rangle + b$ for all \mathbf{e}_i in positive bags
 $t = t + 1$
 Set $y_i^t = \frac{1}{2}(\text{sgn}(f_i) + 1)$ for every $i \in I, \mathcal{Y}_I = 1$
 for every positive bag B_I do
 if $\sum_{i \in I} y_i^t == 0$ then
 Compute $i^* = \text{argmax}_{i \in I} f_i$
 Set $y_{i^*}^t = 1$
while $y_i^t \neq y_i^{t-1}$ for all $i \in I, \mathcal{Y}_I = 1$
return (\mathbf{w}, b)

Algorithm 2: MI-SVM Algorithm [Andrews et al., 2003]

Input: $\mathcal{D}_{\text{train}}^{\text{MIL}} = \{(\mathcal{B}_I, \mathcal{Y}_I)\}_{I=1}^N$
Output: (\mathbf{w}, b)
Initialize $\mathbf{e}_I^0 = \sum_{i \in I} \mathbf{e}_i / |I|$ for every positive bag $B_I, t = 0$
do
 Find SVM solution (\mathbf{w}, b) for Equation (2.7) given the current positive examples $\{\mathbf{e}_I^t : \mathcal{Y}_I = 1\}$
 Compute outputs $f_i = \langle \mathbf{w}, \mathbf{e}_i \rangle + b$ for all \mathbf{e}_i in positive bags
 $t = t + 1$
 Set $\mathbf{e}_I^t = \mathbf{e}_{\text{argmax}_{i \in I} f_i}$ for every $I, \mathcal{Y}_I = 1$
while $\mathbf{e}_I^t \neq \mathbf{e}_I^{t-1}$ for all $I, \mathcal{Y}_I = 1$
return (\mathbf{w}, b)

all the elements contribute to the optimization in Equation (2.6).

Similar to the mi-SVM formulation, the MI-SVM optimization is also a mixed integer programming problem [Andrews et al., 2003]. These problems cannot be directly optimized. However, some heuristics using the fact that both Equations (2.6) and (2.7) can be solved given the labels y_i have been proposed. The pseudo-codes for both mi-SVM and MI-SVM are presented at Algorithm 1 and Algorithm 2 respectively.

2.3 Few-shot Image Classification

In few-shot image classification, we are given an N -way, K -shot support-set of examples along with their labels $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N \times K}$; each example $\mathbf{x} \in \mathbb{R}^d$ belongs to

one of the K classes, i.e., $y_i \in \llbracket K \rrbracket$, and each class has a small number, N , of labeled examples. The number of images per class is usually less than 20. Given the support set, the task is to predict the class $y \in \llbracket K \rrbracket$ of a query example $\mathbf{x} \in \mathbb{R}^d$. We assume the support set examples are sampled from a dataset $\mathcal{D}_{\text{test}}$ at test time. We also assume having access to a large dataset of labeled examples $\mathcal{D}_{\text{train}}$. The classes in the training dataset $\mathcal{C}_{\text{train}}$ have no class in common with the classes in the test dataset $\mathcal{C}_{\text{test}}$, i.e., $\mathcal{C}_{\text{train}} \cap \mathcal{C}_{\text{test}} = \emptyset$.

Few-shot learning has gained a lot of attention in image classification [Doersch et al., 2020; Rodríguez et al., 2020; Liu et al., 2019; Finn et al., 2017; Vinyals et al., 2016]. These methods are broadly categorized into optimization-based meta-learning [Ravi and Larochelle, 2017; Finn et al., 2017], metric-based meta-learning [Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018], feature reuse-based [Tian et al., 2020; Yang et al., 2021], and methods with stand-alone architectures [Santoro et al., 2016; Mishra et al., 2018]. In the following, we introduce two well-known metric-based methods related to the methods developed in this thesis. These methods use an episodic training scheme; the model is optimized over the distribution of tasks, each of which is called an episode, to mimic test-time task-specific learning of image classifiers.

2.3.1 Prototypical Networks

Prototypical networks [Snell et al., 2017] use the mean of embedded support examples to represent novel class prototypes and classifies query examples by comparing their distances to the class prototypes. The m -dimensional embedding $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$ is learned during training. The mean $c_k \in \mathbb{R}^m$ of support examples for class k is computed by

$$c_k = \frac{1}{N} \sum_{\substack{(\mathbf{x}_i, y_i) \in S \\ y_i = k}} f(\mathbf{x}_i). \quad (2.8)$$

As illustrated in Figure 2.1, the query example \mathbf{x} is classified by producing a distribution over the K classes using the Euclidean distance between its embedding $f(\mathbf{x})$ and the class centers c_k

$$p(y = k | \mathbf{x}) = \frac{\exp(-\|f(\mathbf{x}) - c_k\|_2^2)}{\sum_{k'} \exp(-\|f(\mathbf{x}) - c_{k'}\|_2^2)}. \quad (2.9)$$

2.3.2 Relation Networks

Instead of using the Euclidean distance to compare the samples in the embedding space, relation networks [Sung et al., 2018], shown in Figure 2.2, learn a similarity function between support examples and the query example to classify images from unseen classes. The relation score r_k between the samples \mathbf{x}_i of class k and the query \mathbf{x} is computed by

$$r_k = g([c_k, f(\mathbf{x})]), \quad (2.10)$$

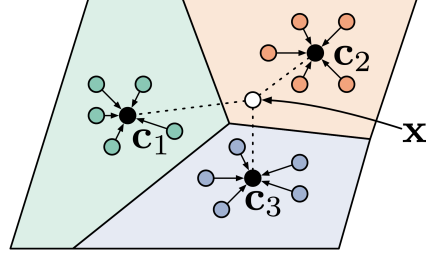


Figure 2.1: *Prototypical Networks classification in the embedding space. Image from [Snell et al., 2017].*

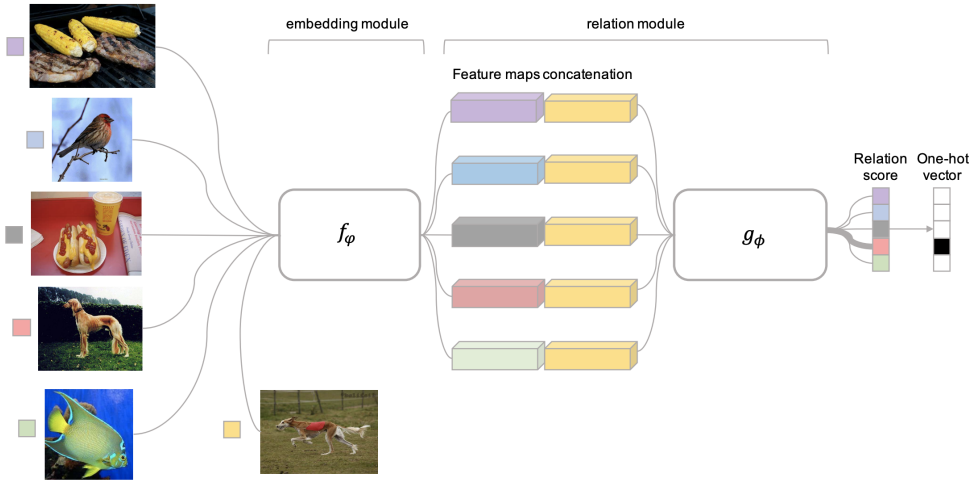


Figure 2.2: *Architecture of the relation networks for 1-shot, 5-way classification problem using a query example. Image from [Sung et al., 2018].*

where g is the learnable relation module, c_k is computed using Equation (2.8), and $[\cdot, \cdot]$ represents the concatenation operation. The class with the highest relation score is returned as class prediction y for the query example \mathbf{x} , i.e., $y = \operatorname{argmax}_{k \in [K]} r_k$.

2.4 Object Detection

In the object detection problem, the task is to find the locations of objects from a pre-defined set of categories in an image. The locations are usually defined with coordinates of axis-aligned bounding boxes that tightly enclose the instances in the image. There have been many different object detection approaches proposed in computer vision literature. In this thesis, we only consider the region-based Faster-RCNN [Ren et al., 2015] method. The conceptual Faster-RCNN architecture is presented in Figure 2.3a. It has two stages:

1. In the first stage, the first stage feature extractor, implemented by a Convolutional Neural Network (CNN), is employed to provide feature maps repre-

presenting the input image. These feature maps are then fed to a Region Proposal Network (RPN) to determine bounding box region proposals that may contain an object. The RPN, illustrated in Figure 2.3b, is a simple convolutional network that predicts k bounding boxes and their corresponding foreground/background scores for each spatial point in the feature map. The k bounding boxes, called anchor boxes, are pre-defined boxes with different aspect ratios with respect to the original image size. A classification layer produces the foreground/background scores, and a regression layer refines the coordinates of the given anchor boxes. These layers are supervised using a training dataset with bounding box annotations. Notice that for the RPN network, the classification layer does not disambiguate different classes and is hence called a *class-agnostic classifier*. The proposals with the highest foreground scores (usually 300 proposals are kept) after applying a *non-maximum suppression* (NMS) operation are selected for the second stage.

2. In the second stage, the top bounding box region proposals and the feature map are input to an ROI align or pooling layer. The ROI align layer crops the feature map using the proposals bounding box coordinates and resizes them to a given size. The resized features are fed to a second stage feature extractor. The final feature for each region is then fed to classification and regression heads. The classification head predicts the class of the corresponding region to be one of the c foreground classes or a background class c_\emptyset . The regression head further refines the region bounding box for each of the c classes. Finally, an NMS operation followed by score thresholding determines the predicted bounding boxes and their corresponding classes.

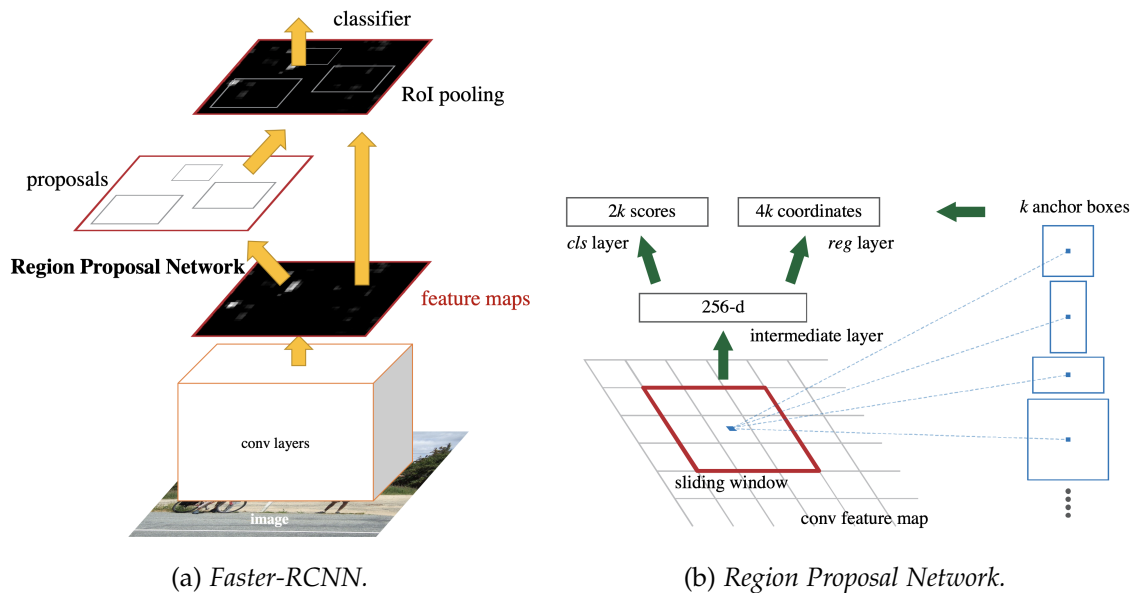


Figure 2.3: *Faster-RCNN and RPN architectures. Images from [Ren et al., 2015].*

2.5 Weakly Supervised Object Localization

In Weakly Supervised Object Localization (WSOL), only image-level annotations are provided in contrast to supervised object detection, where bounding box annotations are available during training.

We review the MIL-based algorithms among other branches in WSOL [Bilen and Vedaldi, 2016; Tang et al., 2014]. These approaches exploit alternating optimization to learn a detector and the optimal selection jointly. The algorithm iteratively alternates between re-localizing the objects given the current detector and re-training the detector given the current selection. In recent years, alternating optimization schemes combined with deep neural networks have been the state-of-the-art in WSOL [Wan et al., 2019; Gao et al., 2019; Zhu et al., 2017]. However, due to the non-convexity of the objective function, these methods are prone to a local minimum which typically leads to sub-optimal results [Bilen et al., 2015; Wan et al., 2018] e.g., selecting the salient parts instead of the whole object. Addressing this issue has been the main focus of research in WSOL in recent years [Cinbis et al., 2016; Kumar et al., 2010; Wan et al., 2019]. In multi-fold [Cinbis et al., 2016] learning, a weakly supervised dataset is split into separate training and testing folds to avoid overfitting. Kumar et al. [2010] propose an iterative self-paced learning algorithm that gradually learns from easy to hard samples to avoid getting stuck in bad local optimum points. Wan et al. [2019] propose a continuation MIL algorithm to smooth out the non-convex loss function in order to alleviate the local optimum problem systematically.

Transfer learning is another way to improve WSOL performance. These approaches utilize the information in a fully annotated dataset to learn an improved object detector on a weakly supervised dataset [Uijlings et al., 2018; Hoffman et al., 2016; Rochan and Wang, 2015; Guillaumin and Ferrari, 2012]. They leverage the common visual information between object classes to improve the localization performance in the target weakly supervised dataset. The fully annotated source dataset is used to learn a class-agnostic objectness measure in a standard knowledge-transfer framework. This measure is incorporated during the alternating optimization step to steer the detector toward objects and away from the background [Uijlings et al., 2018]. Although the objectness measure is a powerful metric in differentiating between background and foreground, it does not discriminate between different object classes. Several works have utilized pairwise similarity measures for improving WSOL [Shaban et al., 2019; Deselaers et al., 2010; Tang et al., 2014]. Deselaers et al. [2010] frame WSOL as a graph labeling problem with pairwise and unary potentials and progressively adapt the potential functions to learn weakly supervised classes. Tang et al. [2014] utilize the pairwise similarity between proposals to capture the inter-class diversity for the co-localization task. Hayder et al. [2014, 2015] use pairwise learning for object co-detection. The method in [Hayder et al., 2014, 2015] assumes Gaussian edge potentials, which is somewhat limited compared to flexible deep comparators. Their method does not assume the target objects are from novel classes. Also, their method does not work in the knowledge-transfer setting, which we think is more practical. Furthermore, [Hayder et al., 2015; Deselaers et al.,

2010] require computing all pairwise potentials, which is the bottleneck in large-scale settings.

Co-localization [Li et al., 2016], co-segmentation [Li et al., 2018; Vicente et al., 2011], and co-saliency [Zhang et al., 2015] methods have the same kind of output as weakly-supervised object localization, but they typically do not utilize negative examples. More recently, several methods were developed for localizing the common novel object under the few-shot setting [Hu et al., 2019; Siam et al., 2020]. SILCO [Hu et al., 2019] finds the common object by computing a dense similarity map between each image in the support set and the query while only exploring the similarity among support images using their coarse image-level features via a global average pooling. Although using global average pooling reduces the computation, ignoring the dense similarities among support images negatively affects the common object localization.

2.5.1 Knowledge-Transfer in MI-SVM WSOD

We briefly describe the method proposed in [Uijlings et al., 2018] for knowledge-transfer in weakly supervised object localization. This method is essentially MI-SVM (Algorithm 2) augmented with objectness scores from a pre-trained Faster-RCNN trained on the training dataset $\mathcal{D}_{\text{train}}$. In this method, each image is represented as a bag of bounding box proposals \mathcal{B} . Learning is performed on one target class $y \in \llbracket c \rrbracket$ at a time. The support set is split into positive images which has the target class and a negative set of images without the target class. Then, a linear SVM appearance model is employed to iteratively learn class y by alternating between two steps:

- **Re-training:** Train a binary SVM given the currently selected proposals from the positive images and the proposals in negative images.
- **Re-localization:** Given the current SVM select the proposal with the highest score from each positive image. In [Uijlings et al., 2018], the re-localization is guided by a class-agnostic objectness measure to guide the selection toward objects. Therefore, the selection for a positive image \mathbf{x} with bounding box proposals \mathcal{B} is updated as

$$B^* = \underset{B \in \mathcal{B}}{\operatorname{argmax}} \quad \text{SVM}(\mathbf{x}, B) + \gamma O(\mathbf{x}, B), \quad (2.11)$$

where O is the objectness model and γ adjusts its importance.

The algorithm is initialized with complete image bounding box proposal and alternates between above steps until convergence. Finally, test proposal \mathbf{x} from the test set $\mathcal{D}_{\text{test}}$ is scored using the SVM trained for each class. A hard negative mining in [Uijlings et al., 2018] is employed to improve the performance of the classifier. In this approach the negative set is initialized with full bounding box negative image features and the hardest negative proposal within each image is added to the negative set after each re-training step.

2.6 Confidence Calibration of Neural Networks

Deep neural networks have shown remarkable accuracy in classification tasks. However, it is known that these models do not have calibrated confidence scores [Guo et al., 2017]; meaning that the confidence score they assign to a set of events does not match the true frequency of those events. These high capacity models are known to overfit the NLL loss resulting in over-confident predictions. Various methods have been proposed to adjust the models such that the confidence scores become calibrated.

Obtaining well calibrated predictors has been studied by meteorologists and statisticians for several decades. Brier [Brier, 1950] studied the verification of weather forecasts when they are expressed in terms of probabilities and introduced the famous Brier score. Many follow up works studied the concept of reliability [Murphy and Winkler, 1977], proper scoring rules [Winkler and Murphy, 1968], and calibration and refinement [DeGroot and Fienberg, 1983] in a similar context. In general, there are three different approaches to enforce better confidence calibration in modern neural networks: (i) post-hoc calibration methods that learn a calibration function, usually using a held-out calibration data, that transforms the pre-trained model’s logits (or confidence scores) such that the transformed confidences become calibrated. (ii) Regularization-based methods that introduce regularization in the training procedure of models to avoid overfitting to the NLL loss. (iii) Bayesian neural networks that derive the uncertainty of the predictions by making stochastic perturbation of the original model. We briefly introduce these approaches in the following paragraphs. However, in this thesis, we only consider post-hoc calibration methods.

Many different post-hoc calibration methods have been studied in the literature [Platt, 1999; Guo et al., 2017; Kull et al., 2019, 2017b,a; Kumar et al., 2019]. Their main difference is in the parametric family of the calibration function. In Platt scaling [Platt, 1999], scale and shift parameters $a, b \in \mathbb{R}$ are used to transform the scalar logit output $x \in \mathbb{R}$ i.e. $f(x) = ax + b$ of a binary classifier. Temperature scaling [Guo et al., 2017] is a simple extension of Platt scaling for multi-class calibration in which only a single scalar temperature parameter is learned. Dirichlet calibration [Kull et al., 2019] allows learning within a richer linear functions family $f(\mathbf{x}) = W\mathbf{x} + \mathbf{b}$, where $W \in \mathbb{R}^{c \times c}$ and $\mathbf{b} \in \mathbb{R}^c$ but the learned calibration function may also change the decision boundary of the original model; [Kull et al., 2019] suggested regularizing the off-diagonal elements of W to avoid overfitting. Earlier works like isotonic regression [Zadrozny and Elkan, 2002], histogram binning [Zadrozny and Elkan, 2001], and Bayesian binning [Zadrozny and Elkan, 2002] are also post-hoc calibration methods.

In contrast to post-hoc calibration methods, several researches proposed to modify the training process to learn a calibrated network in the first place. Data augmentation methods [Thulasidasan et al., 2019; Yun et al., 2019] overcome overfitting by enriching the training data with new artificially generated pseudo data points and labels. Mixup [Zhang et al., 2018] creates pseudo data points by computing the convex combination of randomly sampled pairs. Cutmix [Yun et al., 2019] uses a more efficient combination algorithm specifically designed for image classification in

which two images are combined by overlaying a randomly cropped part of the first image on the second image. In label smoothing [Pereyra et al., 2017; Müller et al., 2019], the training loss is augmented to penalize high confidence outputs. To discourage overconfident predictions, [Seo et al., 2019] modifies the original NLL loss by adding a cross-entropy loss term with respect to the uniform distribution. Similarly, [Kumar et al., 2018] adds a calibration regularization to the NLL loss via kernel mean embedding.

Bayesian neural networks [Gal and Ghahramani, 2016; Maddox et al., 2019] derive the uncertainty of the prediction by making stochastic perturbations of the original model. Notably, [Gal and Ghahramani, 2016] uses dropout as approximate Bayesian inference. [Maddox et al., 2019] estimates the posterior distribution over the parameters and uses samples from this distribution for Bayesian model averaging. These methods are computationally inefficient since they typically feed each sample to the network multiple times.

Efficient Inference for Finding Common Objects Across Small Image Collections

Few-shot learning algorithms have gained a lot of attention in recent years. While these methods perform well in tasks such as few-shot classification, their applicability is relatively restrained in real-world scenarios such as field robotics in which annotating the few examples is conceived as an obstruct. In this chapter, we study the problem of finding common objects among a few weakly supervised bags.

We pose the problem as a minimum cost graph labeling optimization on a densely connected graphical model. Each graph node represents a bag where every element in the bag corresponds to one possible label at the node. First, we employ the relation module to learn pairwise potentials from a fully annotated source dataset and propose a novel attention mechanism to incorporate the information in the negative bag as our unary potentials. Next, we present an efficient greedy inference algorithm that takes advantage of the inherent structure of the problem that a common object among a set of positive bags is also a common object among any subset of those bags. We eliminate the requirement of computing pairwise similarities between all elements in all of the bags using this simple heuristic. Without reducing the performance, our inference method reduces the computational time by $\sim 85\%$ compared to the other known graph inference methods for the problem of few-shot object co-localization.

3.1 Introduction

We study the problem of finding common objects across a collection containing a small number of bags. A bag is a set with multiple elements, each belonging to a single category. The collection contains multiple *positive* bags and an optional *negative* bag with respect to a given *target* category. The target category and its objects have been unknown to our algorithm during training and testing. A bag is labeled as *positive* if at least one of its elements is from the target category and *negative* otherwise. The task is defined as finding one instance of the target category



(a) Few-shot object co-localization. Each image in the top row corresponds to a positive bag containing a set of region proposals (shown with bounding boxes). The task is to select one region per positive bag such that the selected regions contain a shared category (green bounding boxes). Region proposals from the images in the bottom row form a negative bag as they do not contain the common object. The negative bag is optional here but can reduce ambiguity. For example, since a dog is present in the negative bag, it can not be the desired common object.



(b) Few-shot common object recognition. Each rounded box represents a bag containing multiple images as its elements. Green boxes are positive bags and the negative bag is shown by a red rounded box. Here, the common object class is camera which is present in all of the green boxes and **not** present in the red box.

Figure 3.1: Examples of finding common objects problems we consider in this chapter.

in each positive bag. In this scenario, we overcome the difficulty of annotating all the elements in a bag by only providing bag-level annotations.

Object co-segmentation, co-localization, co-saliency, tracking, and video instance segmentation are among the popular computer vision applications that have been expressed as a form of the finding common objects problem [Vicente et al., 2011; Faktor and Irani, 2013; Hsu et al., 2018; Fu et al., 2014; Babenko et al., 2009; Shaban et al., 2019; Zhang et al., 2015]. As an example, consider the problem of few-shot ob-

ject co-localization in Figure 3.1a. Each image corresponds to a bag in this problem, and the region proposals extracted from that image are the bag elements. The goal is to select one region proposal per positive image such that all the selected proposals share a common (and non-background) object category. While our method considers the general problem of finding common objects, we evaluate it on two different tasks (see Figure 3.1): (i) few-shot common object recognition and (ii) few-shot object co-localization.

A common approach to solve the problem of finding common objects is to formulate it as a minimum cost graph labeling optimization on a densely connected and undirected graphical model with learned pairwise and unary potentials [Vicente et al., 2011]. Each node of the graph represents a positive bag in which every element in the bag corresponds to one possible label at the node. The optimization objective, called graph labeling, is to assign a single label to each node in the graph such that the selected labels correspond to the common objects. The pairwise potential functions measure how well two elements from two distinct positive bags are from the same (foreground) class. Similarly, the unary potential function for an element estimates how it differs from the elements in the negative bag.

We follow the relation network [Sung et al., 2018; Shaban et al., 2019] to compute pairwise potentials. In addition, we propose a novel method to compute the relation between an element in a positive bag and all the elements in the negative bag to estimate the unary potential.

After computing unary and pairwise potentials, a simple merge-and-prune inference heuristic is used to find a minimum-cost labelling. This provides a simple but effective solution to the NP-hard problem of optimal graph labelling. The proposed greedy search algorithm for finding the common object is based on the following observation: *the common object for the complete problem is a common object in any subset of the bags as well.*

We note that graphical models have been previously used for Multiple-Instance Learning (MIL) problems [Deselaers and Ferrari, 2010; Hajimirsadeghi et al., 2013]. However, our method is motivated by recent few-shot classification methods to find common objects of novel classes.

3.2 Problem Setup

We consider a bag $\mathcal{B} = \{\mathbf{e}_i\}$ where an element $\mathbf{e}_i \in \mathbb{R}^d$ is represented by a d -dimensional feature vector. We denote $y(\mathbf{e}) \in \mathcal{C} \cup \{c_\emptyset\}$ the label for element \mathbf{e} . Here, \mathcal{C} is a set of categories of interest and c_\emptyset represents the background class. Given a class $c \in \mathcal{C}$, we can also define the unary labeling function

$$y_c(\mathbf{e}) = \begin{cases} 1 & \text{if } y(\mathbf{e}) = c \\ 0 & \text{otherwise.} \end{cases} \quad (3.1)$$

The label for bag \mathcal{B} is written as $\mathcal{Y}(\mathcal{B}) = \bigcup_{\mathbf{e} \in \mathcal{B}} y(\mathbf{e}) - \{c_\emptyset\}$. Let $\Upsilon_c(\mathcal{B}) \in \{0, 1\}$

denote the binary bag label which indicates the presence/absence of class c in bag \mathcal{B} . With this notation, a collection \mathcal{T}_c given a class c is defined as a set with *positive* bags that contain at least one element \mathbf{e} with $y_c(\mathbf{e}) = 1$ and a *negative* bag with all elements $\bar{\mathbf{e}}$ having $y_c(\bar{\mathbf{e}}) = 0$. Formally, a collection is an ordered set $\mathcal{T}_c = (\mathcal{B}_1, \dots, \mathcal{B}_N, \bar{\mathcal{B}})$ comprising of positive bags \mathcal{B}_j and an optional negative bag $\bar{\mathcal{B}}$ with the following two properties:

1. $\forall j \in \{1 \dots N\}, \exists \mathbf{e} \in \mathcal{B}_j$ s.t. $y_c(\mathbf{e}) = 1$.
2. $\forall \bar{\mathbf{e}} \in \bar{\mathcal{B}}, y_c(\bar{\mathbf{e}}) = 0$.

For simplicity, we also introduce a pairwise labeling function between pairs of elements. The pairwise labeling function $r : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \{0, 1\}$ is designated to output 1 when two elements belong to the same object class and 0 otherwise, i.e.,

$$r(\mathbf{e}, \mathbf{e}') = \begin{cases} 1 & \text{if } y(\mathbf{e}) = y(\mathbf{e}') \neq c_\emptyset \\ 0 & \text{otherwise.} \end{cases} \quad (3.2)$$

Likewise, given a class c , two proposals are related under the class conditional pairwise labeling function $r_c : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \{0, 1\}$ if they both belong to class c . Furthermore, we also define the relation between an element \mathbf{e} from a positive bag and a negative bag $\bar{\mathcal{B}}$ as:

$$r(\mathbf{e}, \bar{\mathcal{B}}) = \max_{\bar{\mathbf{e}} \in \bar{\mathcal{B}}} r(\mathbf{e}, \bar{\mathbf{e}}). \quad (3.3)$$

The input to our algorithm is a collection \mathcal{T}_c with a previously unseen class c . The only information we have about the collection is whether a bag is positive or negative in the collection. The task is to output a *selection* of elements, namely an ordered set $S = (\mathbf{e}_1, \dots, \mathbf{e}_N)$ where \mathbf{e}_j is from positive bag \mathcal{B}_j , such that the selected elements are from class c , i.e., $\forall \mathbf{e}_j \in S, y_c(\mathbf{e}_j) = 1$.

Energy function. Here, we follow the definition in [Shaban et al., 2019] to formulate the problem of finding common object as finding a selection S that minimizes the following energy function:

$$E(S \mid \bar{\mathcal{B}}) = \sum_{\substack{\mathbf{e}_i, \mathbf{e}_j \in S \\ i > j}} \psi_\theta^P(\mathbf{e}_i, \mathbf{e}_j) + \eta \sum_{\mathbf{e}_i \in S} \psi_\beta^U(\mathbf{e}_i \mid \bar{\mathcal{B}}), \quad (3.4)$$

where $\psi_\theta^P(\cdot, \cdot)$ and $\psi_\beta^U(\cdot \mid \bar{\mathcal{B}})$ are pairwise and unary potential functions with trained parameters θ and β , and hyperparameter $\eta \geq 0$ controls the effect of the unary terms. We set $\eta = 0$ when there is no element in the negative bag $\bar{\mathcal{B}}$. We will describe the neural network architecture of both unary and pairwise functions in Section 3.3. The pairwise potential function favors selecting pairs having the same object class. The unary potential discourages choosing elements similar to any element in the negative bag.

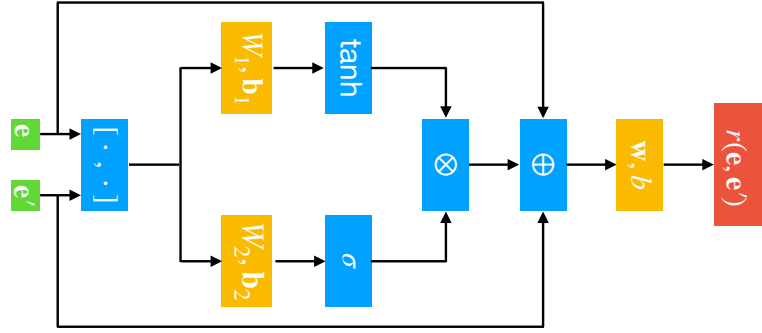


Figure 3.2: Architecture of the relation module used for unary and pairwise potentials.

3.2.1 Dataset Setup

A dataset \mathcal{D} is a set of bags with elements and their corresponding classes. For a bag \mathcal{B} in a dataset, $\mathcal{Y}(\mathcal{B}) \subseteq \mathcal{C}$ is inferred from the elements' classes as mentioned in Section 3.2. To sample collections \mathcal{T}_c from a dataset \mathcal{D} , we first sample a class $c \in \mathcal{C}$. Then, we randomly sample N positive bags \mathcal{B}_j with $Y_c(\mathcal{B}_j) = 1$. We also sample a predefined number of bags with $Y_c(\mathcal{B}) = 0$ and consider their union as a single negative bag $\bar{\mathcal{B}}$. We use the notation $\mathcal{T}_c \sim \mathcal{D}$ to indicate that a random collection for a target class c is drawn from the dataset \mathcal{D} .

We assume having access to a fully annotated training dataset $\mathcal{D}_{\text{train}}$ with classes $\mathcal{C}_{\text{train}}$. We build the groundtruth pairwise and unary labeling functions for $\mathcal{D}_{\text{train}}$ as defined in Section 3.2. To sample a positive bag of size B in the few-shot common object recognition task, we first randomly sample a target class c . Then, we sample one image with label c and $B - 1$ images with any class $c' \in \mathcal{C}$. Similarly, for a negative bag of size \bar{B} , we sample \bar{B} images with classes $c' \neq c$. For the few-shot object co-localization task, a positive bag with respect to a class $c \in \mathcal{C}$ corresponds to an image that contains at least one instance of class c . The elements in the bag are generated by a pre-trained region proposal network (RPN) on the fully supervised training dataset $\mathcal{D}_{\text{train}}$. A negative bag in this case is generated by the union of all the region proposals from \bar{B} images which do not contain class c .

Evaluations are performed on sampled collections from a test dataset $\mathcal{T}_c \sim \mathcal{D}_{\text{test}}$. There are no shared element between the training and test datasets. In addition, the set of classes $\mathcal{C}_{\text{test}}$ used for the test dataset is disjoint from the set of classes used during training, i.e., $\mathcal{C}_{\text{test}} \cap \mathcal{C}_{\text{train}} = \emptyset$. At test time we only know whether a bag is positive or negative with respect to some unknown class c . The ground-truth labeling functions are unknown to the algorithm and are only used for evaluation.

3.3 Potential Functions

Relation module. We adopt the *relation module* [Sung et al., 2018; Shaban et al., 2019] for learning a similarity measure between a pair of elements. The relation

module learns a mapping $\hat{r}_\phi : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ with parameters ϕ , mapping the feature vector of two elements to a scalar value. It consists of an embedding module $f(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ followed by a linear comparator function:

$$f(\mathbf{e}, \mathbf{e}') = \tanh(W_1[\mathbf{e}, \mathbf{e}'] + \mathbf{b}_1)\sigma(W_2[\mathbf{e}, \mathbf{e}'] + \mathbf{b}_2) + (\mathbf{e} + \mathbf{e}')/2, \quad (3.5)$$

$$\hat{r}_\phi(\mathbf{e}, \mathbf{e}') = \mathbf{w}^\top f(\mathbf{e}, \mathbf{e}') + b, \quad (3.6)$$

where $W_1, W_2 \in \mathbb{R}^{d \times 2d}$ and vectors $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^d$ are the parameters of the embedding module, $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ are the parameters of the comparator, $[\cdot, \cdot]$ representing concatenation, and $\tanh(\cdot)$ and $\sigma(\cdot)$ are hyperbolic tangent and sigmoid activation functions respectively, applied component-wise to vectors in \mathbb{R}^d (see Figure 3.2).

Pairwise potentials. The pairwise potential function should have lower energy when its input elements are more similar. Therefore, as described in [Shaban et al., 2019], we define the pairwise potential function between two elements $(\mathbf{e}, \mathbf{e}')$ as the negative of the output of the pairwise relation module with parameters θ : $\psi_\theta^p(\mathbf{e}, \mathbf{e}') = -\hat{r}_\theta(\mathbf{e}, \mathbf{e}')$. We employ sigmoid cross-entropy as the pairwise loss for a sampled collection $\mathcal{T}_c \sim \mathcal{D}_{\text{train}}$:

$$\mathcal{L}^p(\mathcal{T}_c) = -\frac{1}{N_p} \sum_{(\mathbf{e}, \mathbf{e}') \sim \mathcal{T}_c} (r(\mathbf{e}, \mathbf{e}') \log \sigma(\hat{r}_\theta(\mathbf{e}, \mathbf{e}')) + (1 - r(\mathbf{e}, \mathbf{e}')) \log \sigma(1 - \hat{r}_\theta(\mathbf{e}, \mathbf{e}'))), \quad (3.7)$$

where the sum is over all the positive pairs in the collection, $N_p = B^2 N(N-1)/2$ is the total number of such pairs, relation $r(\cdot, \cdot)$ defined in Equation (3.2) is the ground-truth pairwise labeling function, and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. In practice, we train the pairwise potential function on collections with $N = 2$ positive bags. For the few-shot object co-localization task where the number of background elements (i.e., region proposals) are much more than the foreground ones, we sub-sample the elements in each bag pair with 25% positive (or same class) and 75% negative (different classes) pairs.

Unary potentials. The purpose of unary potential function $\psi_\beta^u(\mathbf{e} | \bar{\mathcal{B}})$ is to determine if there is any similarity between the element \mathbf{e} and the elements $\bar{\mathbf{e}} \in \bar{\mathcal{B}}$. If there is such a similarity, the value of the unary potential function should be high. A natural choice would be using

$$\psi_\beta^u(\mathbf{e} | \bar{\mathcal{B}}) = \max_{\bar{\mathbf{e}} \in \bar{\mathcal{B}}} \hat{r}_\beta(\mathbf{e}, \bar{\mathbf{e}}), \quad (3.8)$$

where β is the (new) set of parameters for unary relation function \hat{r}_β . Nevertheless, this results in sparse gradients and using only the maximum element $\bar{\mathbf{e}}$ in the negative bag can be noisy when there are multiple similar elements in $\bar{\mathcal{B}}$ to the element \mathbf{e} . To alleviate the problems, we propose to use a soft-attention based averaging of the relations so that all elements in the negative bag can contribute to the unary potential

function proportionally to their similarity to \mathbf{e}

$$\psi_{\beta,T}^U(\mathbf{e} \mid \vec{\mathcal{B}}) = \frac{\sum_{\bar{\mathbf{e}}_k \in \vec{\mathcal{B}}} \hat{r}_{\beta}(\mathbf{e}, \bar{\mathbf{e}}_k) \exp(\hat{r}_{\beta}(\mathbf{e}, \bar{\mathbf{e}}_k)/T)}{\sum_{\bar{\mathbf{e}}_k \in \vec{\mathcal{B}}} \exp(\hat{r}_{\beta}(\mathbf{e}, \bar{\mathbf{e}}_k)/T)}. \quad (3.9)$$

Here, $T > 0$ is the temperature scaling parameter that controls the behavior of the unary potential function. We remark that when $T \rightarrow +\infty$, the unary potential function captures the mean value of relations between \mathbf{e} and all elements $\bar{\mathbf{e}} \in \vec{\mathcal{B}}$ and it meets the maximum value like Equation (3.8) when $T \rightarrow 0$. The unary loss for a sampled collection \mathcal{T}_c is defined by sigmoid cross-entropy loss between $r(\mathbf{e}, \vec{\mathcal{B}})$ (defined in Equation (3.3)) and the predicted unary potential function

$$\mathcal{L}^U(\mathcal{T}_c) = -\frac{1}{NB} \sum_{\mathcal{B}_j \in \mathcal{T}_c} \sum_{\mathbf{e} \in \mathcal{B}_j} (r(\mathbf{e}, \vec{\mathcal{B}}) \log \sigma(\psi_{\beta,T}^U(\mathbf{e} \mid \vec{\mathcal{B}})) + (1 - r(\mathbf{e}, \vec{\mathcal{B}})) \log \sigma(1 - \psi_{\beta,T}^U(\mathbf{e} \mid \vec{\mathcal{B}}))). \quad (3.10)$$

Denoting all the parameters of unary and pairwise loss functions as $\Theta = (\theta, \beta, T)$, we minimize the total loss function using the collections sampled from the training dataset:

$$\operatorname{argmin}_{\Theta} \mathbb{E}_{\substack{\mathcal{C} \sim \mathcal{C}_{\text{train}} \\ \mathcal{T}_c \sim \mathcal{D}_{\text{train}}}} \left\{ \mathcal{L}^U(\mathcal{T}_c) + \mathcal{L}^P(\mathcal{T}_c) \right\}. \quad (3.11)$$

Optimizing the loss function by sampling collections is similar to the episodic training for few-shot learning methods [Finn et al., 2017; Snell et al., 2017]. The training episodes are similar to the collections that we want to test on. This can capture the prior distribution of related and unrelated pairs [Shaban et al., 2019].

Although both unary and pairwise potential functions utilize relation modules in our implementation, they have disjoint parameter sets. The reason is that the class distribution of their inputs is different. The pairwise potential function only compares pairs of elements in positive bags while the unary potential function computes the relation between an element from a positive bag to all elements in a negative bag. This architectural choice has been validated in [Shaban et al., 2019].

3.3.1 Inference

Minimizing the energy functions defined in Equation (3.4) is an NP-hard problem in general. However, it has a mature field and numerous methods, e.g., AStar search [Bergtholdt et al., 2010], loopy belief propagation [Weiss and Freeman, 2001], and Tree-Rewighted Sequential (TRWS) [Kolmogorov, 2006], each with its own pros and cons, have been proposed to find an approximate solution.

Our approach is designed to decompose the overall problem into smaller subproblems, solve them, and combine their solutions to find a solution to the overall problem. This is based on the observation that “a solution to the overall problem will also be a valid solution to any of the subproblems.” Let $\mathcal{T}_c(p, q) = \{\mathcal{B}_p, \mathcal{B}_{p+1}, \dots, \mathcal{B}_q\}$ be a subsequence of the positive bags in \mathcal{T}_c (see Figure 3.3). Then, a subproblem refers to finding a set of common object proposals \mathcal{P} for $\mathcal{T}_c(p, q)$ with low energy values;

\mathcal{P} represents a collection of proposed selections of elements from the sequence of bags $\mathcal{T}_c(p, q)$. The energy value for a selection $S_{p,q} \in \mathcal{P}$ is defined as sum of all pairwise and unary potentials in the subproblem, similar to how the energy function is defined for the overall problem in Equation (3.4).

The decomposition method starts at the root (i.e., full problem) and divides the problem into two disjoint subproblems¹ and recursively continues dividing each into two subproblems until each subproblem only contains a single bag \mathcal{B}_j . If $N = 2^Z$, then this can be represented as a full binary tree² where each node represents a subproblem. Let \mathcal{N}_j^l be the j -th node at level l . Then root node \mathcal{N}_1^Z represents the full problem, nodes \mathcal{N}_j^l at any given level l represent disjoint subproblems of the same size, and the leaf nodes, \mathcal{N}_j^0 , at level 0 of the tree each represent a subproblem with only one positive bag \mathcal{B}_j .

Each level in the tree maintains a set of partial solutions to the root problem. Computation starts at the lowest level (leaf nodes) where each partial solution is simply one of the elements for all elements in the bag. At the next level, each node combines the partial solutions from its child nodes and prunes the resulting set to form a new set of partial solutions for its own subproblem, which in turn is used as input to nodes at the next level in the tree and so on until we reach the root node, which is the output for the optimization. The joining procedure used to combine the partial solutions from two child nodes is described next.

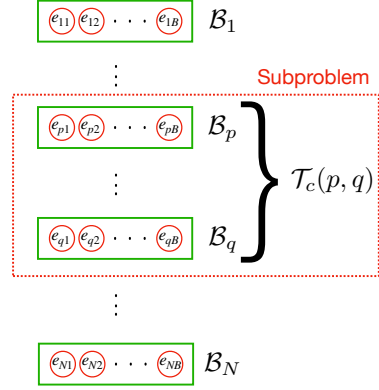


Figure 3.3: Each bag and its elements are shown by a horizontal green box and red circles, respectively. A subproblem between p -th and q -th bag is presented as $\mathcal{T}_c(p, q)$.

Joining. Node j at level l receives as input solution proposals \mathcal{P}_{2j-1}^{l-1} and \mathcal{P}_{2j}^{l-1} from its child nodes \mathcal{N}_{2j-1}^{l-1} and \mathcal{N}_{2j}^{l-1} . The joining operation simply concatenates every possible selection from the first set with every possible selection in the second set and forms a set of selection proposals \mathcal{X}_j^l for the subproblem

$$\mathcal{X}_j^l = \{[S^{\text{left}}, S^{\text{right}}] \mid S^{\text{left}} \in \mathcal{P}_{2j-1}^{l-1}, S^{\text{right}} \in \mathcal{P}_{2j}^{l-1}\}$$

where $[\cdot, \cdot]$ concatenates two selection sequences. We denote the joining operation by the Cartesian product notation, i.e., $\mathcal{X}_j^l = \mathcal{P}_{2j-1}^{l-1} \times \mathcal{P}_{2j}^{l-1}$.

Pruning. Since combining the partial solutions from two nodes results in a quadratic increase in the number of partial solutions, the number of potential solutions grows

¹We perform the division randomly. However, finding an optimal policy for the division is an interesting direction for future works.

²This is without loss of generality since zero padding could be used if the number of positive bags is not a power of 2.

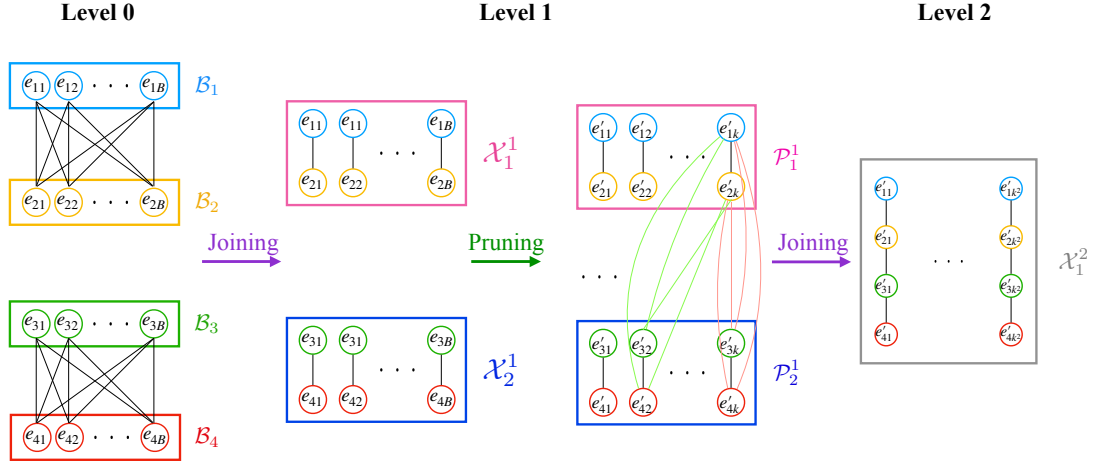


Figure 3.4: An example of the greedy inference algorithm on a collection with $N = 4$ positive bags. The first “Joining” operation (left) creates solution proposals of size two and the second one (right) creates solution proposals of size four. The “Pruning” operation (middle) prunes the solution proposals with high energy values.

exponentially as we ascend the tree. Also, not all the generated partial solutions contain a common object. Therefore, we use a pruning algorithm $\mathcal{P}_j^l = \text{prune}(\mathcal{X}_j^l; k)$ that picks the k selections with the lowest energy values. The energy values for each subproblem can also be efficiently computed from bottom to top. At the lowest level, the energy for each selection is the unary potential from Equation (3.9),

$$E_j^0(S_{j,j}) = \eta \psi_\beta^U(\mathbf{e}_j \mid \bar{\mathcal{B}}) \quad \forall \mathbf{e}_j \in \mathcal{P}_j^0 = \mathcal{B}_j, \quad (3.12)$$

Note that selection $S_{j,j} = (\mathbf{e}_j)$ consists of only one image. Starting at the leaves, energy in all nodes can be computed recursively. Let $S \in \mathcal{X}_j^l$ be formed by joining two selection proposals $S^{\text{left}} \in \mathcal{P}_{2j-1}^{l-1}$ and $S^{\text{right}} \in \mathcal{P}_{2j}^{l-1}$. The energy function $E_j^l(S)$ can be factored as

$$E_j^l(S) = E_{2j-1}^{l-1}(S^{\text{left}}) + E_{2j}^{l-1}(S^{\text{right}}) + P(S^{\text{left}}, S^{\text{right}}) \quad (3.13)$$

where $P(\cdot, \cdot)$ is the sum of all pairwise potentials on edges joining the two subproblems and is computed on the fly. Algorithm 3 summarizes the method. Also see Figure 3.4 for an example of the joining and pruning operations. A good value of k in the pruning method depends on the ambiguity of the task. It is possible to construct an adversarial example that needs *all* possible proposals at the root node to find the optimal solution. However, in practice, we found that k does not need to be large to achieve good performance. Importantly, unlike other methods, this algorithm does not necessarily compute all of the pairwise potentials. For example, if an object class only appears in a small subproblem, the images of that class will get removed by nodes whose subproblem size is large enough. Thus, in the next level of

Algorithm 3: Greedy Optimization Algorithm

Input: $\mathcal{T}_c = \{\mathcal{B}_1, \dots, \mathcal{B}_N, \bar{\mathcal{B}}\}$, and $N = 2^Z$.
Output: Selection $S = (\mathbf{e}_1, \dots, \mathbf{e}_N)$
 $\mathcal{P}_j^0 = \mathcal{B}_j \quad \forall j \in [1, \dots, N]$
 $E_j^0(S_{j,j}) = \eta \psi_{\beta}^U(\mathbf{e}_i \mid \bar{\mathcal{B}}) \quad \forall \mathbf{e}_j \in \mathcal{P}_0^j, j \in [1, \dots, N]$
for $l \leftarrow 1$ **to** Z **do**
 for $j \leftarrow 1$ **to** 2^{Z-l} **do**
 $\mathcal{X}_j^l \leftarrow \mathcal{P}_{2j-1}^{l-1} \times \mathcal{P}_{2j}^{l-1}$ (joining)
 Compute \mathcal{X}_j^l Energies According to Equation (3.13)
 $\mathcal{P}_j^l \leftarrow \text{prune}(\mathcal{X}_j^l; k)$ (pruning)
return $S \in \mathcal{P}_1^Z$ with the minimum energy

the tree, the pairwise potentials between those images and other images is no longer required. In general, the number of pairwise potentials computed depends on both the value of k and the dataset. We observed that only a small fraction of the total pairwise potentials were required in our experiments.

Complexity. The time complexity of computing unary potentials is $O(\bar{B}BN)$. The upper bound time complexity for computing all possible pairs is $O(N^2B^2)$. Since our method does not necessarily compute all the pairs, this time complexity is lower bounded by $\Omega(N^2)$. Overall, the proposed method is practical for medium size problems.

3.4 Experiments

We analyze different aspects of the method discussed in this chapter focusing on the greedy inference algorithm and the effectiveness of the proposed unary potential function. We refer the readers to [Shaban et al., 2019] for complementary information. To assess different architectural choices, we first evaluate our method on the few-shot common object recognition task (see Figure 3.1b). Next, we evaluate on the few-shot object co-localization task which has a more realistic scenario (Figure 3.1a).

As feature extractor for both few-shot common object recognition and few-shot object co-localization, we train deep network based architectures on the fully supervised dataset $\mathcal{D}_{\text{train}}$ for classification and object detection tasks respectively. To have fair comparisons, we use the deep network based features as bag elements in all of the competitive methods.

We make use of stochastic gradient descent with step-wise learning rate decay schedule to learn pairwise and unary potential functions. We denote the complete framework that uses the greedy optimization described in Algorithm 3 as “Ours” in the tables. We use grid search on the validation set to tune the hyperparameters of all the methods. We set the number of kept solution proposals k after every

pruning stage in the greedy optimization algorithm to be 300. All the experiments are performed on a machine with a single Nvidia GTX 2080ti GPU and a 4GHz AMD Ryzen Threadripper 1920X CPU with 12 cores.

3.4.1 Baseline Methods

We compare the greedy optimization algorithm to AStar [Bergtholdt et al., 2010] which has been used for object co-segmentation [Vicente et al., 2011] and the faster TRWS [Kolmogorov, 2006] which has been used for inference on MIL problems [Deselaers and Ferrari, 2010; Deselaers et al., 2012]. We use a highly efficient parallel implementation of these algorithms [Andres et al., 2012]. We also provide comparisons with the traditional SVM based [Hoffman et al., 2015; Andrews et al., 2003; Bunescu and Mooney, 2007] and more recent attention based method of [Ilse et al., 2018] for multiple-instance learning. Additionally, we consider a “baseline” method that uses cosine similarity on the extracted features $\mathbf{e} \in \mathbb{R}^d$ between pairs of elements as the relation modules in pairwise and unary potential functions to assess the effectiveness of the proposed learning strategy for both potential functions. We refer the readers to [Shaban et al., 2019] for more details about these methods.

3.4.2 Few-shot Common Object Recognition

Dataset. We employ the *miniImageNet* dataset [Vinyals et al., 2016] for the few-shot common object recognition task. Since it is a small-scale dataset, we can first assess our architectural choices easily. Once the architectural choices are fixed, we apply them for the large-scale few-shot object co-localization task. Statistical details of the *miniImageNet* dataset are presented in Table 3.1.

Table 3.1: Statistics of the standard split of the *miniImageNet* dataset [Ravi and Larochelle, 2017] used for the few-shot common object recognition task.

Images	60,000
Image size	84×84
Train classes	64
Validation classes	16
Test classes	20

Feature extractor. A Wide Residual Network (WRN) [Zagoruyko and Komodakis, 2016] with depth 28 and width factor 10 is trained on the training split as the feature extractor for each element $\mathbf{e} \in \mathbb{R}^{640}$ on this dataset. The network is trained using a standard 64-class classification task. Features for the last global average pooling layer before the linear classifier are used for all the methods.

Sampling collections. To sample a collection \mathcal{T}_c , we first randomly sample $M \leq |\mathcal{C}|$ classes and select one of them as the target class c . The positive and negative bags are

selected randomly from images of only these M classes as described in Section 3.2.1.

Evaluation metric. For an output selection $S = (\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N)$, we measure the *success rate* as our evaluation criterion for a sampled collection \mathcal{T}_c defined as

$$\text{Success rate}(S, \mathcal{T}_c) = \frac{1}{N} \sum_{\mathbf{e}_j \in S} y_c(\mathbf{e}_j) \quad (3.14)$$

We report the average success rate for 1000 randomly sampled collections from $\mathcal{D}_{\text{test}}$ along with 95% confidence interval.

Setting. We experiment with number of positive bags $N \in \{4, 8, 16\}$, bag sizes $B \in \{5, 10\}$, and negative bag size $\bar{B} \in \{10, 20\}$. We choose different values for the number of classes M in which we sample our collections from to have various levels of difficulties in the sampled collections. When the bag size B is 5 (or 10), we select a random value in the range $M \sim [5, 15]$ (or $M \sim [10, 20]$). Lower values of M make the problem more ambiguous by increasing the chance of generating other common objects in the subproblems. On the other hand, it increases the importance of the negative bag by increasing the chance of having more samples from each non-target class.

Results. Table 3.2 presents the results comparing the proposed method to other MIL based counterparts. The large gap between our method and the other MIL approaches validates our argument that traditional MIL approaches overfit to the training data in small data regimes while our method can better generalize in such scenarios. Our method can leverage the relation learning using the base dataset to compare if two elements are from the same class or not. At the same time, the multiple-instance learning methods in 3.2 could only use features of bag elements. Furthermore, the improvements of the proposed method compared to the cosine similarity based baseline, show the importance of our relation learning while the inference method is fixed.

Average total (potentials computation and inference) runtime versus accuracy plot of different energy minimization methods on different settings is shown in Figure 3.5. Even on this small scale problem, the greedy optimization is faster on average while its accuracy is on par with other inference methods.

Effect of temperature T on unary potential function. In order to evaluate the effectiveness of our proposed unary potential function, we devise the following experiment. In the few-shot common object recognition task with $N = 8$ positive bags, $\bar{B} = 10$ negative images, and $B = 5$, we train the unary potentials with four different settings: (1) SOFTMAX: Unary potential function with the learned T described in Section 3.3, (2) MAX: $T \rightarrow 0$, (3) MEAN: $T \rightarrow +\infty$, and (4) No Unary: the model without using negative bag information. The pairwise potential function is kept

Table 3.2: Results on the miniImageNet dataset using different positive bags N , total number of negative elements \bar{B} , and bag sizes $B = 5$ and $B = 10$. The baseline method uses cosine similarity on the bag elements as relation module*.

N		4		8		16	
\bar{B}		10	20	10	20	10	20
$B = 5$	Ours	63.83 ± 1.49	65.48 ± 1.47	72.49 ± 0.98	73.99 ± 0.96	78.60 ± 0.64	79.93 ± 0.62
	Baseline (cosine)	60.88 ± 1.51	63.83 ± 1.49	64.46 ± 1.05	68.08 ± 1.02	66.78 ± 0.73	70.39 ± 0.77
	MI-SVM [Hoffman et al., 2015]	56.25 ± 1.54	59.03 ± 1.52	62.75 ± 1.06	63.76 ± 1.05	67.91 ± 0.72	73.33 ± 0.69
	sbMIL [Bunescu and Mooney, 2007]	54.55 ± 1.54	59.93 ± 1.52	58.25 ± 1.08	64.68 ± 1.05	61.35 ± 0.75	65.55 ± 0.74
	mi-SVM [Andrews et al., 2003]	54.23 ± 1.54	59.43 ± 1.52	60.43 ± 1.07	66.08 ± 1.04	64.49 ± 0.74	69.69 ± 0.71
	ATNMIL [Ilse et al., 2018]	50.35 ± 1.55	60.33 ± 1.52	56.05 ± 1.09	63.29 ± 1.06	58.97 ± 0.76	67.26 ± 0.73
$B = 10$	Ours	37.42 ± 1.50	38.50 ± 1.51	42.85 ± 1.08	47.63 ± 1.09	51.70 ± 0.77	53.63 ± 0.77
	Baseline (cosine)	35.73 ± 1.49	40.40 ± 1.52	38.01 ± 1.06	43.95 ± 1.09	41.08 ± 0.76	47.83 ± 0.77
	MI-SVM [Hoffman et al., 2015]	29.53 ± 1.41	35.05 ± 1.48	35.25 ± 1.05	39.94 ± 1.07	41.21 ± 0.76	46.63 ± 0.77
	sbMIL [Bunescu and Mooney, 2007]	31.55 ± 1.44	31.50 ± 1.44	34.10 ± 1.04	39.86 ± 1.07	28.80 ± 0.70	43.63 ± 0.77
	mi-SVM [Andrews et al., 2003]	31.55 ± 1.44	35.33 ± 1.48	34.10 ± 1.04	39.86 ± 1.07	39.48 ± 0.76	45.16 ± 0.77
	ATNMIL [Ilse et al., 2018]	26.58 ± 1.37	33.10 ± 1.46	28.48 ± 0.99	35.11 ± 1.05	31.56 ± 0.72	38.14 ± 0.75

* Numbers for baseline methods taken from [Shaban et al., 2019]

Table 3.3: Comparison of our method with other MIL methods (**top**), and graphical model inference methods (**middle**). The effect of unary and pairwise potentials are shown in the bottom part (**bottom**). The common object is found across 8 positive and 8 negative images in these examples*.

Method	COCO	ImageNet
MI-SVM [Hoffman et al., 2015]	60.74 ± 1.07	49.44 ± 1.10
ATNMIL [Ilse et al., 2018]	60.00 ± 1.07	49.35 ± 1.10
Ours	65.34 ± 1.04	55.18 ± 1.09
TRWS [Kolmogorov, 2006]	65.04 ± 1.05	54.20 ± 1.09
AStar [Bergtholdt et al., 2010]	64.99 ± 1.05	54.23 ± 1.09
Unary Only	59.24 ± 1.08	50.29 ± 1.10
Pairwise Only	64.65 ± 1.05	53.00 ± 1.10

* Numbers taken from [Shaban et al., 2019]

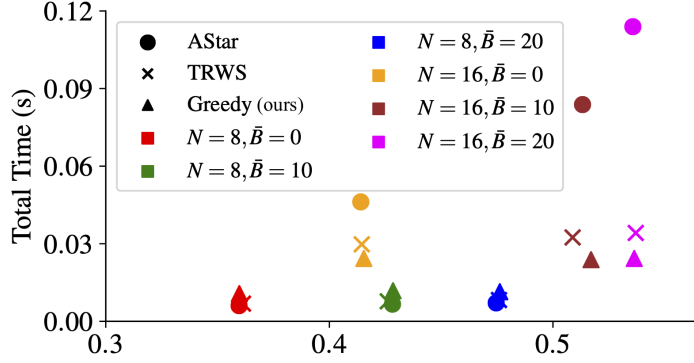


Figure 3.5: Average runtime vs. accuracy of different inference algorithms on miniImageNet for $N \in \{8, 16\}$, $\bar{B} \in \{0, 10, 20\}$, and $B = 10$. Each setting is shown with a distinct color and different inference algorithms are shown with different type of markers.

Table 3.4: Comparison of different unary potential functions on miniImageNet dataset with $N = 8$, $B = 5$ and $\bar{B} = 10$.

Method	No Unary	MEAN	MAX	SOFTMAX
Accuracy (%)	64.48 ± 1.47	70.23 ± 1.00	71.76 ± 0.99	72.49 ± 0.98

identical in all the methods. The performance of our methods on the described settings are presented in Table 3.4. The results show that both MAX and SOFTMAX are performing well while SOFTMAX is getting slightly better results.

3.4.3 Few-shot Object Co-Localization

Datasets. For the few-shot object co-localization task, we perform training on a split of COCO 2017 dataset [Lin et al., 2014] introduced in [Bansal et al., 2018] with 63 training and evaluate on the remaining 17 classes. We also use 148 classes non-overlapping with COCO 2017 training classes from the validation set of ILSVRC2013 detection dataset [Russakovsky et al., 2015] for cross-dataset evaluation. Different statistics of these datasets are shown in Table 3.5.

Table 3.5: Different statistics of the datasets used for the few-shot object co-localization task.

COCO training images	111,085
COCO test images	8,245
ILSVRC test images	12,544
Training classes	63
COCO test classes	17
ILSVRC test classes	148

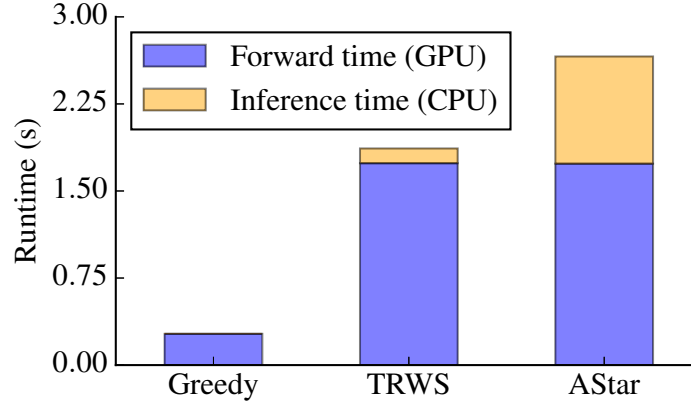


Figure 3.6: Computational time comparison of forward (computing pairwise similarities) and graphical model inference (in sec.) on COCO dataset. Image from [Shaban et al., 2019].

Feature extractor. In the few-shot object co-localization task, each image corresponds to a bag and the bag elements are the proposals generated by running a pre-trained Faster-RCNN detector [Girshick, 2015] on the image. We use ResNet 50 [He et al., 2016] as Faster-RCNN backbone and train it on the COCO 2017 training classes. We save the outputs of the second stage feature extractor for top $B = 300$ proposals according to their objectness scores as bag elements.

Implementation. We make use of the publicly available Faster-RCNN implementation of Tensorflow object-detection API [Huang et al., 2017b] with default settings to extract features on images of size 336×336 .

We map the 2048 dimensional globally averaged output of the second stage feature extractor to a $d = 640$ dimensional feature vector with an additional linear layer to reduce the memory and time complexity of the relation modules. The Faster-RCNN network is trained on four GPUs with batch size of 12 for 600k iterations. All the competitive methods use the same 640 dimensional feature vectors as their bag elements.

Evaluation metric. An element \mathbf{e} for this task corresponds to one bounding box from each image (i.e., positive bag). We use the same success rate as defined in Equation (3.14) for the few-shot object co-localization task. For an element \mathbf{e} in a positive bag in a collection \mathcal{T}_c , we consider its label be $y_c(\mathbf{e}) = 1$ if its corresponding bounding box has IoU overlap greater than 0.5 with a groundtruth bounding box of class c in the corresponding image. It is also called class-agnostic Correct Localization (Cor-Loc [Deselaers et al., 2010]) metric and has been used broadly in weakly supervised object localization literature [Uijlings et al., 2018; Bilen et al., 2015; Cinbis et al., 2016; Shi et al., 2017].

Table 3.6: *Run time and accuracy by varying the bag size and number of positive bags on COCO dataset.*

	N=2	N=8	N=32	N=128
B=100	0.46s	0.63s	1.08s	4.24s
B=200	0.48s	0.66s	1.34s	5.32s
B=300	0.49s	0.71s	1.68s	6.81s
CorLoc	73.01	76.17	76.72	76.84

Results. Quantitative results on 1000 randomly sampled collections with 8 positive bags and 8 negative images (a negative bag with 8×300 elements) on COCO and ILSVRC datasets are shown in Table 3.3. Similar to few-shot common object recognition, our method outperforms other MIL baselines for few-shot object co-localization.

We also present the results when we only use pairwise (“pairwise only”) or unary (“unary only”) potential functions in the energy function defined in Equation (3.4). The results show that using pairwise potentials significantly improves the results. In addition, unary potentials also boost the performance of our algorithm by using the information in the negative bag. However, the performance improvement is relatively less compared to the few-shot common object recognition task in Table 3.2. The reason is that the negative bag can only be helpful if it contains objects similar to the *non-target* objects present in positive bags. This has relatively a low probability given the number of classes we are sampling from and the co-occurrence of objects from different classes in an image.

As presented in Table 3.3, all inference algorithm perform similarly on this task. Our greedy inference algorithm is significantly faster than other inference methods as illustrated in Figure 3.6 and has a slightly better class-agnostic CorLoc. The greedy inference algorithm only requires to compute only 15% of the total pairwise potentials on average for this task. Note that in Figure 3.6, the forward time is the time taken on GPU to compute the potentials given the input features. Inference time is the time the MRF optimization takes on CPU given the computed potentials. The greedy method computes *some of the potentials* and does the optimization in tree nodes on GPU.

Figures 3.7 and 3.8 illustrate qualitative results comparing our approach with other MIL baselines on sampled collections from COCO and ILSVRC datasets respectively.

Effect of bag size and larger N, \bar{B} . To see the effect of increasing the number of bags on COCO dataset, with 128 negative images, we vary positive images from 8 to 128 and B from 100 to 300 (300 is reported to be enough for COCO detection [Huang et al., 2017b]) and report the run time on a single GPU in Table 3.6. We only report the best CorLoc when $B=300$. These results are not comparable to Table 3.3 since we are only using classes with at least 128 images.

Table 3.7: Average energy values for different graphical model inference methods on the *miniImageNet* dataset*.

	N B	4			8			16		
		0	10	20	0	10	20	0	10	20
$B = 5$	TRWS	2.929179	-4.416873	-4.842334	18.300657	-4.425953	-12.602217	86.034355	-6.873013	-10.020649
	ASTAR	2.908970	-4.429455	-4.851543	18.192284	-4.529052	-12.666497	85.560267	-7.277377	-10.398633
	Greedy	2.908970	-4.429455	-4.851543	18.192282	-4.529052	-12.666499	86.692482	-6.909996	-10.002609
$B = 10$	TRWS	0.515563	-6.576048	-8.300273	8.749933	-15.959289	-17.238385	53.324193	-28.602048	-59.609459
	ASTAR	0.502832	-6.597286	-8.315386	8.675015	-16.079914	-17.404502	52.819455	-29.388606	-60.499036
	Greedy	0.502832	-6.597286	-8.315387	8.707342	-16.048676	-17.384832	57.168652	-25.869081	-57.885948

* Numbers taken from [Shaban et al., 2019]

Table 3.8: Average energy values for the few-shot object co-localization task*.

Method	COCO	ImageNet
TRWS	-28.485636	-28.630786
AStar	-28.487422	-28.631678
Greedy	-27.246355	-25.496649

* Numbers taken from [Shaban et al., 2019]

3.4.4 Comparison of Energy Minimization Methods

The average energy value of output selection S for different inference algorithms on *miniImageNet* and COCO datasets are shown in tables 3.7 and 3.8 respectively. The energy values of AStar and TRWS methods are lower for these tasks. However, this does not translate to better success rates implying that finding the approximate solution with our greedy inference method is enough for obtaining high success rate for the tasks while being considerably more efficient.

3.5 Summary

This chapter presents an energy minimization approach for few-shot common object recognition and few-shot object co-localization tasks. The energy minimization problem utilizes unary and pairwise potential functions learned via differentiable pairwise comparators known as relation modules. The pairwise potential function computes a similarity measure between two elements. On the other hand, the unary potential function compares an element with a set of elements accomplished by applying an aggregation function on pairwise comparisons. In contrast with existing inference methods, the introduced greedy inference method exploits the structure of the problem to minimize the energy without necessarily computing all of the pairwise potential functions. The inference method's performance is comparable to other well-established graph inference algorithms for these tasks while being computationally more efficient. Finally, extensive experiments validate the effectiveness of different aspects of the approach.

Few-shot Weakly-Supervised Object Detection via Directional Statistics

Detecting novel objects from few examples has become an emerging topic in computer vision recently. However, these methods need fully annotated training images to learn new object categories which limits their applicability in real world scenarios such as field robotics. In this chapter, we propose a probabilistic multiple-instance learning approach for few-shot Common Object Localization (COL) and few-shot Weakly Supervised Object Detection (WSOD). In these tasks, only image-level labels, which are much cheaper to acquire, are available. We find that operating on features extracted from the last layer of a pre-trained Faster-RCNN is more effective compared to previous episodic learning based few-shot COL methods. Our model simultaneously learns the distribution of the novel objects and localizes them via expectation-maximization steps. As a probabilistic model, we employ von Mises-Fisher (vMF) distribution which captures the semantic information better than Gaussian distribution when applied to the pre-trained embedding space. When the novel objects are localized, we utilize them to learn a linear appearance model to detect novel classes in new images. Our extensive experiments show that the proposed method, despite being simple, outperforms strong baselines in few-shot COL and WSOD, as well as large-scale WSOD tasks.

4.1 Introduction

In this chapter we address the problem of N -way, K -shot Weakly Supervised Object Detection (WSOD), and develop a method with the following capabilities.

Suppose that we are given a set of $N \times K$ previously unseen images consisting of K images of objects from each of N previously unknown (novel) classes. These will be called the “support images.” Each training image has image-level labels, indicating which classes are present in the image. Typically, the number of novel classes N may be up to 20 and the number of training images K from each class may be 5 or 10, but there is no requirement that the number of images in each novel class are equal.

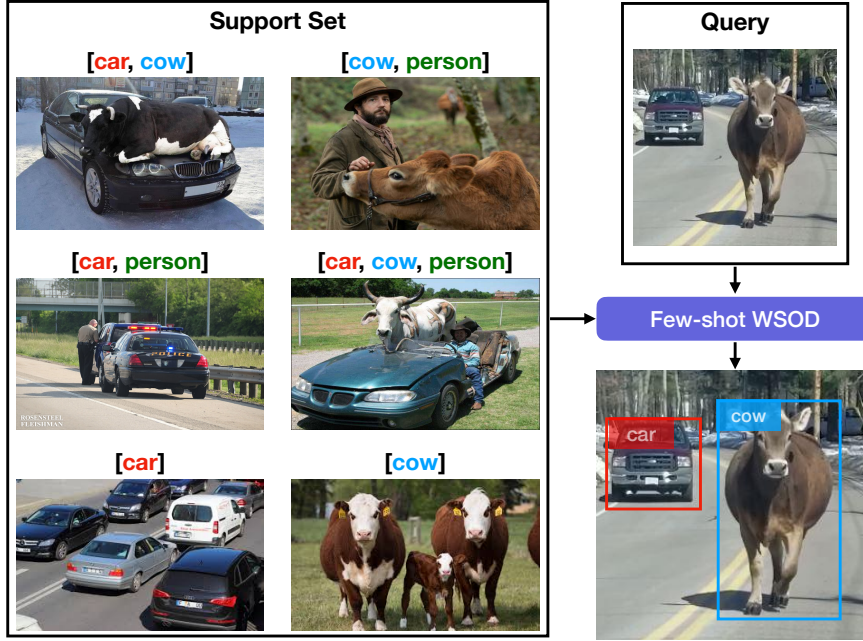


Figure 4.1: *Few-shot WSOD problem. Similar to the few-shot classification problem, the input training set (support set) only contains image labels (car, cow and person are novel classes in this example). The model learns to detect the target objects in the test (query) image. Few-shot WSOD bridges few-shot classification and object detection by learning to detect the novel objects in the query images while only needs image-level labels for the support images.*

Given this small number of support images, the algorithm learns to find instances of (possibly multiple) objects from any of the novel classes in a query image, and will put a bounding box around all such positive instances. As summarized in Figure 4.1, our system provides a flexible object detection algorithm that requires a very small training set of images of novel objects, where each image is annotated only with image-level labels. As such, it is suitable for classifying and detecting objects given only the images provided, for instance, by an internet image search for images of novel classes. In comparison to supervised few-shot object detection approaches, e.g., [Xiao and Marlet, 2020; Wang et al., 2019, 2020; Perez-Rua et al., 2020], where manually labeled bounding box annotations are required, this is a more realistic setting to learn an object detector on novel examples with applications like robotics [Kim et al., 2020] or video object segmentation [Lu et al., 2019].

We first use a Faster-RCNN network to produce bounding box proposals of the possible regions containing an object with their associated feature vectors. This network is pre-trained on a fully annotated *base dataset* with bounding boxes of objects of various classes; the base dataset does not contain any of the novel training classes. Then, the novel objects are learned in a two steps process: 1) A *common object localization* (COL) module is used first to localize the novel objects in the support images. 2) An object detection module to learn the novel object classes found in the COL step.

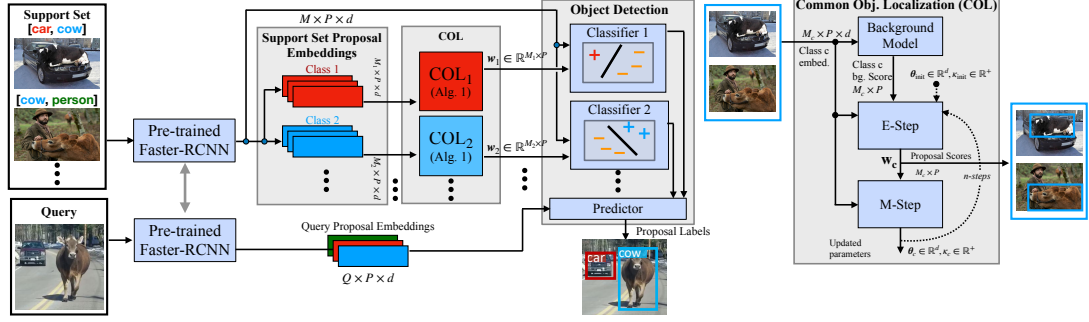


Figure 4.2: The feature maps are shown as the shape of their tensors. Q , M , and C denote the number of queries, support images, and classes respectively. A pre-trained Faster-RCNN (shown in Figure 4.3) on the base dataset is used to extract P proposals from each input image. The embeddings are grouped based on their corresponding image-level labels and each group is fed into a separate Common-Object Localization (COL) module. **COL** module (shown in detail on the right) receives proposal embeddings of images of a class (M_c is the number of images within class c) and simultaneously estimates the common class direction θ_c and concentration κ_c along with bounding-box level labels \mathbf{w}_c via EM steps. **The Object Detection** module uses the top labels of \mathbf{w}_c to learn an appearance model for each novel class in the support set. This appearance model is then tested on the testing proposals to detect novel objects in the query set.

We explain each of these modules in more details below.

COL module. To localize the novel objects in the support images, COL module finds the common object in the K images provided for each of the N novel classes. The COL module is run separately on the images from each of the N novel classes. The input to the common object detector is the set of normalized feature vectors from the images of a novel class c . This set of normalized feature vectors corresponds to the bounding boxes provided by the proposal network¹. An EM algorithm on these feature vectors determines the direction parameter of von Mises-Fisher (vMF) distribution on the unit sphere that is most likely to favour a common object representative from each image. The closest feature from each image identifies the bounding box containing the common object. A distribution for a background class is also trained, using the base dataset to steer the COL away from selecting background objects.

Detection module. Once the novel objects are found by the COL module, their bounding boxes are used to train a box classifier for each novel class c . The classification is done by a 2-class (contains / does not contain the object) classification algorithm, once again working on normalized feature vectors. These bounding boxes (and their associated features) are labeled as either positive or negative for containing

¹In our preliminary experiments, we observed no severe drift in the recall measure of the RPN when being tested on the novel classes compared to the recall value on the base classes. At the same time, there is a significant difference in the final CorLoc values. Hence, we hypothesize that freezing the RPN does not severely affect the performance. We note that finetuning RPN on the support set for the task of few-shot object detection (FSOD). It might be reasonable as the support set is fully annotated. However, finetuning RPN using predicted labels by our COL method is just a chicken-and-egg problem and could cause a drift in the RPN predictions.

the object of the novel class. The positive bounding boxes are those that are determined by the COL module to contain the common object from class c ; the negative samples are chosen from proposals selected from the images of the other classes. Thus, the classifier for class c is trained to distinguish features corresponding to bounding boxes containing an object of class c from those that do not.

Finally, at test time, a query image is passed through the proposal network to provide bounding boxes (and their features). These bounding boxes are then evaluated by each of the classifiers to determine whether they contain a novel class object or belong to the background.

The proposed method is summarized in Figure 4.2. We make several contributions and important observations: 1) We propose a simple yet powerful COL that uses directional statistics for modeling. Our COL module can be built on top of off-the-shelf pre-trained Faster-RCNN models without extra parameters. We observe that by using feature vector directions in our probabilistic model, we can better capture the semantic information compared to the Gaussian distribution. To our knowledge, employing directional statistics for multiple-instance learning is new. 2) We employ a detection module to extend COL to few-shot WSOD. To the best of our knowledge few-shot WSOD has not been studied in the literature before. 3) Despite its simplicity, our method outperforms sophisticated few-shot COL algorithms [Shaban et al., 2019; Hu et al., 2019] on PASCAL VOC [Everingham et al., 2007], MS COCO [Lin et al., 2014], and ILSVRC detection [Deng et al., 2009] benchmarks. In WSOD, our method outperforms recent knowledge-transfer based approaches [Uijlings et al., 2018; Hoffman et al., 2016] in both few-shot and large-scale settings.

4.2 Details of Methodology

4.2.1 Few-Shot WSOD and COL Tasks Definition

The goal in few-shot WSOD is to learn a model that, given support images $\mathcal{D}_{\text{train}}$ containing a set of novel classes \mathcal{L} , detects instances of the novel classes in query images $\mathcal{D}_{\text{test}}$. The support set consists of image-label pairs $(\mathbf{I}, \mathbf{y}) \in \mathcal{D}_{\text{train}}$ where image-level label $\mathbf{y} \subseteq \mathcal{L}$ is a subset of classes present in the image \mathbf{I}^2 . The support set is typically a small K -shot, N -way set sampled from a large dataset $\mathcal{D}_{\text{novel}}$ with a variety of novel classes $\mathcal{C}_{\text{novel}}$. The sampling process for few-shot WSOD follows rules that are similar to few-shot classification problems [Lee et al., 2019; Snell et al., 2017]. A set of N classes $\mathcal{L} \subset \mathcal{C}_{\text{novel}}$, called target classes, are first sampled. Then, for each target class $c \in \mathcal{L}$, K images containing at least an instance of class c are sampled *without replacement* to create the support set $\mathcal{D}_{\text{train}}$. The query set $\mathcal{D}_{\text{test}}$ is sampled similarly, but unlike the support set, query labels also contain bounding box annotations in addition to the image-level labels, as the goal is to detect target objects in the query data. These bounding box annotations are only used for evaluation. Few-shot COL [Shaban et al., 2019; Hu et al., 2019] is a special case of few-shot

²In contrast to few-shot image classification, few-shot WSOD images can have multiple labels.

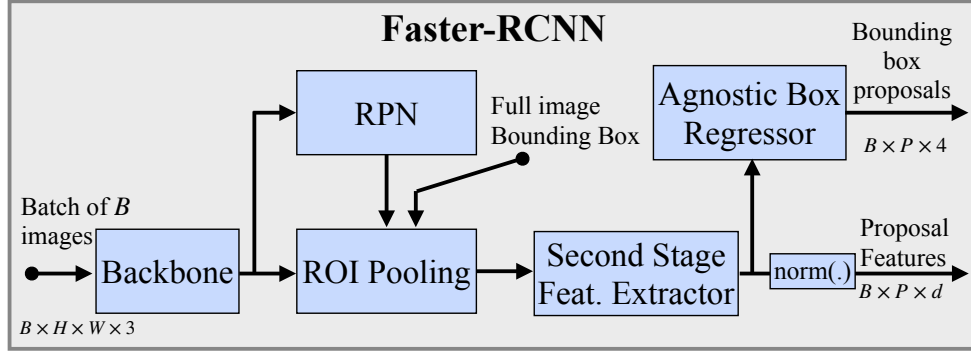


Figure 4.3: *Feature Extraction.* We use a pre-trained Faster-RCNN on the base dataset to extract P proposals from each input image. A ℓ_2 normalization layer is employed to project all the features onto the unit hypersphere.

WSOD where there is only one target class in the support set, i.e., $N = 1$.

For pre-training, the algorithm has access to a large dataset $\mathcal{D}_{\text{base}}$ with a set of base classes $\mathcal{C}_{\text{base}}$. Typically, there is no image in common between the base and novel datasets. Moreover, the set of base classes is disjoint from the set of novel classes used in evaluation, i.e., $\mathcal{C}_{\text{base}} \cap \mathcal{C}_{\text{novel}} = \emptyset$.

4.2.2 Pre-training and Feature Extraction

We pre-train a Faster-RCNN [Ren et al., 2015] on the base dataset for bounding box and feature extraction. The overall architecture is shown in Figure 4.3. To train the network, we use the original bounding box labels within the base dataset to define the Region Proposal Network (RPN) and second-stage losses of the Faster-RCNN. We adapt a class-agnostic bounding box regression model in the second-stage to get one bounding box per feature proposal regardless of the number of base classes. Once trained, we use the trained Faster-RCNN to extract P bounding box proposals $B \in \mathbb{R}^{P \times 4}$ and their corresponding d -dimensional features $F \in \mathbb{R}^{P \times d}$ from each input image I . We also apply an ℓ_2 normalization layer to project all the features to the unit hypersphere. As discussed later, the normalization step is important as our model uses a cosine similarity measure for better generalization.

We need the feature extracted from the full image bounding box to initialize our COL method. This is accomplished by adding the full image bounding box to the box proposals generated by the RPN, thus its feature is extracted by the Faster-RCNN second-stage feature extractor. We denote the first proposal in B and F , the complete image bounding box and its feature, respectively.

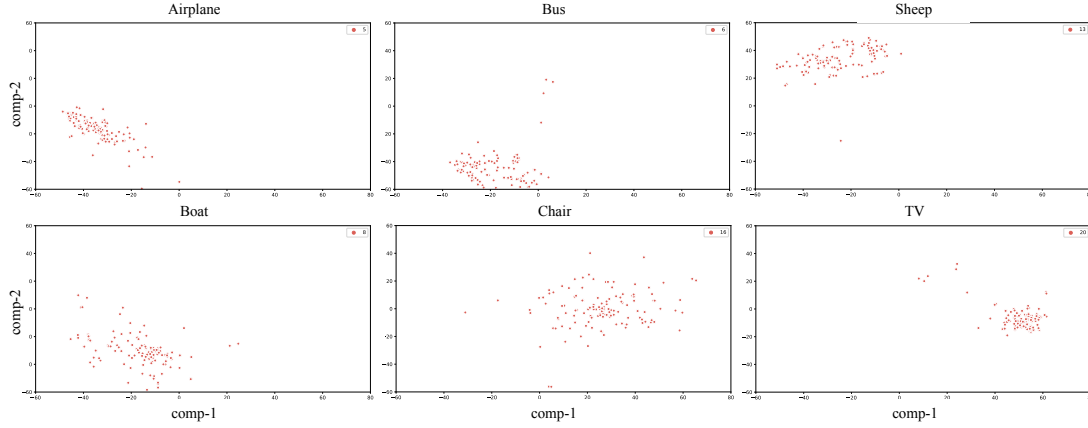


Figure 4.4: Two-dimensional T-SNE projection of proposal features extracted from a pre-trained Faster-RCNN. The proposals are selected using IoU threshold of 0.6 with ground-truth boxes. The instances for each class (approximately) form a single cluster.

4.2.3 Statistical Model Assumptions

Since the support set $\mathcal{D}_{\text{train}}$ provided to the learner is limited, it is crucial to employ proper learning biases in the model to combat overfitting. Inspired by the success of prototypical networks [Snell et al., 2017], we design our model based on the assumption that features of each object class form a single cluster in the embedding space. To illustrate this, we present T-SNE [van der Maaten and Hinton, 2008] plots of the normalized features extracted from proposals of a pre-trained Faster-RCNN corresponding to novel classes in Figure 4.4. The figure shows that the classes are mostly forming a univariate distribution in the 2d T-SNE projection space. We propose to use directional data based on the von Mises-Fisher (vMF) distribution, which arises naturally when each cluster is distributed on the unit hypersphere. Formally, we assume features of each foreground class follow vMF distribution with mean direction θ and positive concentration parameter κ

$$p_{\theta}^{+}(\mathbf{x}) = \frac{1}{Z} \exp(\kappa \theta^{\top} \mathbf{x}) \text{ s.t. } \|\theta\| = 1, \quad (4.1)$$

where Z is the normalizing constant and input $\mathbf{x} \in \mathbb{R}^d$ is a unit vector, i.e., $\|\mathbf{x}\| = 1$ or equivalently $\mathbf{x} \in \mathbb{S}^{d-1}$. We assume the concentration hyperparameter is constant and the same for all novel classes. In Section 4.2.4, we propose an expectation maximization algorithm to estimate the mean direction of a novel class from the support set.

We could also use Gaussian distribution for our model which has an analogous effect to using Euclidean distance. We empirically show that vMF provides superior results to Gaussian distribution when using pre-trained features. Our results support related works in supervised few-shot learning [Qi et al., 2018; Gidaris and Komodakis, 2018] where using the cosine similarity outperforms the Euclidean dis-

tance measure. The underlying reason for this is well-studied by [Wang et al., 2017]; Softmax loss used in the pre-training tends to create a ‘radial’ feature distribution where direction specifies the semantic classes while magnitude decides the classification confidence.

Additionally, a background class distribution is learned to steer the learner toward objects and away from background proposals. Let

$$p_{\omega}^{-}(\mathbf{x}) = \frac{1}{U} u_{\omega}^{-}(\mathbf{x}) , \quad (4.2)$$

represent the background class distribution where U is a constant normalizer. As the base dataset provides a reach set of examples for learning the background model, the background distribution is learned from the base dataset and remains fixed when evaluating on WSOD examples sampled from the novel data.

To learn the background distribution, we collect a set of background proposals with low intersection-over-union (IoU) score (< 0.3) to the objects within the base dataset and use maximum likelihood estimation in [Banerjee et al., 2005] to find the parameters of vMF distribution for the background data.

4.2.4 COL

We first explain the method for few-shot COL with a single novel common object within the support set and employ it for few-shot WSOD later. As shown in Figure 4.5, COL module’s goal is to find the common object representation across a set of images with one novel object in common. Let $\mathcal{F} = \{F_i\}_{i=1}^M$ denote the Faster-RCNN feature proposals extracted from the input images where M is number of images. Each proposal has a (latent) binary label that indicates whether the proposal tightly encloses the common object. Namely, $\mathbf{z}_{ij} \in \{0, 1\}$ is the label of the j -th proposal in the i -th image. Starting from an initial guess for the direction parameter θ of the novel common class, the algorithm alternately refines the mean direction and label estimations in an expectation-maximization optimization framework. We present the update rules here and defer the derivations that bring interesting insights into the proposed method to Section 4.2.4.1. In the E-step, the algorithm uses the current direction to estimate soft labels \mathbf{w} , where $\mathbf{w}_{ik} \in [0, 1]$ is the soft label for the k -th proposal within the i -th image, via attention over the proposals within each image

$$\mathbf{w}_{ik} = \frac{p_{\theta}^{+}(F_{ik}) / p_{\omega}^{-}(F_{ik})}{\sum_{j=1}^P p_{\theta}^{+}(F_{ij}) / p_{\omega}^{-}(F_{ij})} , \quad (4.3)$$

where $F_{ij} \in \mathbb{S}^{d-1}$ is the feature of the j -th proposal in F_i . Recall that p_{θ}^{+} and p_{ω}^{-} are our foreground and learned background distributions introduced in Section 4.2.3. Note that it is unnecessary to know the normalization factors Z and U , since they cancel. In this step, the proposal with a high foreground to background score ratio gets the highest label value within each image.

In the M-step, the mean direction θ and the concentration κ are updated given

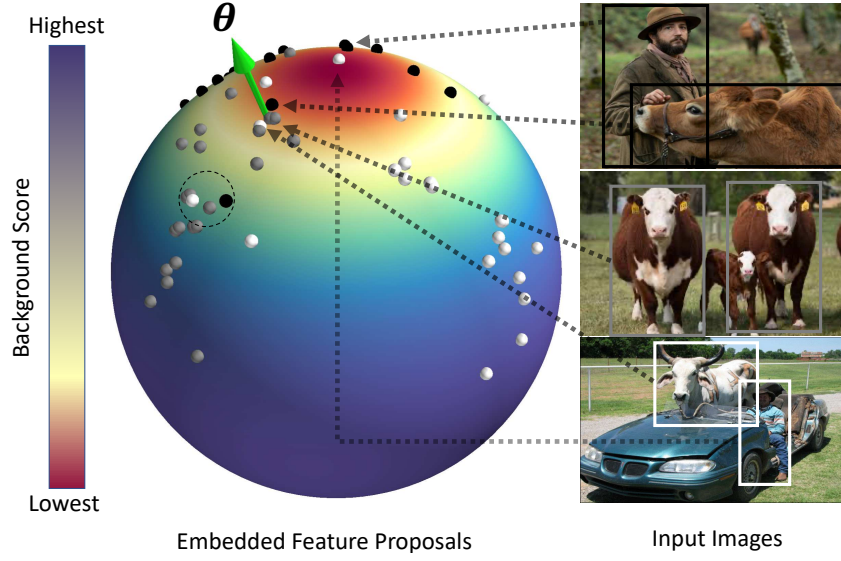


Figure 4.5: Example of COL across three images. Data points on the unit sphere represent feature proposals extracted from all input images. Features extracted from each image are colored the same (shown in white, gray, black colors). Background score function $u_{\omega}^{-}(\mathbf{x})$ is also shown on the unit sphere where blue and red indicate the highest and lowest background scores, respectively. The COL unit’s goal is to find a common object representation θ (shown by green arrow) which is close to at least a white, gray, and black data point. Note that the area marked with dashed circle is also close to proposals from all three images but direction θ is favored as it has a lower background score.

the new labels

$$\begin{aligned} \theta &\leftarrow \frac{\mathbf{r}}{\|\mathbf{r}\|}, \kappa \leftarrow d\|\mathbf{r}\| \\ \text{where } \mathbf{r} &= \frac{1}{M} \sum_{i=1}^M \mathbf{w}_i^{\top} F_i = \frac{1}{M} \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} F_{ik}, \end{aligned} \quad (4.4)$$

where d is the feature dimension. Note that as we only need to know the multiplication of κ and θ , the update rule simplifies to $\kappa\theta \leftarrow d\mathbf{r}$. Intuitively, one can see $\tilde{\mathbf{x}}_i = \mathbf{w}_i^{\top} F_i$ as the common object representation within the i -th image; $\tilde{\mathbf{x}}_i$ is estimated by computing the weighted average over all the proposals where the contribution of proposals are controlled by their soft labels. Given $\tilde{\mathbf{x}}_i$, the novel class direction θ is estimated as the mean of the common object representations similar to the prototypical networks [Snell et al., 2017]. Finally, the estimated mean is projected back onto the unit hypersphere. Updating κ is more involved and is numerically difficult where d and κ are large. We propose several approximations in Section 4.2.4.2 and show that the approximation in Equation (4.4) achieves the best performance in practice.

Algorithm 4 summarizes our COL method. The problem is solved in an iterative fashion by alternating between E-step in Equation (4.3) and M-step in Equation (4.4) until convergence. Following the common practice in WSOD [Rahimi et al., 2020a;

Algorithm 4: Common Object Localization

Input: $\mathcal{F} = \{F_1, \dots, F_M\}, u_{\omega}^-$
Output: Common class mean direction θ and concentration κ

```

 $\kappa\theta \leftarrow \frac{d}{M} \sum_{i=1}^M F_{i1}$  // Initialization
for  $t \leftarrow 1$  to  $T$  do // Iterations
    for  $i \leftarrow 1$  to  $M$  do // E-step
         $\mathbf{o}_{ij} \leftarrow \kappa\theta^\top F_{ij} - \log u_{\omega}^-(F_{ij}) \quad \forall j \in [1, P]$ 
         $\mathbf{w}_i \leftarrow \text{softmax}(\mathbf{o}_i)$  // Update Soft labels
     $\mathbf{r} \leftarrow \frac{1}{M} \sum_{i=1}^M \mathbf{w}_i^\top F_i$ 
     $\kappa\theta \leftarrow d\mathbf{r}$  // M-step

```

[Uijlings et al., 2018; Nguyen et al., 2009], we use the bounding box feature extracted from the complete support images to initialize θ . Recall that we use the first proposal in F_i to represent the complete image feature. Thus, the initialization step can be written as

$$(\kappa\theta)_{\text{init}} \leftarrow \frac{d}{M} \sum_{i=1}^M F_{i1}. \quad (4.5)$$

We remark that our initial direction is similar to what is used as class mean in prototypical networks [Snell et al., 2017]. What makes us different is EM steps that refine the estimated mean by focusing on the common objects and discarding background parts of the image.

4.2.4.1 Expectation-Maximization Derivation

Recall that each proposal has a (latent) binary label $\mathbf{z}_{ij} \in \{0, 1\}$ that indicates whether the proposal tightly encloses the common object. Following the best practice of the previous works in WSOD [Rahimi et al., 2020a; Uijlings et al., 2018; Gokberk Cinbis et al., 2014], we assume there is exactly one proposal with label 1 (positive proposal) in each image and the rest are negative proposals, i.e., $\mathbf{z}_i \in \{\mathbf{e}_1, \dots, \mathbf{e}_P\}$ where $\mathbf{e}_j \in \{0, 1\}^P$ is the j -th canonical basis.

Assuming that the images are sampled independently given the common class c , the full likelihood function is given by

$$\begin{aligned}
 l(\theta; \mathcal{F}) &= p(\mathcal{F}|\theta) = \prod_{i=1}^M p(F_i|\theta) \\
 &= \prod_{i=1}^M \sum_{j=1}^P p(F_i, \mathbf{z}_i = \mathbf{e}_j|\theta),
 \end{aligned} \quad (4.6)$$

where $\theta \in \mathbb{S}^{d-1}$ is the mean direction of the common class distribution. Note that the last equation integrates over all possible values of \mathbf{z}_i . Assuming that proposals

are i.i.d samples from their corresponding distributions given their labels \mathbf{z}_i ³, we can write

$$p(F_i|\mathbf{z}_i = \mathbf{e}_j, \boldsymbol{\theta}) = p_{\boldsymbol{\theta}}^+(F_{ij}) \prod_{\substack{k=1 \\ k \neq j}}^P p_{\omega}^-(F_{ik}) , \quad (4.7)$$

where F_{ij} is the feature of the j -th proposal in F_i , $p_{\boldsymbol{\theta}}^+$ is the generic distribution that generates the common class proposals, and p_{ω}^- represents background proposals' distribution. For brevity, let us re-write Equation (4.7) in a more compact form

$$p(F_i|\mathbf{z}_i = \mathbf{e}_j, \boldsymbol{\theta}) = q_{\boldsymbol{\theta}}(F_{ij}) p_{\omega}^-(F_i) , \quad (4.8)$$

where $q_{\boldsymbol{\theta}}$ is quotient of object and background distributions $q_{\boldsymbol{\theta}}(\mathbf{x}) = p_{\boldsymbol{\theta}}^+(\mathbf{x})/p_{\omega}^-(\mathbf{x})$, and $p_{\omega}^-(F_i) = \prod_{j=1}^P p_{\omega}^-(F_{ij})$.

We adopt the EM algorithm to maximize the likelihood in Equation (4.6) by iteratively optimizing the surrogate expected log-likelihood which is easier to compute. In the E-step, the posterior distribution of the latent variables $\mathbf{w}_{ik} = p(\mathbf{z}_i = \mathbf{e}_k|F_i, \boldsymbol{\theta})$ are computed for the current $\boldsymbol{\theta}$. By Equation (4.8), using Bayes' theorem, and assuming a uniform distribution over image labels \mathbf{z}_i , the posterior can be expressed in terms of quotient of distributions defined above

$$\mathbf{w}_{ik} = \frac{q_{\boldsymbol{\theta}}(F_{ik})}{\sum_{j=1}^P q_{\boldsymbol{\theta}}(F_{ij})} , \quad (4.9)$$

yielding soft label vector $\mathbf{w}_i \in \mathbb{R}^P$ for the i -th image proposals.

By plugging in the vMF probability density function of the common class and background probability density function, the quotient $q_{\boldsymbol{\theta}}$ can be written as

$$q_{\boldsymbol{\theta}}(\mathbf{x}) \propto \exp \left(\kappa \boldsymbol{\theta}^\top \mathbf{x} - \log u_{\omega}^-(\mathbf{x}) \right) . \quad (4.10)$$

As shown in Algorithm 4, one can compute the soft labels via the softmax operation, resembling the attention mechanism recently used for MIL [Ilse et al., 2018].

In the M-step, parameters $\boldsymbol{\theta}$ are updated by maximizing the surrogate expected log-likelihood using the posteriors computed in the E-step

$$l(\boldsymbol{\theta}'; \boldsymbol{\theta}) = \sum_{i=1}^M \mathbb{E}_{p(\mathbf{z}_i|F_i, \boldsymbol{\theta})} \left[\log p(F_i|\mathbf{z}_i, \boldsymbol{\theta}') \right] = \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} \log p(F_i|\mathbf{z}_i = \mathbf{e}_k, \boldsymbol{\theta}') , \quad (4.11)$$

where the weights \mathbf{w}_{ik} are computed in Equation (4.3). Lagrangian function is written as

$$\mathcal{L}(\boldsymbol{\theta}', \lambda) = l(\boldsymbol{\theta}'; \boldsymbol{\theta}) - \lambda(\|\boldsymbol{\theta}'\|^2 - 1) \quad (4.12)$$

By plugging in the log-likelihood term in Equation (4.12) and computing the deriva-

³i.i.d assumption is a standard approach in MIL and works well in practice. See [Maron and Lozano-Pérez, 1998] for more details.

tive w.r.t. θ' and λ we have

$$\begin{aligned}\nabla_{\theta'} \mathcal{L}(\theta', \lambda) &= \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} \nabla_{\theta'} \log \left(p_{\theta'}^+(F_{ij}) \prod_{\substack{k=1 \\ k \neq j}}^P p_{\omega}^-(F_{ik}) \right) - 2\lambda \theta' = \kappa \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} F_{ik} - 2\lambda \theta' . \\ \nabla_{\lambda} \mathcal{L}(\theta', \lambda) &= -\|\theta'\|^2 + 1.\end{aligned}\tag{4.13}$$

Finally, closed-form update rule

$$\begin{aligned}\theta &\leftarrow \text{norm} \left(\sum_{i=1}^M \tilde{\mathbf{x}}_i \right), \\ \text{where } \tilde{\mathbf{x}}_i &= \mathbf{w}_i^\top F_i = \sum_{k=1}^P \mathbf{w}_{ik} F_{ik},\end{aligned}\tag{4.14}$$

is derived by setting the derivatives to zero and solving for θ' and λ .

4.2.4.2 Updating κ in M-Step

In this section, we propose a simple update rule for parameter κ that can be used along Equation (4.14) in the M-step. As shown in our experiments (Table 4.3), updating κ with a our order-0 rule further improves our vMF-MIL COL results.

To find the optimal κ , we compute the derivative of the Lagrangian function in Equation (4.12) w.r.t. κ

$$\begin{aligned}\partial_{\kappa} \mathcal{L}(\theta', \lambda) &= \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} \partial_{\kappa} \log \left(p_{\theta'}^+(F_{ij}) \prod_{\substack{k=1 \\ k \neq j}}^P p_{\omega}^-(F_{ik}) \right) = \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} \partial_{\kappa} \log p_{\theta'}^+(F_{ik}) \\ &= \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} \partial_{\kappa} (-\log Z(\kappa) + \kappa F_{ik}^\top \theta') = -M \frac{\partial_{\kappa} Z(\kappa)}{Z(\kappa)} + \mathbf{r}^\top \theta',\end{aligned}\tag{4.15}$$

where $\mathbf{r} = \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} \mathbf{x}$ and $Z(\kappa)$ is vMF distribution normalization factor. A precise formula is known for $Z(\kappa)$, namely

$$Z(\kappa) = \frac{(2\pi)^{d/2} I_{d/2-1}(\kappa)}{\kappa^{d/2-1}}.\tag{4.16}$$

where d is the feature dimension and I is the modified Bessel function. This formula is quoted in [Banerjee et al., 2005, 2.2], but it is upside-down compared to Equation (4.16), since we are defining $c_d(\kappa) = 1/Z(\kappa)$.

By plugging in $\theta' = \mathbf{r}/\|\mathbf{r}\|$ from Equation (4.14) and setting the derivative to zero we get

$$\frac{\partial_{\kappa} Z(\kappa)}{Z(\kappa)} = \frac{\|\mathbf{r}\|}{M} = \bar{r}.\tag{4.17}$$

Equation (4.17) is similar to what we see in vMF maximum-likelihood estima-

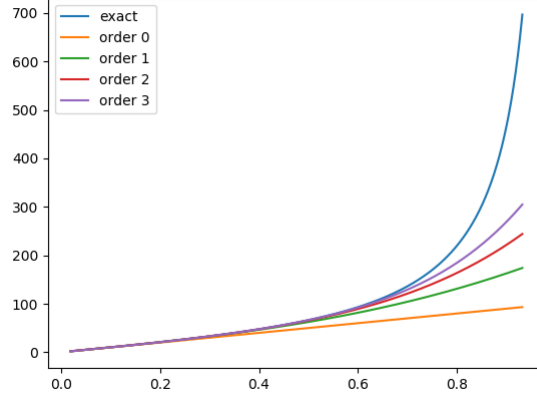


Figure 4.6: Plot of different estimates of $\hat{\kappa}$ as a function of \bar{r} , for dimension $d = 100$. At this resolution, the exact estimate is indistinguishable from the estimate Equation (4.21). The graph also shows approximations of different orders, such as Equation (4.23) and Equation (4.24), which are accurate for small-to-medium values of \bar{r} , but not for larger values. However, the exact value of $\bar{\kappa}$ is extremely sensitive to small variations in the value of \bar{r} , and it diverges to infinity as \bar{r} approaches 1. For this reason it may not be good practice (as is verified by our experiments) to use the exact estimate of $\bar{\kappa}$ in clustering.

tion, therefore, we can use the maximum-likelihood derivations from now on (See Appendix of [Banerjee et al., 2005] equations A.7 to A.8) which leads to maximum-likelihood estimation

$$\kappa = A_d^{-1}(\bar{r}) , \quad (4.18)$$

where A_d is the ratio of Bessel functions,

$$A_d(\kappa) = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)} . \quad (4.19)$$

It is still numerically difficult to compute the Bessel function in cases where d and κ are large. We are able to compute it in python using the `scipy.special.iv` function, only for values of d up to about 120 and κ up to about 700.

To address this difficulty, different formulae are given for estimating the optimal value of $\hat{\kappa}$.

1. A formula due to [Mardia and Jupp, 2009] is given in [Banerjee et al., 2005](4.2) as

$$\hat{\kappa} \approx d\bar{r} \left(1 + \frac{d}{d+2}\bar{r}^2 + \frac{d^2(d+8)}{(d+2)^2(d+4)}\bar{r}^4 \right) .$$

For large values of d , this is almost the same as

$$\hat{\kappa} \approx d\bar{r} \left(1 + \bar{r}^2 + \bar{r}^4 \right) . \quad (4.20)$$

2. [Banerjee et al., 2005] derive an estimate (unnumbered, above [Banerjee et al., 2005, 4.4]),

$$\hat{\kappa} \approx \frac{d\bar{r}}{1 - \bar{r}^2} \quad (4.21)$$

which has series expansion

$$\hat{\kappa} \approx d\bar{r}(1 + \bar{r}^2 + \bar{r}^4 + \dots) . \quad (4.22)$$

Since $0 \leq \bar{r} < 1$, this series will converge, albeit slowly for \bar{r} close to 1. Thus, it is seen that Equation (4.20) is simply a truncated approximation to the infinite series Equation (4.22). We shall refer to Equation (4.20) (perhaps somewhat inexactly) as the “order-2” approximation to Equation (4.22), since the approximation to $1/(1 - \bar{r}^2)$ contains terms up to second order in \bar{r}^2 .

3. It is also possible to consider approximations of other orders for Equation (4.21), including in particular the 0-order approximation

$$\hat{\kappa} \approx d\bar{r} , \quad (4.23)$$

first order approximation

$$\hat{\kappa} \approx d\bar{r} (1 + \bar{r}^2) , \quad (4.24)$$

and the third-order approximation.

4. Another empirically derived formula is also given in [Banerjee et al., 2005, 4.4]. However, we observe (see Figure 4.6) that the approximation Equation (4.21) is already a very close approximation, and the use of [Banerjee et al., 2005, 4.4] is not warranted.

We show graphs of the approximations of $\hat{\kappa}$ for various approximation, and the exact solution in Figure 4.6.

4.2.5 Finding the Common Object in the Query Set

For a single feature proposal $\mathbf{x} \in \mathbb{S}^{d-1}$ extracted from query image \mathbf{I} , our goal is to estimate the class label $c \in \{0, 1\}$ which indicates if the query proposal tightly encloses the target object. Given the estimated common object mean $\boldsymbol{\theta}$ and the background class distribution p_{ω}^- , we compute conditional class distribution function $P(c|\mathbf{x}) \propto P(c)p(\mathbf{x}|c)$ where $P(c)$ and $p(\mathbf{x}|c)$ are the class prior and likelihood, respectively. We assume that the foreground class prior is a constant value α , i.e., $P(c = 1) = \alpha$. Using the background and vMF foreground class likelihoods, the conditional class distribution is written as

$$P(c|\mathbf{x}) \propto \begin{cases} \exp(\kappa \boldsymbol{\theta}^\top \mathbf{x} - \log u_{\omega}^-(\mathbf{x})) & c = 1 \\ \lambda & c = 0 , \end{cases} \quad (4.25)$$

where $\lambda = (1 - \alpha)/\alpha \times Z/U$ encapsulates all the constants and Z is the vMF normalizing constant. Equivalently, proposal \mathbf{x} can be classified via a softmax over the logits

$$\text{logit}(c|\mathbf{x}) = \begin{cases} \kappa \boldsymbol{\theta}^\top \mathbf{x} - \log u_{\omega}^-(\mathbf{x}) & c = 1 \\ \log \lambda & c = 0. \end{cases} \quad (4.26)$$

We set $\lambda = 1$ for all the COL experiments. Changing λ adjusts the confidence values but keeps the order of the final scores the same, therefore, its value does not affect the mean Average Precision (mAP) or Correct Localization (CorLoc) metrics.

4.2.6 WSOD

For the task of WSOD where we have more than one target class, our COL algorithm is first used to label instances of each class. Once the support set is labeled, an off-the-shelf few-shot object detection model can be used for learning novel classes. Inspired by the success of the recent few-shot object detection method in [Wang et al., 2020], we employ a single layer cosine similarity classifier for learning.

Learning is performed on one target class $c \in \mathcal{L}$ at a time. Let $\mathbf{v}_c \in \mathbb{R}^d$ denote the classifier weight for class c . The classification score for this class is computed as

$$s_c(\mathbf{x}) = \frac{\tau \mathbf{v}_c^\top \mathbf{x}}{\|\mathbf{v}_c\|}, \quad (4.27)$$

where $\mathbf{x} \in \mathbb{S}^{d-1}$ is the ℓ_2 normalized feature proposal extracted by our Faster-RCNN model and τ is temperature hyperparameter. For class c , the input training set $\mathcal{D}_{\text{train}}$ is split into positive images $\mathcal{D}_{\text{train}}^c$ of images that have the target class and negative set $\mathcal{D}_{\text{train}} \setminus \mathcal{D}_{\text{train}}^c$, images without the target class. Then, we label $\mathcal{D}_{\text{train}}^c$ by running the COL algorithm on the positive images and select the proposal with the highest soft label from each image as the common object representative. All the proposals in the negative set are used as negative examples. Finally, \mathbf{v}_c is learned by minimizing the sigmoid cross entropy loss over the positive and negative proposals. We use the L-BFGS optimizer with strong Wolfe line search for faster convergence.

At the test time, a test proposal \mathbf{x} from the query set $\mathcal{D}_{\text{test}}$ is scored using the classifiers learned for each novel class.

Recently, [Qi et al., 2018] propose a weight imprinting process to learn novel class prototypes on the unit hypersphere. Learning on the unit hypersphere has been employed by other few-shot learning algorithms for better generalization [Gidaris and Komodakis, 2018] and to stabilize the training [Wang et al., 2020]. Most recently, [Yang et al., 2021] propose to make the distributions more Gaussian by transforming the features of the support set and query set using Tukey’s Ladder of Powers transformation [Tukey, 1977]. It is shown that Tukey’s normalization significantly improves the performance of few-shot prototypical learning. As the scope of these methods is limited to supervised learning, we compare different normalizations and transformations used in the literature for the weakly supervised task of COL in Section 4.3.4.

4.3 Experiments

We evaluate the proposed method in few-shot COL and WSOD problems. We compare our work (vMF-MIL) with Greedy Tree Chapter 3 and SILCO [Hu et al., 2019], two state-of-the-art methods for the task of few-shot common object localization.

To the best of our knowledge there is no WSOD algorithm for few-shot setting in the literature. However, WSOD with knowledge-transfer methods [Rahimi et al., 2020a; Uijlings et al., 2018; Hoffman et al., 2016; Deselaers et al., 2010] are closely related to our work. We describe a slightly modified version of [Uijlings et al., 2018], called MI-SVM in our experiments, in Appendix A.2, and discuss its differences to the proposed method. The MI-SVM baseline is not applicable to the COL problem as it always requires negative examples for training. To compare MI-SVM against other COL methods, we provide MI-SVM with an extra set of K negative images that do not have the target class when sampling the support set.

The original version of Greedy Tree selects only one proposal from each image in the support set and does not perform detection on a new query image. To make it compatible with other methods, we add a simple inference step to the Greedy Tree algorithm. Let $S = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ denote the set of selected proposals, one from each image in the support set. We score feature proposal \mathbf{x} from the query image as a sum of its pairwise similarities to all the selected proposals, i.e., $\text{score}(\mathbf{x}) = \sum_{j=1}^M r(\mathbf{x}, \mathbf{x}_j)$, where r is the learned pairwise similarity function by Greedy Tree. The computed score measures the negative change in the energy value if \mathbf{x} were added as a new node to the graph labeling problem used in Chapter 3.

In all the methods, we first hold out 20 base classes for validation and hyperparameter tuning and then re-train on all the base classes with the best found parameters. For evaluation, we compute the correct localization (CorLoc) rate [Deselaers et al., 2010] and mean Average Precision (mAP) with IoU overlap threshold of 0.5 on the query image.

4.3.1 Common Object Localization

Table 4.1: CorLoc (*top*) and mAP (*bottom*) performance of different few-shot common object localization methods on VOC07 test set. All of the models are trained on COCO60 and evaluated on a test query with $K = 5$ images in the support set. The best and second best performing methods are shown in **bold** and *gray* backgrounds respectively. *MI-SVM receives K extra negative images.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	CorLoc
MI-SVM* [Uijlings et al., 2018]	29.4	13.2	53.7	32.7	12.9	70.4	66.4	67.4	15.7	81.6	10.0	67.6	67.1	27.1	10.1	16.7	84.2	38.9	43.5	41.1	42.5
SILCO [Hu et al., 2019]	51.0	30.3	50.7	34.5	11.3	72.2	63.6	58.9	11.2	86.8	6.7	56.9	51.9	49.2	13.0	16.7	52.6	41.1	46.8	34.2	42.0
Greedy Tree (Chapter 3)	35.3	21.1	59.7	34.5	24.2	77.8	73.4	61.1	23.1	89.5	15.0	64.7	73.4	25.4	12.8	13.3	100.0	64.2	61.3	46.6	48.8
vMF-MIL (ours)	62.7	42.1	53.7	49.1	6.5	68.5	73.8	69.5	19.4	97.4	36.7	65.7	82.3	40.7	21.7	15.0	94.7	64.2	69.4	31.5	53.2
method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
MI-SVM* [Uijlings et al., 2018]	17.7	7.7	31.6	10.6	4.3	46.5	40.1	53.3	3.6	56.8	3.3	56.3	42.3	17.1	1.7	8.1	37.9	25.9	27.3	19.2	25.6
SILCO [Hu et al., 2019]	33.0	13.4	34.5	14.8	3.9	48.8	38.4	55.5	4.0	52.8	4.5	54.2	36.3	27.3	3.3	7.7	27.0	31.3	36.7	23.5	27.5
Greedy Tree (Chapter 3)	26.0	8.7	37.4	11.5	7.5	52.4	47.7	45.8	7.4	61.1	4.5	47.3	50.7	15.6	2.3	5.0	40.0	46.3	47.3	25.7	29.5
vMF-MIL (ours)	36.7	20.6	38.1	14.2	1.9	55.4	50.2	56.5	7.4	71.4	9.5	56.2	63.4	16.8	4.9	3.4	39.3	43.0	51.2	23.0	33.1

Table 4.2: CorLoc(%) and mAP(%) results of different methods for the task of common object localization on novel object classes on the COCO60 dataset with support set size $K = 5$ and $K = 10$. *MI-SVM receives K extra negative images.

Model	K = 5		K=10	
	CorLoc@0.5	mAP@0.5	CorLoc@0.5	mAP@0.5
MI-SVM* [Uijlings et al., 2018]	30.5	15.9	33.2	16.2
SILCO [Hu et al., 2019]	29.7	14.8	31.3	15.8
Greedy Tree (Chapter 3)	32.7	16.0	33.8	16.4
vMF-MIL (ours)	35.7	19.6	38.2	20.2

We use the official implementations of SILCO and Greedy Tree for this experiment. To have a fair comparison with SILCO, we employ Faster-RCNN with a VGG16 [Simonyan and Zisserman, 2015] backbone architecture for feature extraction in both Greedy Tree and our method.

We evaluate on a popular MS COCO 2014 [Lin et al., 2014] split used in few-shot object detection methods [Wang et al., 2020; Perez-Rua et al., 2020; Xiao and Marlet, 2020; Yan et al., 2019; Kang et al., 2019], named COCO60. In the COCO60 split, 60 categories disjoint with the PASCAL VOC dataset are used as base classes and the remaining 20 classes are used as novel classes. This allows us to also perform a cross-dataset evaluation on the PASCAL VOC07 [Everingham et al., 2007] test set. We evaluate the performance of each method over 2000 randomly sampled tasks.

Table 4.1 and Table 4.2 summarize the results on PASCAL VOC and MS COCO datasets, respectively. Despite its simplicity, our method outperforms all the methods by a large margin, followed by Greedy Tree and SILCO. Specifically, we gain between 10% to 20% relative improvement in mAP metric against the second best performing method. The proposed method and Greedy Tree both estimate latent proposal-level labels of the support images to find the common object. However, SILCO explores the dense similarity between each support image and the query image while using coarse image-level features via a global average pooling to estimate the relation of support images. This experiment confirms that estimating proposal-level labels within the support images is quite important for common object localization.

4.3.1.1 Effect of Updating κ in M-Step

As pointed out in the caption to Figure 4.6, the exact estimate of $\hat{\kappa}$ may not be a good choice for clustering, in the case where \bar{r} approaches 1 (meaning that the data has small spread).

In Table 4.3, we try all the different formulas described in Section 4.2.4.2 to estimate a value of $\hat{\kappa}$. We also report the results where κ is kept constant. The experiments show the following outcomes.

1. Order- ∞ in Equation (4.21) performs notably worse, presumably because of the sensitivity to the value of \bar{r} , which is computed from a relatively small number of samples in our few-shot learning scenario.

Table 4.3: *CorLoc(%) and mAP(%) results with κ estimations for the task of COL on novel object classes on the COCO60 dataset with support set size $K = 5$ and $K = 10$.*

$\hat{\kappa}$	K = 5		K=10	
	CorLoc@0.5	mAP@0.5	CorLoc@0.5	mAP@0.5
Constant	34.8	18.6	36.9	20.0
$d\bar{r}(1 + \bar{r}^2 + \bar{r}^4 + \dots)$	20.1	11.3	24.6	13.7
$d\bar{r}(1 + \bar{r}^2 + \bar{r}^4 + \bar{r}^6)$	31.8	17.7	34.7	19.0
$d\bar{r}(1 + \bar{r}^2 + \bar{r}^4)$	33.1	18.6	36.3	19.5
$d\bar{r}(1 + \bar{r}^2)$	34.7	19.0	37.5	19.9
$d\bar{r}$	35.7	19.6	38.2	20.2

2. The order-1 to order-3 estimates perform approximately the same as fixing $\hat{\kappa}$.
3. The order-0 approximation, $\hat{\kappa} = d\bar{r}$, gives the best results. In our COL experiments, $d = 512$, so setting $\hat{\kappa} = d\bar{r}$ places an upper bound of 512 on the value of $\hat{\kappa}$.

4.3.1.2 Direct Comparison to Greedy Tree

To ensure a fair comparison, we also compare our common object localization unit to the Greedy Tree algorithm by exactly following the original experimental protocol in Chapter 3. The Greedy Tree algorithm utilizes a split of the COCO 2017 dataset with 63 base classes for training and 17 held-out novel classes for testing the algorithm. The trained model is also tested on a subset of the ILSVRC 2013 detection dataset with 148 novel classes that have no overlap with the base classes. In Greedy Tree, a Faster-RCNN with ResNet50 [He et al., 2016] backbone is first trained on the base classes and used to extract features from all the images. To allow a fair comparison, we use the same feature set provided by the authors. To mimic common object localization during training, we sample tasks with $N = 1$ and $K = 8$ for training.

Similar to Chapter 3, we evaluate our model over 1000 randomly sampled tasks each containing $K = 8$ images with an object class in common. For each image, the proposal with the highest soft label in Equation (4.3) is returned as the common object. We report the class-agnostic CorLoc ratio on COCO and ILSVRC datasets in Table 4.4 and compare it with the results in Chapter 3. vMF-MIL outperforms Greedy Tree by 2.20% and 1.75% in MS COCO and ILSVRC datasets, respectively.

Table 4.4: *Class-agnostic CorLoc(%) with 95% confidence interval of the method in Chapter 3 compared to our method. All methods use $K = 8$ positive images for finding the common object.*

method	COCO	ILSVRC13
Greedy Tree (Chapter 3)	64.65 ± 1.05	53.00 ± 1.10
vMF-MIL	66.85 ± 1.03	54.75 ± 1.09

4.3.2 Few-shot WSOD

We train our model on COCO60 for the task of few-shot WSOD with different N -way, K -shot problems and compare it with the knowledge-transfer MI-SVM model described in Appendix A.2 on PASCAL VOC 2007 and MS COCO novel classes in Table 4.5. To highlight the importance of EM refinement, we also train our model with full image prototypical initialization without EM refinement. In both datasets, vMF-MIL outperforms MI-SVM in all the scenarios, demonstrating the strong generalization ability of our learning approach.

Table 4.5: $mAP(\%)$ of different few-shot WSOD methods on COCO60 and PASCAL VOC datasets.

Method	Dataset	$N = 5$		$N = 10$		$N = 20$	
		$K = 5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=10$
Prototypical Init	VOC07	16.01	17.93	10.56	11.02	5.41	6.72
MI-SVM [Uijlings et al., 2018]		17.99	20.27	12.09	13.07	7.04	8.32
vMF-MIL (ours)		21.22	22.01	14.54	15.83	8.83	10.19
Prototypical Init	COCO60	8.90	9.28	4.65	6.07	2.99	3.26
MI-SVM [Uijlings et al., 2018]		11.40	11.60	7.30	7.80	2.97	3.70
vMF-MIL (ours)		12.35	13.19	8.53	10.07	4.23	4.85

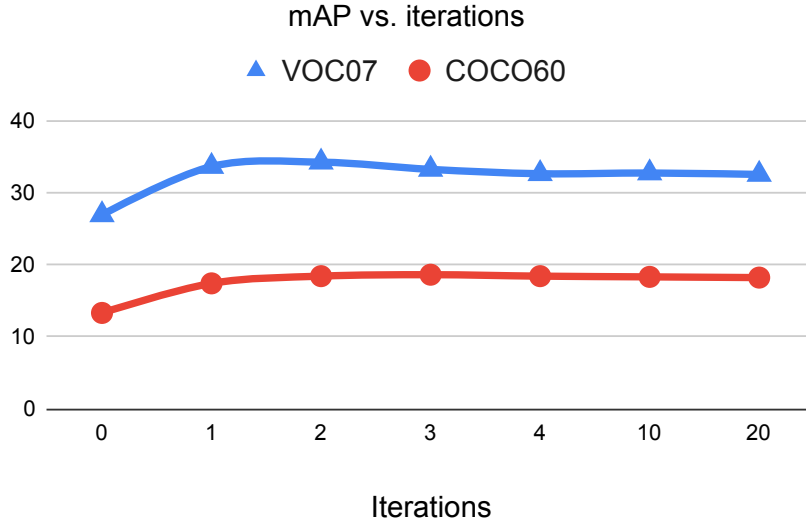


Figure 4.7: $mAP(\%)$ vs. number of EM iterations in common object localization task with $K = 5$ on COCO60 and VOC07 datasets. The performance reaches a plateau at step 4.

4.3.3 Large-Scale WSOD

Although our method is designed for low-shot settings, it is interesting to evaluate its performance in the standard WSOD setting as well. Related to our work is transfer

learning approaches for large-scale WSOD [Uijlings et al., 2018; Hoffman et al., 2016] with ImageNet detection as the standard benchmark. Typically, the first 100 classes are used as the base dataset and the remaining 100 classes with 65k images are used as novel objects. We follow the setup in [Uijlings et al., 2018] and use the pre-trained Inception-Resnet Faster-RCNN model and weights provided by the authors to extract proposals, features, and the objectness scores. We apply our EM algorithm to the extracted features and use $u_{\omega}^{-}(\mathbf{x}) = \alpha(1 - \text{obj}(\mathbf{x}))$ for each proposal \mathbf{x} where $\text{obj}(\mathbf{x})$ is the Faster-RCNN objectness score and α is a hyper-parameter we tuned for the task. To our surprise, vFM-MIL outperforms [Uijlings et al., 2018] in Table 4.6 while being about $100\times$ faster. We believe these results can be further improved by relaxing some of the assumptions in our statistical model as overfitting may not be as significant in large-scale settings. For instance, we can learn a separate concentration parameter for each novel class in the EM steps. Furthermore, we can utilize the novel dataset to update the background scoring function u_{ω}^{-} . We defer these improvements and further analysis to the future work.

Table 4.6: Large-Scale WSOD on ImageNet Detection.

Model	CorLoc@0.5	Time (min.)
LSDA (JMLR 2016) [Hoffman et al., 2016]	28.8	-
Uijlings et al. (CVPR 18) [Uijlings et al., 2018]	74.2	900 (estimated)
vMF-MIL (ours)	76.5	10

4.3.4 Ablation Study

To understand which parts of the proposed method are critical for common object localization, we analyzed results in Table 4.2 with $K = 5$ for each of the important components of the proposed method in Table 4.7. These components are: initializing θ and κ using features extracted from the complete image (Prototypical Init), updating θ , updating κ , and learning background distribution p_{ω}^{-} to steer the algorithm toward objects. The first entry (#1) in Table 4.7 shows that there is a huge performance gap when the background model is not used. This is expected, since without using the background model it may localize non-object patterns such as grass, water, building, etc. with similar appearances as the common object. Comparing the third entry (#3) with #5 and #6 reveals that updating both θ and κ in the EM refinements is important and that increases CorLoc by 4.8% and mAP by 6.3%. The fourth entry shows the importance of initialization; the EM steps are only effective if θ is initialized with the complete image proposal otherwise EM reaches a low quality local minimum.

The second part of Table 4.7 shows the advantage of using vMF to Gaussian distribution in the EM algorithm (see Appendix A.1 for the details). Tukey’s transformation Tukey [1977] further improves the performance of the Gaussian model but vMF distribution still exhibits the best performance. We believe this is because



Figure 4.8: Bounding box adjustments at each iteration for the common object localization experiment on COCO60 with $K = 5$. Only the top prediction in the query image is shown (in pink color) for each iteration. Ground-truth bounding boxes of the target classes are shown in green. EM refinements improve the target object localization in the query image.

feature vectors’ direction better captures the semantic information.

Table 4.7: Ablation study on COCO60 dataset. #1-6 show the importance of initialization, iterative EM updates, and learning the background model. #7-9 compare different statistical models in the EM algorithm.

#	Random Init	Prototypical Init	Update θ	Update κ	p_w^-	CorLoc	mAP
1		✓	✓			1.9	0.6
2					✓	22.8	9.3
3		✓			✓	30.9	13.3
4	✓		✓		✓	30.1	14.2
5		✓	✓		✓	34.8	18.6
6		✓	✓	✓	✓	35.7	19.6
	Gaussian	Tukey+Gaussian	vMF				
7	✓					29.8	13.9
8		✓				34.0	17.3
9			✓			35.7	19.6

Finally, we illustrate the performance improvement vs. the number of EM steps in Figure 4.7. In both VOC07 and COCO60 datasets, mAP reaches a plateau showing that the algorithm converges quickly. Qualitative results in Figure 4.8 depict successful cases where EM refinements improve the top prediction.

4.3.5 Qualitative Results

We show some of the success cases of our method. Our first example in Figure 4.9 shows vMF-MIL performance on a single few-shot WSOD problem. Given the support set with only image-level annotations the algorithm learns to detect the target objects in the query set. We sample 4 query images to evaluate the algorithm performance in detecting different target objects. Except person in the first query image, vMF-MIL successfully detects other target objects. More few-shot WSOD tasks are shown in Figure 4.10, 4.11, 4.12, 4.13 and 4.14.

Finally, Figure 4.15 shows some of the success cases in localizing the target object in the query image for the task of common object localization in Section 4.3.1. All the target objects (dog, car, cow, train, boat, bus, sofa, horse, person) shown in this figure are novel. Also, ground-truth annotations are only shown for better visualization and are not used in learning.

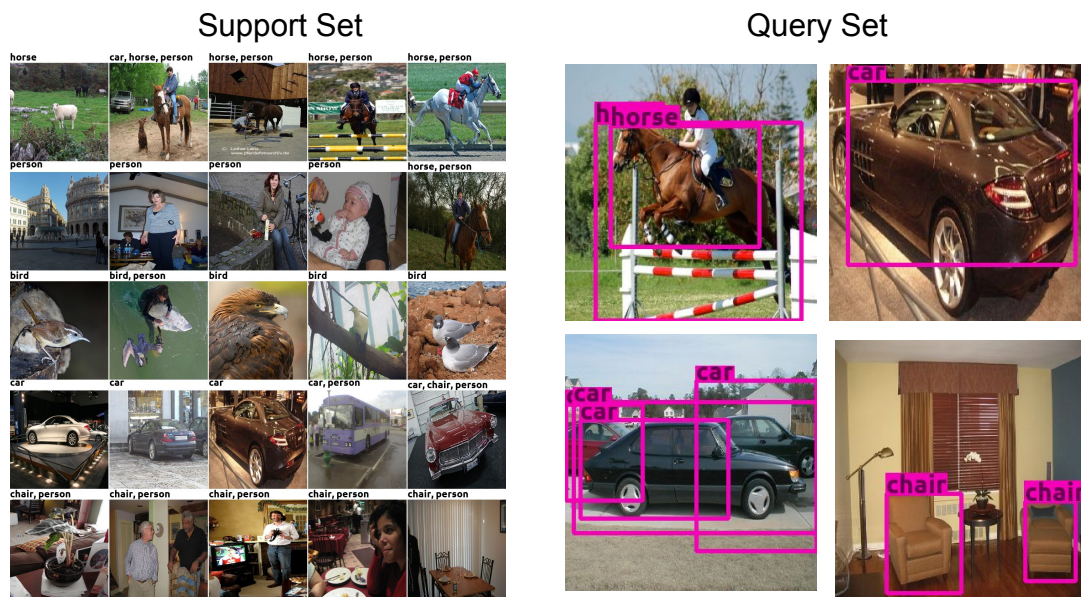


Figure 4.9: Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object on the 4 different query images on the right side. The algorithm fails to detect person in the first query image but successfully detects other target objects.

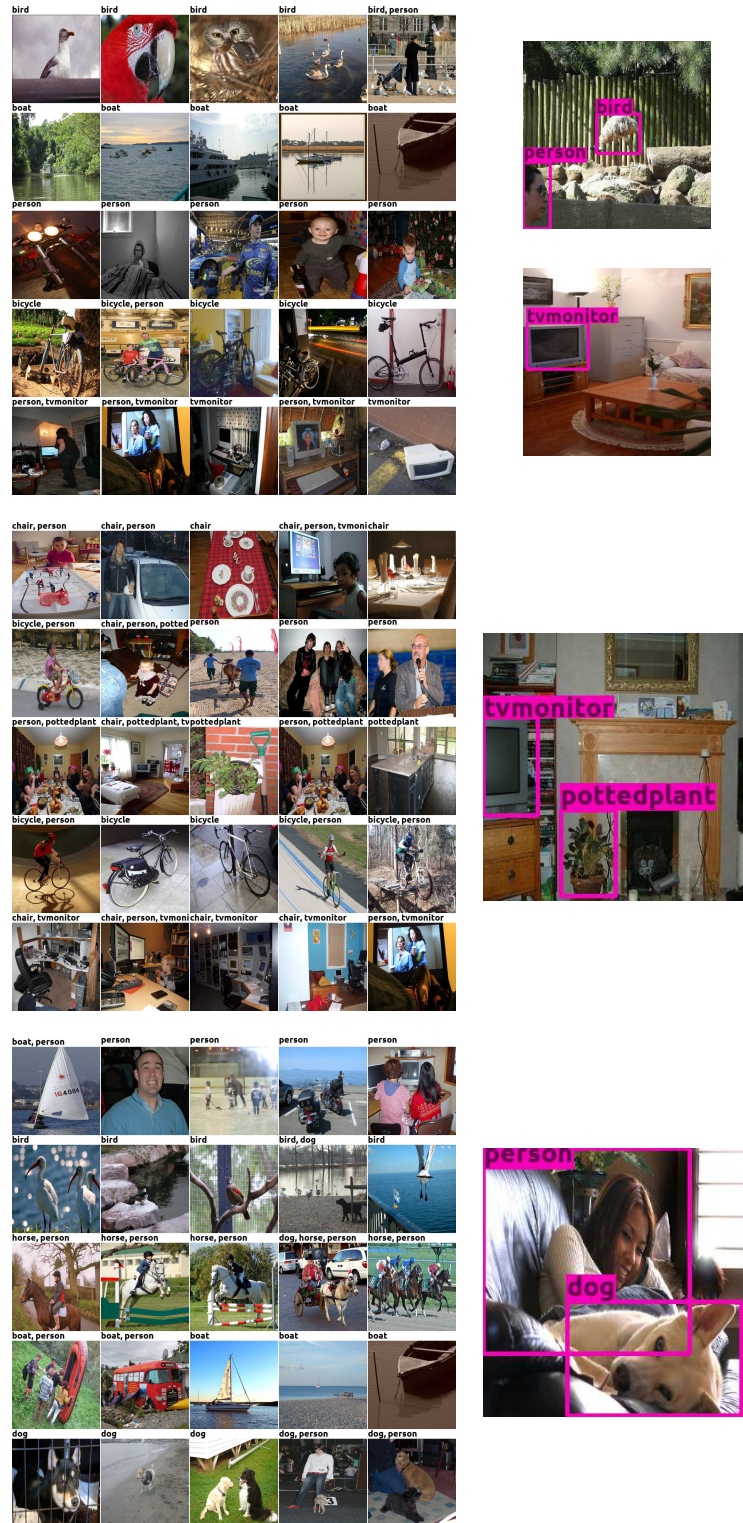


Figure 4.10: *Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.*



Figure 4.11: *Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.*

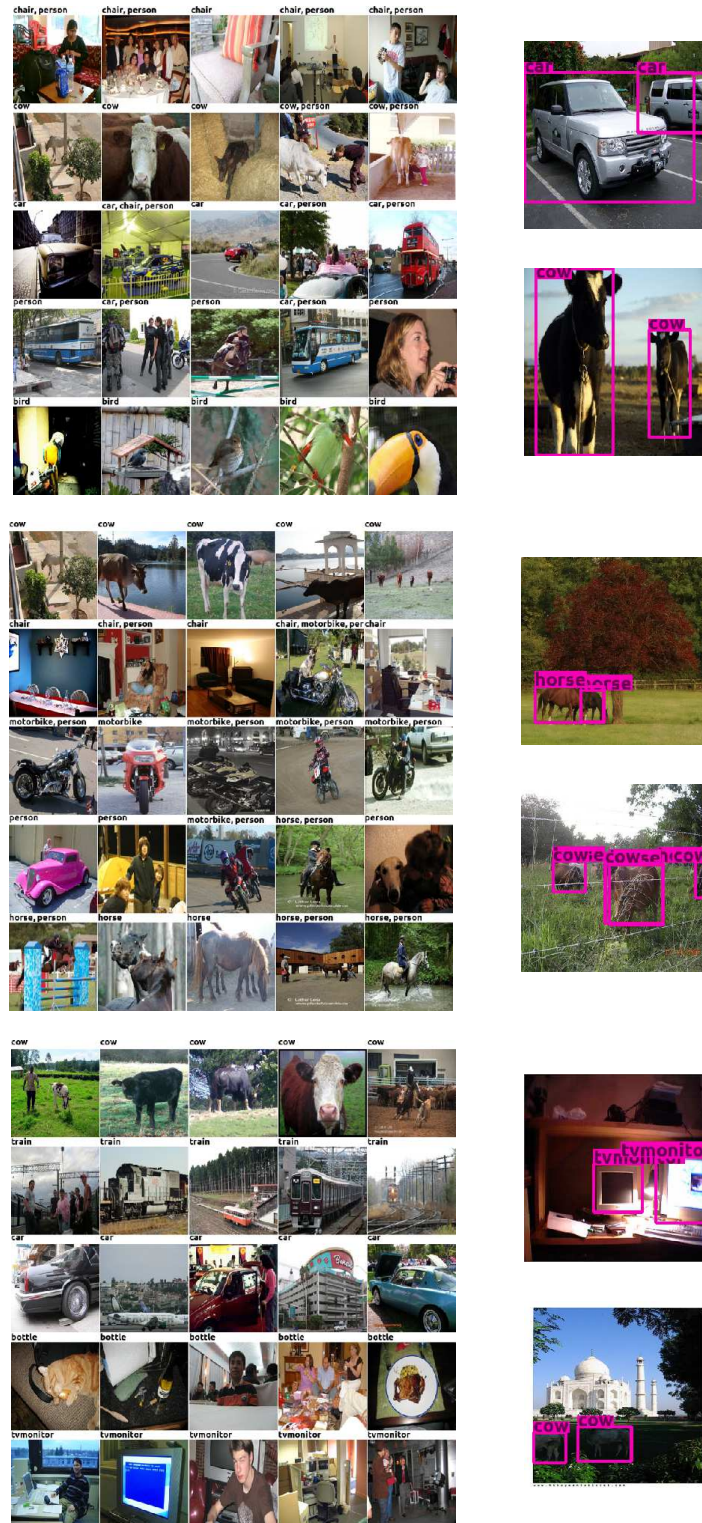


Figure 4.12: Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.

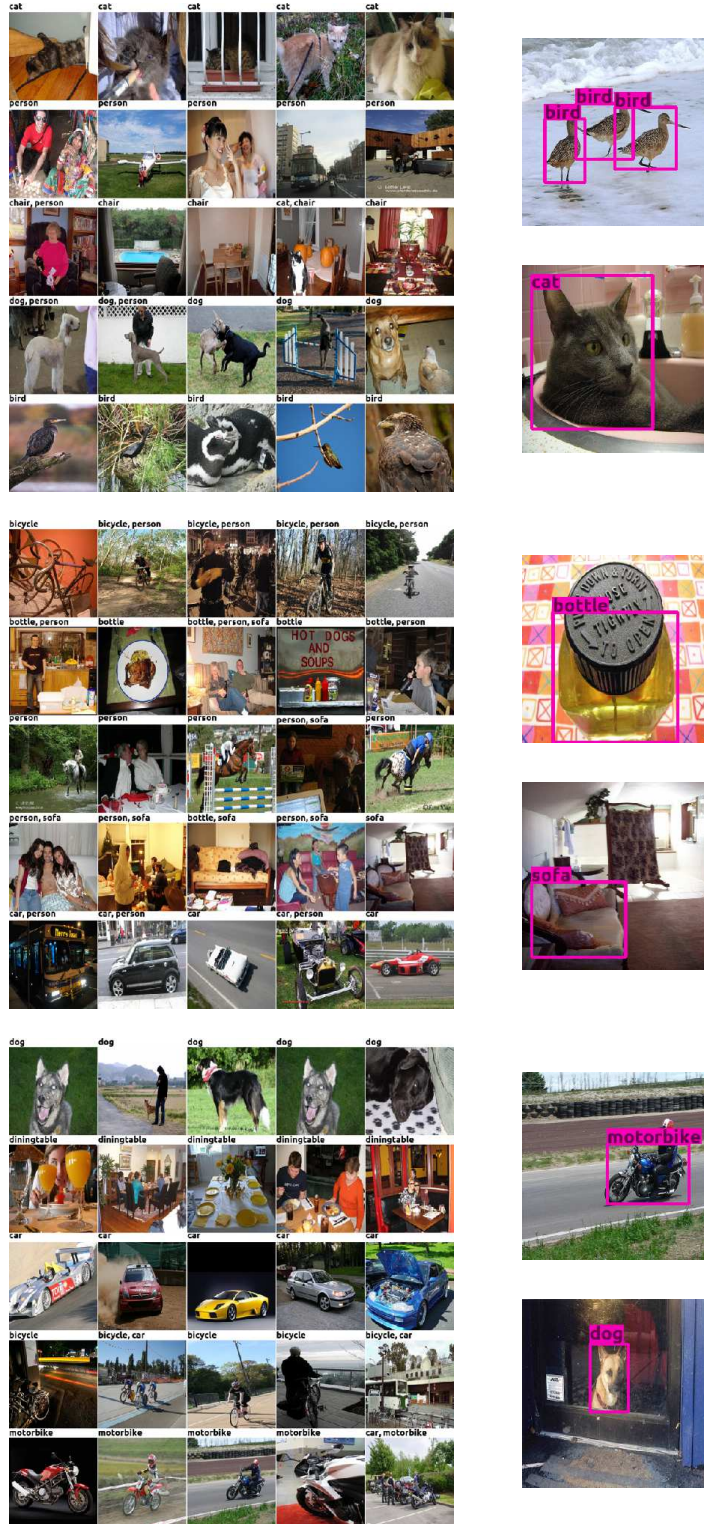


Figure 4.13: Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.

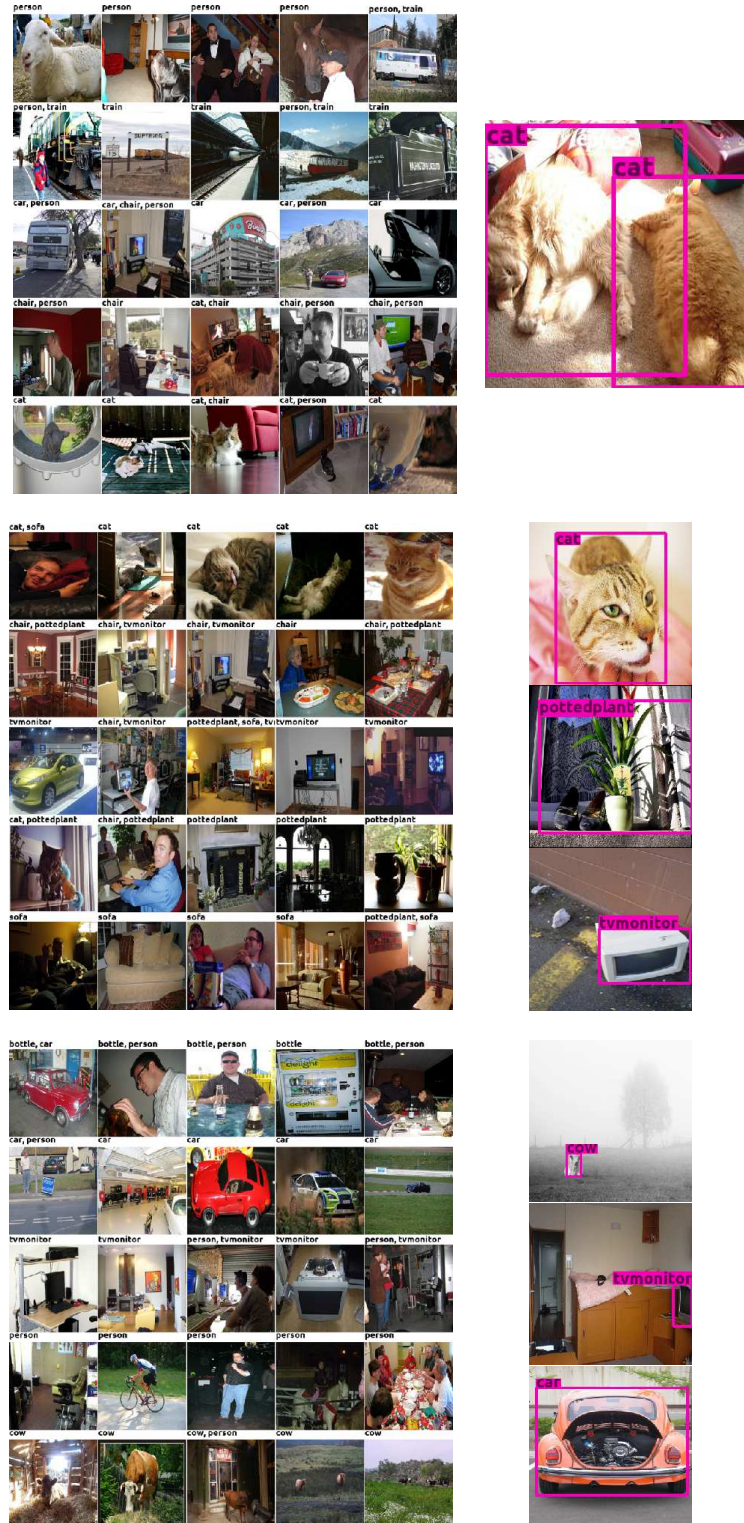


Figure 4.14: Few-shot WSOD on PASCAL VOC with $N = K = 5$. Given the support set shown on the left side the algorithm detects the object in query images on the right side.



Figure 4.15: 5-shot common object localization ($N = 1$) on MS COCO. Each row shows one common object localization problem. Ground-truth annotations (shown in green) are just for visualization and are not used in the algorithm. Top query bounding box prediction for each problem is shown in pink.

4.4 Summary

We have presented vMF-MIL, a multiple instance learning framework to address the problem of few-shot common object localization and WSOD. vMF-MIL uses a simple inductive bias in learning to combat the overfitting issue in few-shot learning. Specifically, instances of each class are assumed to form a cluster on a unit hypersphere, where the mean corresponds to the class prototype. Our experiments on few-shot common object localization illustrate the advantage of our simple approach over several state-of-the-art methods, improving the few-shot WSOD performance compared with the strong MI-SVM baseline.

Pairwise Similarity Knowledge Transfer for Weakly Supervised Object Localization

Weakly Supervised Object Localization (WSOL) methods only require image level labels as opposed to expensive bounding box annotations required by fully supervised algorithms. In this chapter, we study the problem of learning localization model on target classes with weakly supervised image labels, helped by a fully annotated source dataset. Typically, a WSOL model is first trained to predict class generic objectness scores on an off-the-shelf fully supervised source dataset and then it is progressively adapted to learn the objects in the weakly supervised target dataset. Inspired by the success of our finding common objects in previous chapter, we argue that learning only an objectness function is a weak form of knowledge-transfer and propose to learn a classwise pairwise similarity function that directly compares two input proposals as well. The combined localization model and the estimated object annotations are jointly learned in an alternating optimization paradigm as is typically done in standard WSOL methods. In contrast to the existing work that learns pairwise similarities, our approach optimizes a unified objective with convergence guarantee and it is computationally efficient for large-scale applications. Experiments on the COCO and ILSVRC 2013 detection datasets show that the performance of the localization model improves significantly with the inclusion of pairwise similarity function. For instance, in the ILSVRC dataset, the Correct Localization (CorLoc) performance improves from 72.8% to 78.2% which is a new state-of-the-art for WSOL task in the context of knowledge-transfer.

5.1 Introduction

Weakly Supervised Object Localization (WSOL) methods have gained a lot of attention in computer vision [Wan et al., 2019; Gao et al., 2019; Arun et al., 2019; Uijlings et al., 2018; Cinbis et al., 2016; Bilen et al., 2014; Deselaers et al., 2010; Tang et al., 2017, 2018]. Despite their supervised counterparts [He et al., 2017; Redmon et al.,

2016; Liu et al., 2016; Singh et al., 2018; Lin et al., 2017] that require the object class and their bounding box annotations, WSOL methods only require the image level labels indicating presence or absence of object classes. In spite of major improvements [Wan et al., 2019; Cinbis et al., 2016] in this area of research, there is still a large performance gap between weakly supervised and fully supervised object localization algorithms. In a successful attempt, WSOL methods are adopted to use an already annotated object detection dataset, called source dataset, to improve the weakly supervised learning performance in new classes [Uijlings et al., 2018; Hoffman et al., 2016]. These approaches learn transferable knowledge from the source dataset and use it to speed up learning new categories in the weakly supervised setting.

Multiple-Instance Learning (MIL) methods like MI-SVM [Andrews et al., 2003] are the predominant methods in weakly supervised object localization [Wan et al., 2019; Cinbis et al., 2016; Bilen et al., 2014]. Typically, images are decomposed into bags of object proposals and the problem is posed as selecting one proposal from each bag that contains an object class. MIL methods take advantage of alternating optimization to progressively learn a classwise objectness (unary) function and the optimal selection in re-training and re-localization steps, respectively. Typically, the source dataset is used to learn an initial generic objectness function which is used to steer the selection toward objects and away from background proposals [Uijlings et al., 2018; Hoffman et al., 2016; Bilen and Vedaldi, 2016; Rochan and Wang, 2015; Tang et al., 2014; Guillaumin and Ferrari, 2012]. However, solely learning an objectness measure is a sub-optimal form of knowledge-transfer as it can only discriminate objects from background proposals, while it is unable to discriminate between different object classes. Deselaers et al. [2010] propose to additionally learn a pairwise similarity function from the fully annotated dataset and frame WOSL as a graph labeling problem where nodes represent bags and each proposal corresponds to one label for the corresponding node. The edges which reflect the cost of wrong pairwise labeling are derived from the learned pairwise similarities. Additionally, they propose an ad-hoc algorithm to progressively adapt the scoring functions to learn the weakly supervised classes using alternating re-training and re-localization steps. Unlike the alternating optimization in MIL, re-training and re-localization steps in [Deselaers et al., 2010] does not optimize a unified objective and therefore the convergence of their method could not be guaranteed. Despite good performance on medium scale problems, this method is less popular especially in large scale problems where computing all the pairwise similarities is intractable.

In this work, we adapt the localization model in MIL to additionally learn a pairwise similarity function and use a two-step alternating optimization to jointly learn the augmented localization model and the optimal selection. In the re-training step, the pairwise and unary functions are learned given the current selected proposals for each class. In the re-localization step, the selected proposals are updated given the current pairwise and unary similarity functions. We show that with a properly chosen localization loss function, the objective in the re-localization step can be equivalently expressed as a graph labeling problem very similar to the model in [De-

[Deselaers et al., 2010]. We use the computationally effective iterated conditional modes (ICM) graph inference algorithm [Besag, 1986] in the re-localization step which updates the selection of one bag in each iteration. Unfortunately, the ICM algorithm is prone to local minimum and its performance is highly dependent on the quality of its initial conditions. Inspired by our work on few-shot object localization in Chapter 3, we divide the dataset into smaller mini-problems and solve each mini-problem individually using TRWS [Kolmogorov, 2006]. We combine the solutions of these mini-problems to initialize the ICM algorithm. Surprisingly, we observe that initializing ICM with the optimal selection from mini-problems of small sizes considerably improves the convergence point of ICM.

Our work addresses the main disadvantages of graph labeling algorithm in [Deselaers et al., 2010]. First, we formulate learning pairwise and unary functions and updating the optimal proposal selections with graph labeling within a two-step alternating optimization framework where each step is optimizing a unified objective and the convergence is guaranteed. Second, we propose a computationally efficient graph inference algorithm which uses a novel initialization method combined with ICM updates in the re-localization step. Our experiments show our method significantly improves the performance of MIL methods in large-scale COCO [Lin et al., 2014] and ILSVRC 2013 detection [Russakovsky et al., 2015] datasets. Particularly, our method sets a new state-of-the-art performance of 78.2% correct localization [Deselaers et al., 2010] for the WSOL task in the ILSVRC 2013 detection dataset¹.

5.2 Problem Description and Background

We review the standard dataset definition and optimization method for the weakly supervised object localization problem [Wan et al., 2019; Uijlings et al., 2018; Cinbis et al., 2016; Deselaers et al., 2010]. We follow the notations introduced in Chapter 3 with slight modifications specific to our application. Although some concepts have been introduced before, we will include them for the sake of completeness.

5.2.1 Dataset and Notation.

Suppose each image is decomposed into a collection of object proposals which form a bag $\mathcal{B} = \{\mathbf{e}_i\}_{i=1}^m$ where an object proposal $\mathbf{e}_i \in \mathbb{R}^d$ is represented by a d -dimensional feature vector. We denote $y(\mathbf{e}) \in \mathcal{C} \cup \{c_\emptyset\}$ the label for object proposal \mathbf{e} . In this definition \mathcal{C} is a set of object classes and c_\emptyset denotes the background class. Given a class $c \in \mathcal{C}$ we can also define the binary label

$$y_c(\mathbf{e}) = \begin{cases} 1 & \text{if } y(\mathbf{e}) = c \\ 0 & \text{otherwise.} \end{cases} \quad (5.1)$$

¹Source code is available on <https://github.com/AmirooR/Pairwise-Similarity-knowledge-Transfer-WSOL>

With this notation a dataset is a set of bags along with the labels. For a weakly supervised dataset, only bag-level labels that denote the presence/absence of objects in a given bag are available. More precisely, the label for bag \mathcal{B} is written as $\mathcal{Y}(\mathcal{B}) = \{c \mid \exists \mathbf{e} \in \mathcal{B} \text{ s.t. } y(\mathbf{e}) = c \in \mathcal{C}\}$. Let $Y_c(\mathcal{B}) \in \{0, 1\}$ denote the binary bag label which indicates the presence/absence of class c in bag \mathcal{B} .

Given a weakly supervised dataset $\mathcal{D}_{\mathcal{T}} = \{\mathcal{T}, \mathcal{Y}_{\mathcal{T}}\}$ called the target dataset, with $\mathcal{T} = \{\mathcal{B}_j\}_{j=1}^N$ and corresponding bag labels $\mathcal{Y}_{\mathcal{T}} = \{\mathcal{Y}(\mathcal{B})\}_{\mathcal{B} \in \mathcal{T}}$, the goal is to estimate the latent proposal unary labeling² \mathbf{y}_c for all object classes $c \in \mathcal{C}_{\mathcal{T}}$ in the target set.

For ease of notation, we also introduce a pairwise labeling function between pairs of proposals. The pairwise labeling function $r : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \{0, 1\}$ is designated to output 1 when two object proposals belong to the same object class and 0 otherwise, i.e.,

$$r(\mathbf{e}, \mathbf{e}') = \begin{cases} 1 & \text{if } y(\mathbf{e}) = y(\mathbf{e}') \neq c_{\emptyset} \\ 0 & \text{otherwise.} \end{cases} \quad (5.2)$$

Likewise, given a class c , two proposals are related under the class conditional pairwise labeling function $r_c : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \{0, 1\}$ if they both belong to class c . Similar to the unary labeling, since the pairwise labeling function is also defined over a finite set of variables, it can be seen as a vector. Unless we use the word vector or function, the context will determine whether we use the unary or pairwise labeling as a vector or a function. We use the “hat” notation to refer to the estimated (pseudo) unary or pairwise labeling by the weakly supervised learning algorithm.

5.2.2 Multiple-Instance Learning (MIL).

In standard MIL [Andrews et al., 2003], the problem is solved by jointly learning a unary score function $\psi_c^U : \mathbb{R}^d \rightarrow \mathbb{R}$ (typically represented by a neural network) and a feasible (pseudo) labeling $\hat{\mathbf{y}}_c$ that minimize the empirical unary loss

$$\mathcal{L}_c^U(\psi_c^U, \hat{\mathbf{y}}_c \mid \mathcal{T}) = \sum_{\mathcal{B} \in \mathcal{T}} \sum_{\mathbf{e} \in \mathcal{B}} \ell(\psi_c^U(\mathbf{e}), \hat{y}_c(\mathbf{e})), \quad (5.3)$$

where the loss function $\ell : \mathbb{R} \times \{0, 1\} \rightarrow \mathbb{R}$ measures the incompatibility between predicted scores $\psi_c^U(\mathbf{e})$ and the pseudo labels $\hat{y}_c(\mathbf{e})$. Here, likewise to the labeling, we denote the class score for all the proposals as a vector ψ_c^U . Note that the unary labeling $\hat{\mathbf{y}}_c$ is feasible if exactly one proposal has label 1 in each positive bag, and every other proposal has label 0 [Cinbis et al., 2016]. To this end, the set of feasible labeling \mathcal{F} can be defined as

$$\mathcal{F} = \left\{ \hat{\mathbf{y}}_c \mid \hat{y}_c(\mathbf{e}) \in \{0, 1\}, \sum_{\mathbf{e} \in \mathcal{B}} \hat{y}_c(\mathbf{e}) = Y_c(\mathcal{B}), \forall \mathcal{B} \in \mathcal{T} \right\}. \quad (5.4)$$

²Notice, the labeling is a function defined over a finite set of variables, which can be treated as a vector. Here, \mathbf{y}_c denotes the vector of labels $y_c(\mathbf{e})$ for all proposals \mathbf{e} .

Finally, the problem is framed as minimizing the loss over all possible vectors ψ_c^U (i.e., unary functions represented by the neural network) and the feasible labels \hat{y}_c

$$\begin{aligned} \min_{\psi_c^U, \hat{y}_c} \mathcal{L}_c^U(\psi_c^U, \hat{y}_c \mid \mathcal{T}), \\ \text{s.t. } \hat{y}_c \in \mathcal{F}. \end{aligned} \quad (5.5)$$

5.2.3 Optimization.

This objective is typically minimized in an iterative two-step alternating optimization paradigm [Ortega and Rheinboldt, 1970]. The optimization process starts with some initial value of the parameters and labels, and iteratively alternates between *re-training* and *re-localization* steps until convergence. In the re-training step, the parameters of the unary score function ψ_c^U are optimized while the labels \hat{y}_c are fixed. In the re-localization step, proposal labels are updated given the current unary scores. The optimization in the re-localization step is equivalent to assigning positive label to the proposal with the highest unary score within each positive bag and label 0 to all other proposals [Andrews et al., 2003]. Formally, label of the proposal $\mathbf{e} \in \mathcal{B}$ in bag \mathcal{B} is updated as

$$\hat{y}_c(\mathbf{e}) = \begin{cases} 1 & \text{if } Y_c(\mathcal{B}) = 1 \text{ and } \mathbf{e} = \operatorname{argmax}_{\mathbf{e}' \in \mathcal{B}} \psi_c^U(\mathbf{e}') \\ 0 & \text{otherwise.} \end{cases} \quad (5.6)$$

5.2.4 Knowledge Transfer.

In this chapter, we also assume having access to an auxiliary fully annotated dataset \mathcal{D}_S (source dataset) with object classes in \mathcal{C}_S which is a disjoint set from the target dataset classes, i.e., $\mathcal{C}_T \cap \mathcal{C}_S = \emptyset$. In the standard practice [Uijlings et al., 2018; Roohan and Wang, 2015; Guillaumin and Ferrari, 2012], the source dataset is used to learn a class-agnostic unary score $\psi^U : \mathbb{R}^d \rightarrow \mathbb{R}$ which measures how likely the input proposal \mathbf{e} tightly encloses a foreground object. Then, the unary score vector used in Equation (5.6) is adapted to $\psi_c^U \leftarrow \lambda \psi_c^U + (1 - \lambda) \psi^U$ for some $0 \leq \lambda \leq 1$. This steers the labeling toward choosing proposals that contain complete objects. Although the class-agnostic unary score function ψ^U is learned on the source classes, since objects share common properties, it transfers to the unseen classes in the target set.

5.3 Proposed Method

In addition to learning the unary scores, we also learn a classwise pairwise similarity function $\psi_c^P : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ that estimates the pairwise labeling between pairs of proposals. That is for the target class c , pairwise similarity score $\psi_c^P(\mathbf{e}, \mathbf{e}')$ between two input proposals $\mathbf{e}, \mathbf{e}' \in \mathbb{R}^d$ has a high value if two proposals are related, i.e., $\hat{r}_c(\mathbf{e}, \mathbf{e}') = 1$ and a low value otherwise. We define the empirical pairwise similarity loss to measure the incompatibility between pairwise similarity function predictions

and the pairwise labeling $\hat{\mathbf{r}}_c$

$$\mathcal{L}_c^P(\boldsymbol{\psi}_c^P, \hat{\mathbf{r}}_c | \mathcal{T}) = \sum_{\substack{\mathcal{B}, \mathcal{B}' \in \mathcal{T} \\ \mathcal{B} \neq \mathcal{B}'}} \sum_{\substack{\mathbf{e} \in \mathcal{B} \\ \mathbf{e}' \in \mathcal{B}'}} \ell(\psi_c^P(\mathbf{e}, \mathbf{e}'), \hat{\mathbf{r}}_c(\mathbf{e}, \mathbf{e}')), \quad (5.7)$$

where $\boldsymbol{\psi}_c^P$ denotes the vector of the pairwise similarities of all pairs of proposals, and $\ell : \mathbb{R} \times \{0, 1\} \rightarrow \mathbb{R}$ is the loss function. We define the overall loss as the weighted sum of the empirical pairwise similarity and the unary loss

$$\mathcal{L}_c(\boldsymbol{\psi}_c, \hat{\mathbf{z}}_c | \mathcal{T}) = \alpha \mathcal{L}_c^P(\boldsymbol{\psi}_c^P, \hat{\mathbf{r}}_c | \mathcal{T}) + \mathcal{L}_c^U(\boldsymbol{\psi}_c^U, \hat{\mathbf{y}}_c | \mathcal{T}), \quad (5.8)$$

where $\boldsymbol{\psi}_c = [\boldsymbol{\psi}_c^U, \boldsymbol{\psi}_c^P]$ is the vector of unary and pairwise similarity scores combined, and $\hat{\mathbf{z}}_c = [\hat{\mathbf{y}}_c, \hat{\mathbf{r}}_c]$ denotes the concatenation of unary and pairwise labeling vectors, and $\alpha > 0$ controls the importance of the pairwise similarity loss.

We employ alternating optimization to jointly optimize the loss over the parameters of the scoring functions $\boldsymbol{\psi}_c^U$ and $\boldsymbol{\psi}_c^P$ (re-training) and labelings $\hat{\mathbf{z}}_c$ (re-localization). In re-training, the objective function is optimized to learn the pairwise similarity and the unary scoring functions from the pseudo labels. In re-localization, we use the current scores to update the labelings.

Training the model with fixed labels, i.e., re-training step, is straightforward and can be implemented within any common neural network framework. We use sigmoid cross entropy loss in both empirical unary and pairwise similarity losses

$$\ell(x, y) = -(1 - y) \log(1 - \sigma(x)) - y \log(\sigma(x)), \quad (5.9)$$

where $x \in \mathbb{R}$ is the predicted logit, $y \in \{0, 1\}$ is the label, and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ denotes the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$. The choice of the loss function directly affects the objective function in the re-localization step. As we will show later, since sigmoid cross entropy loss is a linear function of label y it leads to a *linear objective function* in the re-localization step. To speed up the re-training step, we train pairwise similarity and unary scoring functions for all the classes together by optimizing the total loss

$$\mathcal{L}(\boldsymbol{\psi} | \hat{\mathbf{z}}, \mathcal{T}) = \sum_{c \in \mathcal{C}_{\mathcal{T}}} \mathcal{L}_c(\boldsymbol{\psi}_c, \hat{\mathbf{z}}_c | \mathcal{T}), \quad (5.10)$$

where $\boldsymbol{\psi} = [\boldsymbol{\psi}_c]_{c \in \mathcal{C}_{\mathcal{T}}}$ and $\hat{\mathbf{z}} = [\hat{\mathbf{z}}_c]_{c \in \mathcal{C}_{\mathcal{T}}}$ are the concatenation of respective vectors for all classes. Note that we learn the parameters of the scoring functions that minimize the loss, while $\hat{\mathbf{z}}$ remains fixed in this step. Since the dataset is large, we employ Stochastic Gradient Descent (SGD) with momentum for optimization. Additionally, we subsample proposals in each bag by sampling 3 proposals with foreground and 7 proposal with background label in each training iteration.

5.3.1 Re-localization

In this step, we minimize the empirical loss function in Equation (5.8) over the feasible labeling $\hat{\mathbf{z}}_c$ for the given model parameters. We first define feasible labeling set

\mathcal{A} and simplify the objective function to an equivalent, simple linear form. Then, we discuss algorithms to optimize the objective function in the large scale settings.

For $\hat{\mathbf{z}}_c$ to be feasible, labeling should be feasible, i.e., $\hat{\mathbf{y}}_c \in \mathcal{F}$ and pairwise labeling $\hat{\mathbf{r}}_c$ should also be consistent with the unary labeling. For dataset $\mathcal{D}_{\mathcal{T}}$ and target class c , this constraint set is expressed as

$$\mathcal{A} = \left\{ \hat{\mathbf{z}}_c \left| \begin{array}{ll} \sum_{\mathbf{e} \in \mathcal{B}} \hat{\mathbf{y}}_c(\mathbf{e}) = Y_c(\mathcal{B}) & \mathcal{B} \in \mathcal{T} \\ \sum_{\mathbf{e} \in \mathcal{B}} \hat{\mathbf{r}}_c(\mathbf{e}, \mathbf{e}') = \hat{\mathbf{y}}_c(\mathbf{e}') & \mathcal{B}, \mathcal{B}' \in \mathcal{T}, \mathcal{B}' \neq \mathcal{B}, \mathbf{e}' \in \mathcal{B}' \\ \hat{\mathbf{r}}_c(\mathbf{e}, \mathbf{e}'), \hat{\mathbf{y}}_c(\mathbf{e}) \in \{0, 1\} & c \in \mathcal{C}, \text{ for all } \mathbf{e} \text{ and } \mathbf{e}' \end{array} \right. \right\}. \quad (5.11)$$

Next, we simplify the loss function in the re-localization step. Let

$\mathcal{T}_c = \{\mathcal{B} \mid \mathcal{B} \in \mathcal{T}, c \in \mathcal{Y}(\mathcal{B})\}$ and $\mathcal{T}_{\bar{c}} = \mathcal{T} \setminus \mathcal{T}_c$ denote the set of positive and negative bags with respect to class c . The loss function in Equation (5.8) can be decomposed into three parts

$$\begin{aligned} \mathcal{L}_c(\boldsymbol{\psi}_c, \hat{\mathbf{z}}_c | \mathcal{T}) &= \mathcal{L}_c(\boldsymbol{\psi}_c, \hat{\mathbf{z}}_c | \mathcal{T}_c) + \mathcal{L}_c(\boldsymbol{\psi}_c, \hat{\mathbf{z}}_c | \mathcal{T}_{\bar{c}}) + \\ &\quad \sum_{\substack{\mathbf{e} \in \mathcal{B} \in \mathcal{T}_c \\ \mathbf{e}' \in \mathcal{B}' \in \mathcal{T}_{\bar{c}}}} \ell(\boldsymbol{\psi}_c^{\text{P}}(\mathbf{e}, \mathbf{e}'), \hat{\mathbf{r}}_c(\mathbf{e}, \mathbf{e}')) + \ell(\boldsymbol{\psi}_c^{\text{P}}(\mathbf{e}', \mathbf{e}), \hat{\mathbf{r}}_c(\mathbf{e}', \mathbf{e})), \end{aligned}$$

were the first two terms are the loss function in Equation (5.8) defined over the positive set \mathcal{T}_c and negative set $\mathcal{T}_{\bar{c}}$, and last term is the loss defined by the pairwise similarities between these two sets. Since for any feasible labeling all the proposals in negative bags has label 0 and remain fixed, only the value of $\mathcal{L}_c(\boldsymbol{\psi}_c, \hat{\mathbf{z}}_c | \mathcal{T}_c)$ changes within \mathcal{A} and other terms are constant. Furthermore, by observing that for sigmoid cross entropy loss in Equation (5.9) we have $\ell(x, y) = \ell(x, 0) - yx$, for $y \in [0, 1]^3$, we can further break down $\mathcal{L}_c(\boldsymbol{\psi}_c, \hat{\mathbf{z}}_c | \mathcal{T}_c)$ as

$$\begin{aligned} \mathcal{L}_c(\boldsymbol{\psi}_c, \hat{\mathbf{z}}_c | \mathcal{T}_c) &= \mathcal{L}_c(\boldsymbol{\psi}_c, \mathbf{0} | \mathcal{T}_c) \\ &\quad - \underbrace{\alpha \sum_{\substack{\mathcal{B}, \mathcal{B}' \in \mathcal{T}_c \\ \mathcal{B} \neq \mathcal{B}'}} \sum_{\substack{\mathbf{e} \in \mathcal{B} \\ \mathbf{e}' \in \mathcal{B}'}} \boldsymbol{\psi}_c^{\text{P}}(\mathbf{e}, \mathbf{e}') \hat{\mathbf{r}}_c(\mathbf{e}, \mathbf{e}') - \sum_{\mathcal{B} \in \mathcal{T}} \sum_{\mathbf{e} \in \mathcal{B}} \boldsymbol{\psi}_c^{\text{U}}(\mathbf{e}) \hat{\mathbf{y}}_c(\mathbf{e})}_{\mathcal{L}_{\text{reloc}}(\hat{\mathbf{z}}_c | \boldsymbol{\psi}_c, \mathcal{T}_c)}, \end{aligned} \quad (5.12)$$

where $\mathbf{0}$ is zero vector of the same dimension as $\hat{\mathbf{z}}_c$. Since the first term is constant with respect to $\hat{\mathbf{z}}_c = [\hat{\mathbf{y}}_c, \hat{\mathbf{r}}_c]$, re-localization can be equivalently done by optimizing $\mathcal{L}_{\text{reloc}}(\hat{\mathbf{z}}_c | \boldsymbol{\psi}_c, \mathcal{T}_c)$ over the feasible set \mathcal{A}

$$\begin{aligned} \min_{\hat{\mathbf{z}}_c} & -\alpha \hat{\mathbf{r}}_c^{\top} \boldsymbol{\psi}_c^{\text{P}} - \hat{\mathbf{y}}_c^{\top} \boldsymbol{\psi}_c^{\text{U}}, \\ \text{s.t. } & \hat{\mathbf{z}}_c \in \mathcal{A}, \end{aligned} \quad (5.13)$$

where we use the equivalent vector form to represent the re-localization loss in Equation (5.12). The re-localization optimization is an Integer Linear Program (ILP) and has been widely studied in literature [Schrijver, 1998]. The optimization can be

³See Appendix for the proof.

Algorithm 5: Re-localization

Input: Dataset $\mathcal{D}_{\mathcal{T}}$, batch size K , #epochs E
Output: Optimal unary labeling $\hat{\mathbf{y}}^*$

```

for  $c \in \mathcal{C}_{\mathcal{T}}$  do
   $T \leftarrow \text{round}(\frac{|\mathcal{T}_c|}{K})$ ,  $\hat{\mathbf{y}}_c \leftarrow \mathbf{0}$ 
  for  $t \leftarrow 1$  to  $T$  do
    // Sample next mini-problem
     $\mathcal{X} \sim \mathcal{T}_c$ 
    // Solve mini-problem with TRWS [Kolmogorov, 2006]
     $[\bar{\mathbf{y}}_c^*, \bar{\mathbf{r}}_c^*] \leftarrow \text{argmin}_{\bar{\mathbf{z}}_c} -\alpha \bar{\mathbf{r}}_c^\top \bar{\boldsymbol{\psi}}_c^P - \bar{\mathbf{y}}_c^\top \bar{\boldsymbol{\psi}}_c^U \text{ s.t. } \bar{\mathbf{z}}_c \in \bar{\mathcal{A}}$ 
    Update corresponding block of  $\hat{\mathbf{y}}_c$  with  $\bar{\mathbf{y}}^*$ 
  // Finetune for  $E$  epochs
   $\hat{\mathbf{y}}_c^* \leftarrow \text{ICM}(\hat{\mathbf{y}}_c, E)$ 
return  $\{\hat{\mathbf{y}}_c^*\}_{c \in \mathcal{C}_{\mathcal{T}}}$ 

```

equivalently expressed as a graph labeling problem with pairwise and unary potentials [Savchynskyy et al., 2019]. In the equivalent graph labeling problem, each bag is represented by a node in the graph where each proposal of the bag corresponds to a label of that node, and pairwise and unary potentials are equivalent to the negative pairwise similarity and negative unary scores in our problem. We discuss different graph inference methods and their limitations and present a practical method for large-scale settings.

Inference. Finding an optimal solution $\hat{\mathbf{z}}_c^*$ that minimizes the loss function defined in Equation (5.13) is NP-hard and thus not feasible to compute exactly, except in small cases. Loopy belief propagation [Weiss and Freeman, 2001], TRWS [Kolmogorov, 2006], and AStar [Bergtholdt et al., 2010], are among the many inference algorithms used for approximate graph labeling problem. Unfortunately, finding an approximate labeling quickly becomes impractical as the size of \mathcal{T}_c increases, since the dimension of $\hat{\mathbf{z}}_c$ increases quadratically with the numbers of bags in \mathcal{T}_c due to dense pairwise connectivity. Due to this limitation, we employ an older well-known iterated conditional modes (ICM) algorithm for optimization [Besag, 1986]. In each iteration, ICM only updates one unary label in $\hat{\mathbf{y}}_c$ along with the pairwise labels that are related to this unary label while all the other elements of $\hat{\mathbf{z}}_c$ are fixed. The block that gets updated in each iteration is shown in Figure 5.1. ICM performs coordinate descent type updates which monotonically decrease the objective function value. However, as the problem in Equation (5.13) is non-convex due to its discrete constraint set, ICM can get stuck at a local minimum and its convergence point is dependent on the quality of the initial labeling.



Figure 5.1: ICM iteration (left) and initialization (right) graphical models. In both graphs, each node represents a bag (with B proposals) within a dataset with $|\mathcal{T}_c| = 9$ bags. **Left:** ICM updates the unary label of the selected node (shown in green). Edges show all the pairwise labels that gets updated in the process. Since the unary labeling of other nodes are fixed each blue edge represents B elements in vector $\hat{\mathbf{r}}_c$. **Right:** For initialization we divide the dataset into smaller mini-problems (with size $K = 3$ in this example) and solve each of them individually. Each edge represents B^2 pairwise scores that need to be computed.

To initialize the ICM algorithm, we divide the large-scale problem into several disjoint smaller re-localization problems that can be efficiently solved by the proposed co-localization algorithm in Chapter 3.

Let $\mathcal{X} \in \mathcal{T}_c$ be one of the disjoint mini-problems. We optimize the re-localization problem $\mathcal{L}_{\text{reloc}}(\bar{\mathbf{z}}_c \mid \bar{\boldsymbol{\psi}}_c, \mathcal{X})$ where vectors $\bar{\mathbf{z}}_c$ and $\bar{\boldsymbol{\psi}}_c$ are parts of vectors $\hat{\mathbf{z}}_c$ and $\boldsymbol{\psi}_c$ that are within the mini-problem defined by \mathcal{X} (see Figure 5.1). We repeat this problem for all the mini-problems. We note that the division of the dataset is performed randomly. In our pre-liminary experiments, we observed that our method always converged to similar CorLoc values when we repeated our experiments multiple times. However, finding an optimal policy for this task is an interesting problem for future works. The complete re-localization step is illustrated in Algorithm 5.

Complexity. We practically observed that computing the pairwise similarity scores is the computation bottleneck, thus we analyze the time complexities in terms of the number of pairwise similarity scores each algorithm computes. Let $M = \max_{c \in \mathcal{C}_T} |\mathcal{T}_c|$ denotes the maximum number of positive bags, and $B = \max_{B \in \mathcal{T}} |B|$ be the maximum bag size. To solve the exact optimization in Equation (5.13), we need to compute the vector $\boldsymbol{\psi}_c$ with $\mathcal{O}(B^2 M^2)$ elements. On the other hand, each iteration of ICM only computes $\mathcal{O}(BM)$ pairwise similarity scores. We additionally compute a total of $\mathcal{O}(MKB^2)$ pairwise similarity scores for the initialization where K is the size of the mini-problem. Thus, ICM algorithm would be asymptotically more efficient than the exact optimization in terms of total number of pairwise similarity scores it computes, if it is run for $\Omega(MB)$ iterations or $E = \Omega(B)$ epochs. We practically observe that by initializing ICM with the result of the proposed initialization scheme, it convergences in few epochs.

Even though Equation (5.13) is similar to the DenseCRF formulation [Krähenbühl and Koltun, 2011], the pairwise potentials are not amenable to the efficient filtering method [Adams et al., 2010] which is the backbone of DenseCRF methods [Krähenbühl and Koltun, 2011; Ajanthan et al., 2017]. Therefore, it is intractable to use any existing sophisticated optimization algorithm except for ICM and MeanField [Blake et al., 2011]. Nevertheless, our block-wise application of TRWS provides an effective initialization for ICM. We additionally experimented with the block version of

ICM [Savchynskyy et al., 2019] but it performs similarly while being slower.

5.3.2 Knowledge Transfer

To transfer knowledge from the fully annotated source set \mathcal{D}_S , we first learn *class generic* pairwise similarity $\psi^P : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ and unary $\psi^U : \mathbb{R}^d \rightarrow \mathbb{R}$ functions from the source set. Since the labels are available for all the proposals in the source set, learning the pairwise and unary functions is straightforward. We simply use stochastic gradient descent (SGD) to optimize the loss

$$\mathcal{L}^T(\psi^P, \psi^U | \mathcal{S}, \mathbf{r}, \mathbf{o}) = \alpha \sum_{\substack{\mathcal{B}, \mathcal{B}' \in \mathcal{S} \\ \mathcal{B} \neq \mathcal{B}'}} \sum_{\substack{\mathbf{e} \in \mathcal{B} \\ \mathbf{e}' \in \mathcal{B}'}} \ell(\psi^P(\mathbf{e}, \mathbf{e}'), r(\mathbf{e}, \mathbf{e}')) + \sum_{\mathcal{B} \in \mathcal{S}} \sum_{\mathbf{e} \in \mathcal{B}} \ell(\psi^U(\mathbf{e}), o(\mathbf{e})), \quad (5.14)$$

where $o(\mathbf{e}) \in \{0, 1\}$ is class generic objectness label, i.e., ,

$$o(\mathbf{e}) = \begin{cases} 1 & \text{if } y(\mathbf{e}) \neq c_\emptyset \\ 0 & \text{otherwise,} \end{cases} \quad (5.15)$$

and relation function $r : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is defined by Equation (5.6). Here we do not use hat notation since groundtruth proposal labels are available for the source dataset \mathcal{D}_S . We skip the details as the loss in Equation (5.14) has a similar structure to the re-training loss. Note that in general the class generic functions ψ^U and ψ^P and class specific functions ψ_c^U and ψ_c^P use different feature sets extracted from different networks. Having learned these functions, we adapt both pairwise similarity and score vectors in the re-localization step in Algorithm 5 as

$$\begin{aligned} \psi_c^P &\leftarrow (1 - \lambda_1) \psi_c^P + \lambda_1 \psi^P \\ \psi_c^U &\leftarrow (1 - \lambda_2) \psi_c^U + \lambda_2 \psi^U, \end{aligned}$$

where $0 \leq \lambda_1, \lambda_2 \leq 1$ controls the weight of transferred and adaptive functions in pairwise similarity and unary functions respectively. We start the alternating optimization with a *warm-up* re-localization step where only the learned class generic pairwise and unary functions above are used in the re-localization algorithm, i.e., , $\lambda_1, \lambda_2 = 1$. The warm-up re-localization step provides high quality pseudo labels to the first re-training step and speeds up the convergence of the alternating optimization algorithm.

5.3.3 Network Architectures

Proposal and feature extraction. Following the experiment protocol in [Uijlings et al., 2018], we use a Faster-RCNN [Ren et al., 2015] model trained on the source dataset \mathcal{D}_S to extract region proposals from each image. We keep the box features in the last layer of Faster-RCNN as transferred features to be used in the class generic score functions. Following [Uijlings et al., 2018; Hoffman et al., 2016; Tang et al.,

2016], we extract AlexNet [Krizhevsky et al., 2012] feature vectors from each proposal as input to the class specific scoring functions ψ_c^U and ψ_c^P .

Scoring functions. Let \mathbf{e} and \mathbf{e}' denote features in \mathbb{R}^d extracted from two image proposals. Linear layers are employed to model the class generic unary function ψ^U and all the classwise unary functions ψ_c^U i.e. $\psi_c^U(\mathbf{e}) = \mathbf{w}_c^\top \mathbf{e} + b_c$ where $\mathbf{w}_c \in \mathbb{R}^d$ is the weight and $b_c \in \mathbb{R}$ is the bias parameter. We employ the relation network architecture from Chapter 3 to model the pairwise similarity functions ψ^P and ψ_c^P . We share the parameters of the embedding functions in $\psi_c^P(\mathbf{e}, \mathbf{e}')$ for all the classes $c \in \mathcal{C}_T$ to reduce the number of parameters.

5.4 Experiments

We evaluate the main applicability of our technique on different weakly supervised datasets and analyze how each part affects the final results in our method. We report the widely accepted Correct Localization (CorLoc) metric [Deselaers et al., 2010] for the object localization task as our evaluation metric. We report with 0.5 and 0.7 thresholds in our experiments. All experiments are done on a single Nvidia GTX 1080 GPU and 3.2GHz Intel(R) Xeon(R) CPU with 128 GB of RAM.

5.4.1 COCO 2017 Dataset

We employ a split of COCO 2017 [Lin et al., 2014] dataset to evaluate the effect of different initialization strategies and our pairwise retraining and re-localization steps. The dataset has 80 classes in total. We take the same split of [Bansal et al., 2018; Shaban et al., 2019] with 63 source \mathcal{C}_S and 17 target \mathcal{C}_T classes and follow [Shaban et al., 2019] to create the source and target splits to create source and target datasets with 111,085 and 8,245 images, respectively.

Similar to [Shaban et al., 2019], we use Faster-RCNN [Ren et al., 2015] with ResNet 50 [He et al., 2016] backbone as our proposal generator and feature extractor for knowledge-transfer. We keep the top $B = 100$ proposals generated by Faster-RCNN for experiments on the COCO 2017.

We first study different approaches for initializing the ICM method in the re-localization step. Then, we present the result of the full proposed method and compare it with other baselines.

5.4.1.1 Initialization Scheme

Since the ICM algorithm is sensitive to initialization, we devise the following experiment to evaluate different initialization methods. To limit total running time of the experiment, we only do this evaluation in the warm-up re-localization step. We start by training class generic unary and pairwise similarity scoring functions on the source dataset \mathcal{D}_S . Next, we initialize the labeling of the images in \mathcal{D}_T using the following initialization strategies:

Table 5.1: Performance and time comparison of different initialization algorithms. Our method exhibits the highest initialization performance for $K = 64$, however, we get similar performance for $4 \leq K \leq 64$ after applying ICM.

Method	Init. CorLoc (%)	Init.+ICM CorLoc (%)	Init.+ICM Energy ($\times 10^5$)	Init. Time (Minutes)
Random	0.0	46.0	7.95	0
Objectness	38.7	46.0	7.95	0
Ours $K = 2$	40.9	46.0	7.95	0
Ours $K = 4$	44.4	46.9	7.88	1
Ours $K = 8$	45.2	46.9	7.88	1
Ours $K = 64$	46.1	47.0	7.88	1706

- Random: randomly select a proposal from each bag.
- Objectness: select the proposal with the highest unary score from each bag.
- Proposed initialization method: Proposed initialization method discussed in Section 5.3.1. We conduct the experiment with different mini-problem sizes $K \in \{2, 4, 8, 64\}$. We use TRWS [Kolmogorov, 2006] algorithm for inference in each mini-problem.

After initialization, we perform ICM to converge to a local minimum of the original labeling problem. Table 5.1 compares performance and time of different initialization algorithms. Our method exhibits the highest initialization performance when $K = 64$, however, we get similar performance for $4 \leq K \leq 64$ after applying ICM. On the other hand, $K = 2$ and other initialization algorithms have a lower performance. We note that increasing K beyond 64 might increase the ICM performance even further but it becomes impractical due to its computational cost as the time column in Table 5.1 shows.

5.4.1.2 Full Pipeline

Here, we conduct an experiment to determine the importance of learning pairwise similarities on the COCO dataset. We compare our full method with the unary method which only learns and uses unary scoring functions during, warm-up, re-training and re-localization steps. This method is analogous to [Uijlings et al., 2018]. The difference is that it uses cross entropy loss and SGD training instead of Support Vector Machine used in [Uijlings et al., 2018]. Also, we do not employ hard-negative mining after each re-training step. For this experiment, we use mini-problems of size $K = 4$ for initializing ICM. We run both methods for 5 iterations of alternating optimization on the target dataset. Our method achieves 48.3% compared to 39.4% CorLoc@0.5 of the unary method. This clearly shows the effectiveness of our pairwise similarity learning.

Table 5.2: Performance of different methods on ILSVRC 2013. Proposal generators and their backbone models are shown in the second and third column. Total time is shown in “Training+Inference” format. CorLoc is reported on the target set. The last column shows the performance of an object detector trained on the target set and evaluated on the target test set. *The first 3 methods use RCNN detector with AlexNet backbone while other methods utilize Faster-RCNN detector with Inception-Resnet backbone.

Method	Proposal Generator	Backbone	CorLoc@0.5	CorLoc@0.7	Time(hours)	mAP@0.5
LSDA [Hoffman et al., 2016]	Selective Search [Uijlings et al., 2013]	AlexNet [Krizhevsky et al., 2012]	28.8	-	-	18.1*
[Tang et al., 2016]	Selective Search [Uijlings et al., 2013]	AlexNet [Krizhevsky et al., 2012]	-	-	-	20.0*
[Uijlings et al., 2018]	SSD [Liu et al., 2016]	Inception-V3 [Szegedy et al., 2016]	70.3	58.8	-	23.3*
[Uijlings et al., 2018]	Faster-RCNN	Inception-Resnet	74.2	61.7	-	36.9
Warm-up (unary)	Faster-RCNN	Inception-Resnet	68.9	59.5	0	-
Warm-up	Faster-RCNN	Inception-Resnet	73.8	62.3	5+3	-
Unary	Faster-RCNN	Inception-Resnet	72.8	62.0	13+2	38.1
Full (ours)	Faster-RCNN	Inception-Resnet	78.2	65.5	65+13	41.7
Supervised[Uijlings et al., 2018]						46.2

5.4.2 ILSVRC 2013 Detection Dataset

We closely follow the experimental protocol of [Uijlings et al., 2018; Hoffman et al., 2016; Tang et al., 2016] to create source and target datasets on ILSVRC 2013 [Russakovsky et al., 2015] detection dataset. The val1 split is augmented with images from the training set such that each class has 1000 annotated bounding boxes in total [Girshick et al., 2014]. The dataset has 200 categories with full bounding box annotations. We use the first 100 alphabetically ordered classes as source categories \mathcal{C}_S and the remaining 100 classes as target categories \mathcal{C}_T . The dataset is divided into source training set \mathcal{D}_S with 63k images, target set \mathcal{D}_T with 65k images, and a target test set with 10k images. The source dataset \mathcal{D}_S is formed by all images in the augmented val1 set that have an object in \mathcal{C}_S . As for our target dataset \mathcal{D}_T , all images which have an object in the target categories \mathcal{C}_T are used and all the bounding box annotations are removed and only the bag labels \mathcal{Y}_T are kept.

We report CorLoc of different algorithms on \mathcal{D}_T . Similar to previous works [Uijlings et al., 2018; Hoffman et al., 2016; Tang et al., 2016], we additionally train a detector from the output of our method on target set \mathcal{D}_T , and evaluate it on the target test set. For a fair comparison, we use a similar proposal generator and multi-fold strategy as [Uijlings et al., 2018]. We use Faster-RCNN [Ren et al., 2015] with Inception-Resnet [Szegedy et al., 2017] backbone trained on source dataset \mathcal{D}_S for object bounding box generation. Following [Uijlings et al., 2018], we perform multi-folding strategy [Cinbis et al., 2016] to avoid overfitting: the target dataset is split into 10 random folds and then re-training is done on 9 folds while re-localization is performed on the remaining fold. Values of hyper-parameters are obtained using cross validation. The experiment on COCO suggests a small mini-problem size K would be sufficient to achieve good performance in the re-localization step. We use $K = 8$ to balance the time and accuracy in this experiment.

5.4.2.1 Baselines and Results

We compare our method with two knowledge transfer techniques [Hoffman et al., 2016; Uijlings et al., 2018] for WSOL. In addition, we demonstrate the results of the following baselines that only use unary scoring function:

- Warm-up (unary): To see the importance of learning pairwise similarities in knowledge-transfer, we perform the warm-up re-localization with only the transferred unary scores ψ^U . This can be achieved by simply selecting the box with the highest unary score within each bag. We compare this results with the result of the warm-up step which uses both pairwise and unary scores in knowledge-transfer.
- Unary: Standard MIL objective in Equation (5.5) which only learns labeling and the unary scoring function.

We compare these results with our full pipeline which starts with a warm-up re-localization step followed by alternating re-training and re-localization steps. The results are illustrated in Table 5.2. Compared to [Uijlings et al., 2018], our method improves the CorLoc@0.5 performance on the target set by 4% and mAP@0.5 on the target test set by 4.8%. Warm-up re-localization improves CorLoc performance of warm-up (unary) by 4.9% with transferring a pairwise similarity measure from the source classes. Note that the result of warm-up step without any re-training performs on par with the [Uijlings et al., 2018] MIL method. The CorLoc performance at the stricter IoU > 0.7 also shows similar results. Compared to [Uijlings et al., 2018], our implementation of the MIL method performs worse with IoU threshold 0.5 but better with stricter threshold 0.7. We believe the reason is having a different loss function and hard-negative mining in [Uijlings et al., 2018].

Some qualitative success cases on ILSVRC 2013 dataset are illustrated in Figure 5.2, and Figure 5.3. Failure cases on this dataset are also presented in Figure 5.4. Refer to Figure 5.2 for more information on bounding box tags. As a success example, in the 3rd row and second image from left in Figure 5.2, note that the target object is the tiny bottle. Although, the dog is the most salient object in this image and is selected by other methods, our method could find the bottle perfectly. Overall, selection of a visually similar object in the image, occlusion and disconnected objects, multi-part objects, and even errors in dataset annotations are the source of most of the failures on this dataset. Figure 5.5 shows the qualitative results on the COCO dataset.

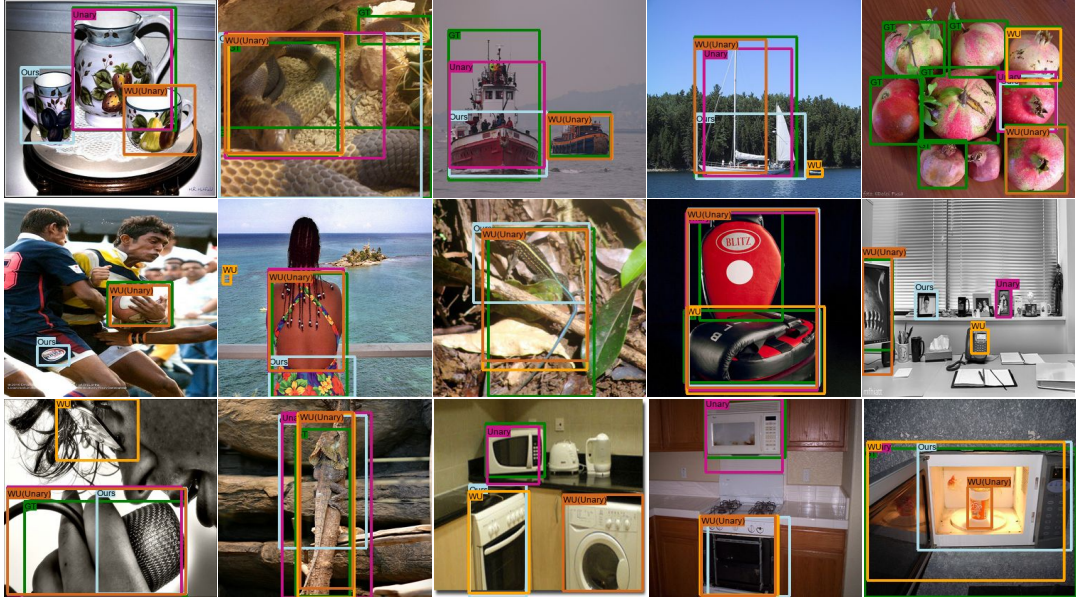


Figure 5.4: Failure cases on ILSVRC 2013 dataset (see Figure 5.2 for details).

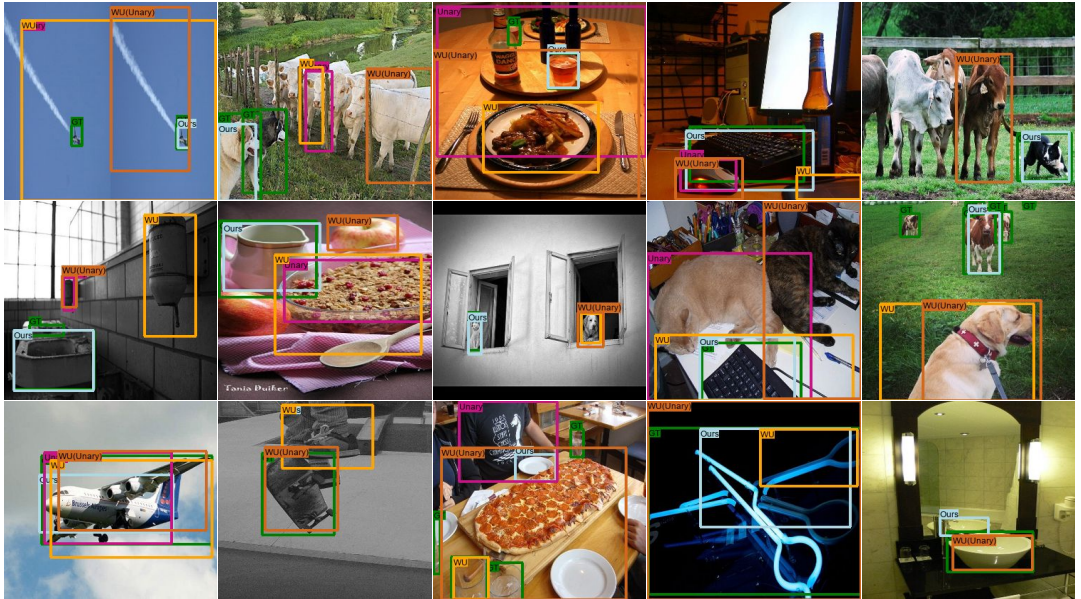


Figure 5.5: Success and failure cases on COCO dataset. First two rows show the success cases of our method while the last row shows the failure cases. For a success case example, the task for the middle image in the second row is to find the cat. However, other methods selected the dog in this case. For a failure case, in the bottom right image, our method selected the object inside the mirror (see Figure 5.2 for details).

5.5 Summary

We have studied the problem of learning localization models on target classes from weakly supervised training images, helped by a fully annotated source dataset. We have adapted MIL localization model by adding a classwise pairwise similarity module that learns to directly compare two input proposals. Similar to the standard MIL approach, we have learned the augmented localization model and annotations jointly by two-step alternating optimization. We have represented the re-localization step as a graph labeling problem and proposed a computationally efficient inference algorithm for optimization. Compared to the previous work [Deselaers et al., 2010] that uses pairwise similarities for this task, the proposed method is represented in alternating optimization framework with convergence guarantee and is computationally efficient in large-scale settings. The experiments show that learning pairwise similarity function improves the performance of WSOL over the standard MIL.

Intra Order-Preserving Functions for Calibration of Multi-Class Neural Networks

Predicting calibrated confidence scores for multi-class deep networks is important for avoiding rare but costly mistakes. A common approach is to learn a post-hoc calibration function that transforms the output of the original network into calibrated confidence scores while maintaining the network’s accuracy. However, previous post-hoc calibration techniques work only with simple calibration functions, potentially lacking sufficient representation to calibrate the complex function landscape of deep networks. In this chapter, we aim to learn general post-hoc calibration functions that can preserve the top- k predictions of any deep network. We call this family of functions *intra order-preserving* functions. We propose a new neural network architecture that represents a class of intra order-preserving functions by combining common neural network components. Additionally, we introduce *order-invariant* and *diagonal* sub-families, which can act as regularization for better generalization when the training data size is small. We show the effectiveness of the proposed method across a wide range of datasets and classifiers. Our method outperforms state-of-the-art post-hoc calibration methods, namely temperature scaling and Dirichlet calibration, in several evaluation metrics for the task.

6.1 Introduction

Deep neural networks have demonstrated impressive accuracy in classification tasks, such as image recognition [He et al., 2016; Ren et al., 2015] and medical research [Jiang et al., 2012; Caruana et al., 2015]. These exciting results have recently motivated engineers to adopt deep networks as default components in building decision systems; for example, a multi-class neural network can be treated as a probabilistic predictor and its softmax output can provide the confidence scores of different actions for the downstream decision making pipeline [Girshick, 2015; Cao et al., 2017; Mozafari et al., 2019]. While this is an intuitive idea, recent research has found that deep networks, despite being accurate, can be overconfident in their predictions, exhibiting

high calibration error [Maddox et al., 2019; Guo et al., 2017; Kendall and Gal, 2017]. In other words, trusting the network’s output naively as confidence scores in system design could cause undesired consequences: a serious issue for applications where mistakes are costly, such as medical diagnosis and autonomous driving.

A promising approach to address the miscalibration is to augment a given network with a parameterized calibration function, such as extra learnable layers. This additional component is tuned post-hoc using a held-out calibration dataset, so that the effective full network becomes calibrated [Guo et al., 2017; Kull et al., 2019, 2017b,a; Platt, 1999; Zadrozny and Elkan, 2001]. In contrast to usual deep learning, the calibration dataset here is typically *small*. Therefore, learning an overly general calibration function can easily overfit and actually reduce the accuracy of the given network [Guo et al., 2017; Kull et al., 2019]. Careful design regularization and parameterization of calibration functions is imperative.

A classical non-parametric technique is isotonic regression [Zadrozny and Elkan, 2002], which learns a monotonic staircase calibration function with minimal change in the accuracy. But the complexity of non-parametric learning can be too expensive to provide the needed generalization [Kull et al., 2017b,a]. By contrast, [Guo et al., 2017] proposed to learn a scalar parameter to rescale the original output logits, at the cost of being suboptimal in calibration [Maddox et al., 2019]; see also Section 6.5. Recently, [Kull et al., 2019] proposed to learn linear transformations of the output logits. While this scheme is more expressive than the temperature scaling above, it does not explore non-linear calibration functions.

In general, a preferable hypothesis space needs to be expressive and, at the same time, provably preserve the accuracy of any given network it calibrates. Limiting the expressivity of calibration functions can be an issue, especially when calibrating deep networks with complicated landscapes.

The main contribution of this chapter is introducing a learnable space of functions, called *intra order-preserving* family. Informally speaking, an intra order-preserving function $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a vector-valued function whose output values always share the same ordering as the input values across the n dimensions. For example, if $\mathbf{x} \in \mathbb{R}^n$ is increasing from coordinate 1 to n , then so is $f(\mathbf{x})$. In addition, we introduce order-invariant and diagonal structures, which utilize the shared characteristics between different input dimensions to improve generalization. We leverage the inductive bias that the calibration functions can have similar behavior across different classes (due to the common characteristics between the classes) to design these subfamilies of order-preserving functions. For illustration, we depict instances of 3-dimensional intra order-preserving and order-invariant functions defined on the unit simplex and compare them to an unconstrained function in Figure 6.1. We use arrows to show how inputs on the simplex are mapped by each function. Each colored subset in the simplex denotes a region with the same input order; for example, we have $x_3 > x_2 > x_1$ inside the red region where the subscript i denotes the i th element of a vector. For the intra order-preserving function shown in Figure 6.1a arrows stay within the same colored region as the inputs, but the vector fields in two different colored region are independent to each other. Order-invariant function in

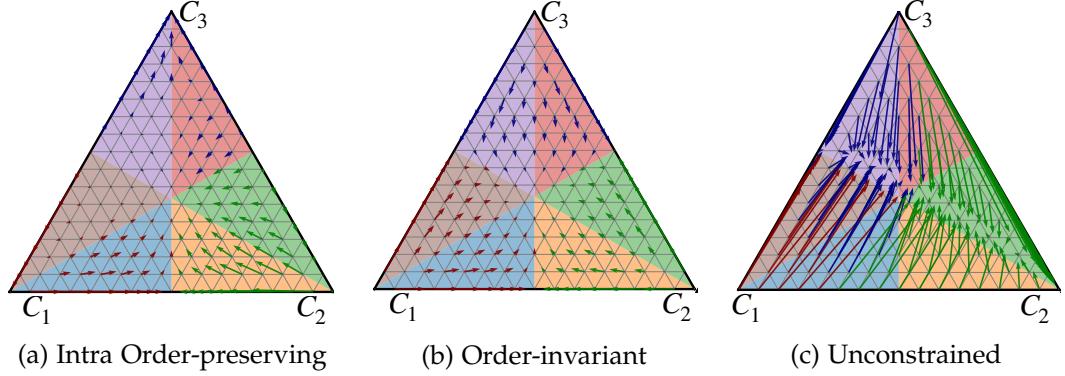


Figure 6.1: Comparing instances of intra order-preserving and order-invariant family defined on the 2-dimensional unit simplex. Points $C_1 = [1, 0, 0]^\top$, $C_2 = [0, 1, 0]^\top$, $C_3 = [0, 0, 1]^\top$ are the simplex corners. Arrows depict how an input is mapped by each function. Unconstrained function freely maps the input probabilities, intra order-preserving function enforces the outputs to stay within the same colored region as the inputs, and order-invariant function further enforces the vector fields to be the same among all the 6 colored regions as reflected in the symmetry in the visualization.

Figure 6.1b further keeps the function permutation invariant, enforcing the vector fields to be the same among all the 6 colored regions (as reflected in the symmetry in Figure 6.1b). This property of order-preserving functions significantly reduce the hypothesis space in learning, from the functions on whole simplex to functions on one colored region, for better generalization.

We identify necessary and sufficient conditions for describing intra order-preserving functions, study their differentiability, and propose a novel neural network architecture that can represent complex intra order-preserving function through common neural network components. From practical point of view, we devise a new post-hoc network confidence calibration technique using different intra order-invariant sub-families. Because a post-hoc calibration function keeps the top- k class prediction *if and only if* it is an intra order-preserving function, learning the post-hoc calibration function within the intra order-preserving family presents a solution to the dilemma between accuracy and flexibility faced in the previous approaches. We conduct several experiments to validate the benefits of learning with these new functions for post-hoc network calibration. The results demonstrate improvement over various calibration performance metrics, compared with the original network, temperature scaling [Guo et al., 2017], and Dirichlet calibration [Kull et al., 2019].

6.2 Problem Setup

We address the problem of calibrating neural networks for n -class classification. Let define $\llbracket n \rrbracket := \{1, \dots, n\}$, $\mathcal{Z} \subseteq \mathbb{R}^d$ be the domain, $\mathcal{Y} = \llbracket n \rrbracket$ be the label space, and let Δ^n denote the $n - 1$ dimensional unit simplex. Suppose we are given a trained probabilistic predictor $\phi_o : \mathbb{R}^d \rightarrow \Delta^n$ and a small calibration dataset \mathcal{D}_c of i.i.d. samples

drawn from an unknown distribution π on $\mathcal{Z} \times \mathcal{Y}$. For simplicity of exposition, we assume that ϕ_o can be expressed as the composition $\phi_o =: \sigma_{\text{SM}} \circ \mathbf{g}$, with $\mathbf{g} : \mathbb{R}^d \rightarrow \mathbb{R}^n$ being a non-probabilistic n -way classifier and $\sigma_{\text{SM}} : \mathbb{R}^n \rightarrow \Delta^n$ being the softmax operator¹, i.e. $\sigma_{\text{SM}}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)}$, for $i \in \mathcal{Y}$, where the subscript i denotes the i th element of a vector. When queried at $\mathbf{z} \in \mathcal{Z}$, the probabilistic predictor ϕ_o returns $\arg\max_i \phi_{o,i}(\mathbf{z})$ as the predicted label and $\max_i \phi_{o,i}(\mathbf{z})$ as the associated confidence score. (The top- k prediction is defined similarly.) We say $\mathbf{g}(\mathbf{z})$ is the *logits* of \mathbf{z} with respect to ϕ_o .

Given ϕ_o and \mathcal{D}_c , our goal is to learn a post-hoc calibration function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ such that the new probabilistic predictor $\phi := \sigma_{\text{SM}} \circ \mathbf{f} \circ \mathbf{g}$ is better calibrated *and* keeps the accuracy (or similar performance concepts like top- k accuracy) of the original network ϕ_o . That is, we want to learn new logits $\mathbf{f}(\mathbf{g}(\mathbf{z}))$ of \mathbf{z} . As we will discuss, this task is non-trivial, because while learning \mathbf{f} might improve calibration, doing so could also risk over-fitting to the small dataset \mathcal{D}_c and damaging accuracy. To make this statement more precise, below we first review the definition of perfect calibration [Guo et al., 2017] and common calibration metrics and then discuss challenges in learning \mathbf{f} with \mathcal{D}_c .

Definition 1. For a distribution π on $\mathcal{Z} \times \mathcal{Y}$ and a probabilistic predictor $\psi : \mathbb{R}^d \rightarrow \Delta^n$, let random variables $\mathbf{z} \in \mathcal{Z}$, $y \in \mathcal{Y}$ be distributed according to π , and define random variables $\hat{y} := \arg\max_i \psi_i(\mathbf{z})$ and $\hat{p} := \psi_{\hat{y}}(\mathbf{z})$. We say ψ is *perfectly calibrated* with respect to π , if for any $p \in [0, 1]$, it satisfies $\text{Prob}(\hat{y} = y | \hat{p} = p) = p$.

Note that \mathbf{z} , y , \hat{y} and \hat{p} are correlated random variables. Therefore, Definition 1 essentially means that, if ψ is perfectly calibrated, then for any $p \in [0, 1]$, the true label y and the predicted label \hat{y} match, with a probability exactly p in the events where \mathbf{z} satisfies $\max_i \psi_i(\mathbf{z}) = p$.

In practice, learning a perfectly calibrated predictor is unrealistic, so we need a way to measure the calibration error. A common calibration metric is called Expected Calibration Error (ECE) [Naeini et al., 2015]: $\text{ECE} = \sum_{m=1}^M \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)|$. This equation is calculated in two steps: First the confidence scores of samples in \mathcal{D}_c are partitioned into M equally spaced bins $\{B_m\}_{m=1}^M$. Second the weighted average of the differences between the average confidence $\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}^i$ and the accuracy $\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(y^i = \hat{y}^i)$ in each bin is computed as the ECE metric, where $|B_m|$ denotes the size of bin B_m , $\mathbb{1}$ is the indicator function, and the superscript i indexes the sampled random variable. In addition to ECE, other calibration metrics have also been proposed [Guo et al., 2017; Nixon et al., 2019; Brier, 1950; Kumar et al., 2019]; e.g., Classwise-ECE [Kull et al., 2019] and Brier score [Brier, 1950] are proposed as measures of classwise-calibration. All the metrics for measuring calibration have their own pros and cons. Here, we consider the most commonly used metrics for measuring calibration and leave their analysis for future work.

¹The softmax requirement is not an assumption but for making the notation consistent with the literature. The proposed algorithm can also be applied to the output of general probabilistic predictors.

While the calibration metrics above measure the deviation from perfect calibration in Definition 1, they are usually not suitable loss functions for optimizing neural networks, e.g., due to the lack of continuity or non-trivial computation time. Instead, the calibration function \mathbf{f} in $\phi = \sigma_{\text{SM}} \circ \mathbf{f} \circ \mathbf{g}$ is often optimized indirectly through a surrogate loss function (e.g. the negative log-likelihood) defined on the held-out calibration dataset \mathcal{D}_c [Guo et al., 2017].

6.2.1 Importance of Inductive Bias

Unlike regular deep learning scenarios, here the calibration dataset \mathcal{D}_c is relatively small. Therefore, controlling the capacity of the hypothesis space of \mathbf{f} becomes a crucial topic [Guo et al., 2017; Kull et al., 2017a, 2019]. There is typically a trade-off between preserving accuracy and improving calibration: Learning \mathbf{f} could improve the calibration performance, but it could also change the decision boundary of ϕ from ϕ_0 decreasing the accuracy. While using simple calibration functions may be applicable when ϕ_0 has a simple function landscape or is already close to being well calibrated, such a function class might not be sufficient to calibrate modern deep networks with complex decision boundaries as we will show in the experiments in Section 6.5.

The observation above motivates us to investigate the possibility of learning calibration functions within a hypothesis space that can provably guarantee preserving the accuracy of the original network ϕ_0 . The identification of such functions would address the previous dilemma and give precisely the needed structure to ensure generalization of calibration when the calibration dataset \mathcal{D}_c is small.

6.3 Intra Order-Preserving Functions

In this section, we formally describe this desirable class of functions for post-hoc network calibration. We name them *intra order-preserving functions*. Learning within this family is both necessary and sufficient to keep the top- k accuracy of the original network unchanged. We also study additional function structures on this family (e.g. limiting how different dimensions can interact), which can be used as regularization in learning calibration functions. Last, we discuss a new neural network architecture for representing these functions.

6.3.1 Setup: Sorting and Ranking

We begin by defining sorting functions and ranking in preparation for the formal definition of intra order-preserving functions. Let $\mathbb{P}^n \subset \{0,1\}^{n \times n}$ denote the set of $n \times n$ permutation matrices. Sorting can be viewed as a permutation matrix; Given a vector $\mathbf{x} \in \mathbb{R}^n$, we say $S : \mathbb{R}^n \rightarrow \mathbb{P}^n$ is a *sorting function* if $\mathbf{y} = S(\mathbf{x})\mathbf{x}$ satisfies $y_1 \geq y_2 \geq \dots \geq y_n$. In case there are ties in the input vector \mathbf{x} , the sorting matrix can not be uniquely defined. To resolve this, we use a pre-defined *tie breaker* vector which is used as a tie breaking protocol. We say a vector $\mathbf{t} \in \mathbb{R}^n$ is a tie breaker if $\mathbf{t} = P\mathbf{r}$, for some $P \in \mathbb{P}^n$, where $\mathbf{r} = [1, \dots, n]^\top \in \mathbb{R}^n$. Tie breaker pre-assigns priorities to

indices of the input vector and is used to resolve ties. For instance, $S_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $S_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ are the unique sorting matrices of input $\mathbf{x} = [0, 0]^\top$ with respect to tie breaker $\mathbf{t}_1 = [1, 2]^\top$ and $\mathbf{t}_2 = [2, 1]^\top$, respectively. We say two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ share the same ranking if $S(\mathbf{u}) = S(\mathbf{v})$ for any tie breaker \mathbf{t} .

6.3.2 Intra Order-Preserving Functions

We define the *intra* order-preserving property with respect to different coordinates of a vector input.

Definition 2. We say a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is *intra order-preserving*, if, for any $\mathbf{x} \in \mathbb{R}^n$, both \mathbf{x} and $\mathbf{f}(\mathbf{x})$ share the same ranking.

The output of an intra order-preserving function $\mathbf{f}(\mathbf{x})$ maintains *all* ties and strict inequalities between elements of the input vector \mathbf{x} . Namely, for all $i, j \in \llbracket n \rrbracket$, we have $x_i > x_j$ (or $x_i = x_j$) if and only if $\mathbf{f}_i(\mathbf{x}) > \mathbf{f}_j(\mathbf{x})$ (or $\mathbf{f}_i(\mathbf{x}) = \mathbf{f}_j(\mathbf{x})$). For example, a simple intra order-preserving function is the temperature scaling $\mathbf{f}(\mathbf{x}) = \mathbf{x}/t$ for some $t > 0$. Another common instance is the softmax operator.

Clearly, applying an intra order-preserving function as the calibration function in $\phi = \sigma_{\text{SM}} \circ \mathbf{f} \circ \mathbf{g}$ does not change top- k predictions between ϕ and $\phi_o = \sigma_{\text{SM}} \circ \mathbf{g}$.

Next, we provide a necessary and sufficient condition for constructing continuous, intra order-invariant functions. This theorem will be later used to design neural network architectures for learning calibration functions. Note that for a vector $\mathbf{v} \in \mathbb{R}^n$ and an upper-triangular matrix of ones U , $U\mathbf{v}$ is the *reverse* cumulative sum of \mathbf{v} (i.e. $(U\mathbf{v})_i = \sum_{j=i}^n \mathbf{v}_j$).

Theorem 1. A continuous function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is *intra order-preserving*, if and only if $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}U\mathbf{w}(\mathbf{x})$ with U being an upper-triangular matrix of ones and $\mathbf{w} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ being a continuous function such that

- $\mathbf{w}_i(\mathbf{x}) = 0$, if $\mathbf{y}_i = \mathbf{y}_{i+1}$ and $i < n$,
- $\mathbf{w}_i(\mathbf{x}) > 0$, if $\mathbf{y}_i > \mathbf{y}_{i+1}$ and $i < n$,
- $\mathbf{w}_n(\mathbf{x})$ is arbitrary,

where $\mathbf{y} = S(\mathbf{x})\mathbf{x}$ is the sorted version of \mathbf{x} .

The proof is deferred to Appendix. Here we provide as sketch as to why Theorem 1 is true. Since $\mathbf{w}_i(\mathbf{x}) \geq 0$ for $i < n$, applying the matrix U on $\mathbf{w}(\mathbf{x})$ results in a sorted vector $U\mathbf{w}(\mathbf{x})$. Thus, applying $S(\mathbf{x})^{-1}$ further on $U\mathbf{w}(\mathbf{x})$ makes sure that $\mathbf{f}(\mathbf{x})$ has the same ordering as the input vector \mathbf{x} . The reverse direction can be proved similarly. For the continuity, observe that the sorting function $S(\mathbf{x})$ is piece-wise constant with discontinuities only when there is a tie in the input \mathbf{x} . This means that if the corresponding elements in $U\mathbf{w}(\mathbf{x})$ are also equally valued when a tie happens, the discontinuity of the sorting function S does not affect the continuity of \mathbf{f} inherited from \mathbf{w} .

6.3.3 Order-Invariant and Diagonal Sub-families

Different classes in a classification task typically have shared characteristics. Therefore, calibration functions sharing properties across different classes can work as a suitable inductive bias in learning. Here we use this idea to define two additional structures interesting to intra order-preserving functions: *order-invariant* and *diagonal* properties. Similar to the purpose of the previous section, we will study necessary and sufficient conditions for functions with these properties.

First, we study the concept of order-invariant functions.

Definition 3. We say a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is *order-invariant*², if $\mathbf{f}(P\mathbf{x}) = P\mathbf{f}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$ and permutation matrices $P \in \mathbb{P}^n$.

For an order-invariant function \mathbf{f} , when two elements \mathbf{x}_i and \mathbf{x}_j in the input \mathbf{x} are swapped, the corresponding elements $\mathbf{f}_i(\mathbf{x})$ and $\mathbf{f}_j(\mathbf{x})$ in the output $\mathbf{f}(\mathbf{x})$ are also swapped. In this way, the mapping learned for the i th class can also be used for the j th class. Thus, the order-invariant family shares the calibration function between different classes while allowing the output of each class be a function of all other class predictions.

We characterize in the theorem below the properties of functions that are both intra order-preserving and order-invariant (an instance is the softmax operator). It shows that, to make an intra order-preserving function also order-invariant, we just need to feed the function \mathbf{w} in Theorem 1 with the sorted input $\mathbf{y} = S(\mathbf{x})\mathbf{x}$ instead of \mathbf{x} . This scheme makes the learning of \mathbf{w} easier since it always sees sorted vectors (which are a subset of \mathbb{R}^n).

Theorem 2. A continuous, intra order-preserving function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is order-invariant, if and only if $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}U\mathbf{w}(\mathbf{y})$, where U , \mathbf{w} , and \mathbf{y} are in Theorem 1.

Below, we provide an example to illustrate the difference between the order-invariant subfamily and the general intra order-preserving functions.

Example 1. Consider two 3-class logit vectors $\mathbf{x}_1 = [1, 2, 3]^\top$ and $\mathbf{x}_2 = [2, 3, 1]^\top$. An order-preserving function $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ can have the outputs $f(\mathbf{x}_1) = [4, 5, 6]^\top$ and $f(\mathbf{x}_2) = [8, 9, 7]^\top$. An intra order-preserving and order-invariant function $g : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ only receives sorted inputs. So, it can not have the vectors $[4, 5, 6]^\top$ and $[8, 9, 7]^\top$ as output. If $g(\mathbf{x}_1) = [4, 5, 6]^\top$, the output of g on \mathbf{x}_2 should be $g(\mathbf{x}_2) = [5, 6, 4]^\top$.

Another structure of interest here is the diagonal property.

Definition 4. We say a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is *diagonal*, if $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}_1), \dots, f_n(\mathbf{x}_n)]$ for $f_i : \mathbb{R} \rightarrow \mathbb{R}$ with $i \in \llbracket n \rrbracket$.

In the context of calibration, a diagonal calibration function means that different class predictions do not interact with each other in \mathbf{f} . Defining diagonal family is mostly motivated by the success of temperature scaling method [Guo et al., 2017],

²This function is also called permutation equivariant.

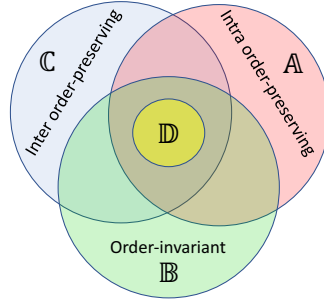


Figure 6.2: Relationship between different function families. Theorem 1 specifies the intra order-preserving functions \mathcal{A} . Theorem 2 specifies the intra order-preserving and order-invariant functions $\mathcal{A} \cap \mathcal{B}$. Theorem 3 specifies the diagonal intra order-preserving functions \mathcal{D} . By Corollary 1, these functions are also order-invariant and inter order-preserving i.e. $\mathcal{D} \subseteq \mathcal{A} \cap \mathcal{B} \cap \mathcal{C}$.

which is a linear diagonal intra order-preserving function. Therefore, although diagonal intra order-preserving functions may sound limiting in learning calibration functions, they still represent a useful class of functions.

The next theorem relates diagonal intra order-preserving functions to increasing functions.

Theorem 3. A continuous, intra order-preserving function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is diagonal, if and only if $\mathbf{f}(\mathbf{x}) = [\bar{f}(x_1), \dots, \bar{f}(x_n)]$ for some continuous and increasing function $\bar{f} : \mathbb{R} \rightarrow \mathbb{R}$.

Compared with general diagonal functions, diagonal intra order-preserving automatically implies that the same function \bar{f} is shared across all dimensions. Thus, learning with diagonal intra order-preserving functions benefits from parameter-sharing across different dimensions, which could drastically decrease the number of parameters.

Finally, below we show that functions in this sub-family are also order-invariant and inter order-preserving. Note that inter and intra order-preserving are orthogonal definitions. Inter order-preserving is also an important property for calibration functions, since this property guarantees that $\mathbf{f}_i(\mathbf{x})$ increases with the original class logit x_i . The set diagram in Figure 6.2 depicts the relationship among different intra order-preserving families.

Definition 5. We say a function $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is *inter order-preserving* if, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{x} \geq \mathbf{y}$, $\mathbf{f}(\mathbf{x}) \geq \mathbf{f}(\mathbf{y})$, where \geq denotes elementwise comparison.

Corollary 1. A diagonal, intra order-preserving function is order-invariant and inter order-preserving

6.3.4 Practical Considerations

Theorems 1 and 2 describe general representations of intra order-preserving functions through a function \mathbf{w} that satisfies certain non-negative constraints. Inspired

by these theoretical results, we propose a neural network architecture, Figure 6.3, to represent exactly a family of intra order-preserving functions.

The main idea in Figure 6.3 is to parameterize \mathbf{w} through a composition of smaller functions. For $i < n$, we set $\mathbf{w}_i(\mathbf{x}) = \sigma(\mathbf{y}_i - \mathbf{y}_{i+1})\mathbf{m}_i(\mathbf{x})$, where $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is a positive function such that $\sigma(a) = 0$ only when $a = 0$, and \mathbf{m}_i is a strictly positive function. It is easy to verify that this parameterization of \mathbf{w} satisfies the requirements on \mathbf{w} in Theorem 1. However, we note that this class of functions cannot represent all possible \mathbf{w} stated in Theorem 1. In general, the speed $\mathbf{w}_i(\mathbf{x})$ converges to 0 can be a function of \mathbf{x} , but in the proposed factorization above, the rate of convergence to zero is a function of only two elements \mathbf{y}_i and \mathbf{y}_{i+1} . Fortunately, such a limitation does not substantially decrease the expressiveness of \mathbf{f} in practice, because the subspace where \mathbf{w}_i vanishes has zero measure in \mathbb{R}^n (i.e. subspaces where there is at least one tie in $\mathbf{x} \in \mathbb{R}^n$).

By Theorem 1 and Theorem 2, the proposed architecture in Figure 6.3 ensures $\mathbf{f}(\mathbf{x})$ is continuous in \mathbf{x} as long as $\sigma(\mathbf{y}_i - \mathbf{y}_{i+1})$ and $\mathbf{m}_i(\mathbf{x})$ are continuous in \mathbf{x} . In the appendix, we show that this is true when σ and \mathbf{m}_i are continuous functions. Additionally, we prove that when σ and \mathbf{m} are continuously differentiable, $\mathbf{f}(\mathbf{x})$ is also directionally differentiable with respect to \mathbf{x} . Note that the differentiability to the input is not a requirement to learn the parameters of \mathbf{m} with a first order optimization algorithm which only needs \mathbf{f} to be differentiable with respect to the parameters of \mathbf{m} . The latter condition holds in general, since the only potential sources of non-differentiable \mathbf{f} , $S(\mathbf{x})^{-1}$ and \mathbf{y} are constant with respect to the parameters of \mathbf{m} . Thus, if \mathbf{m} is differentiable with respect to its parameters, \mathbf{f} is also differentiable with respect to the parameters of \mathbf{m} .

6.4 Implementation

Given a calibration dataset $\mathcal{D}_c = \{(\mathbf{z}^i, y^i)\}_{i=1}^N$ and a calibration function \mathbf{f} parameterized by some vector $\boldsymbol{\theta}$, we define the empirical calibration loss as $\frac{1}{N} \sum_{i=1}^N \ell(y^i, \mathbf{f}(\mathbf{x}^i)) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2$, where $\mathbf{x}^i = \mathbf{g}(\mathbf{z}^i)$, $\ell : \mathcal{Y} \times \mathbb{R}^n \rightarrow \mathbb{R}$ is a classification cost function, and $\lambda \geq 0$ is the regularization weight. Here we follow the calibration literature [Thulasidasan et al., 2019; Guo et al., 2017; Kull et al., 2019] and use the negative log likelihood (NLL) loss, i.e., $\ell(y, \mathbf{f}(\mathbf{x})) = -\log(\sigma_{\text{SM}}(\mathbf{f}(\mathbf{x}))_y)$, where σ_{SM} is the softmax operator and $\sigma_{\text{SM}}(\mathbf{x})_y$ is its y th element. We use the NLL loss in all the experiments to study the benefit of learning \mathbf{f} with different structures. The study of other loss functions for calibration [Seo et al., 2019; Xing et al., 2020] is outside the scope of this thesis.

To ensure \mathbf{f} is within the intra order-preserving family, we restrict \mathbf{f} to have the structure in Theorem 1 and set $\mathbf{w}_i(\mathbf{x}) = \sigma(\mathbf{y}_i - \mathbf{y}_{i+1})\mathbf{m}_i(\mathbf{x})$, as described in Section 6.3.4. We parameterize function \mathbf{m} by a generic multi-layer neural network and utilize the softplus activation $s^+(a) = \log(1 + \exp(a))$ on the last layer when strict positivity is desired and represent σ as $\sigma(a) = |a|$. For example, when $\mathbf{m}_i(\mathbf{x})$ is constant, our architecture recovers the temperature scaling scheme [Guo et al., 2017].

The order-invariant version in Theorem 2 can be constructed similarly. The only

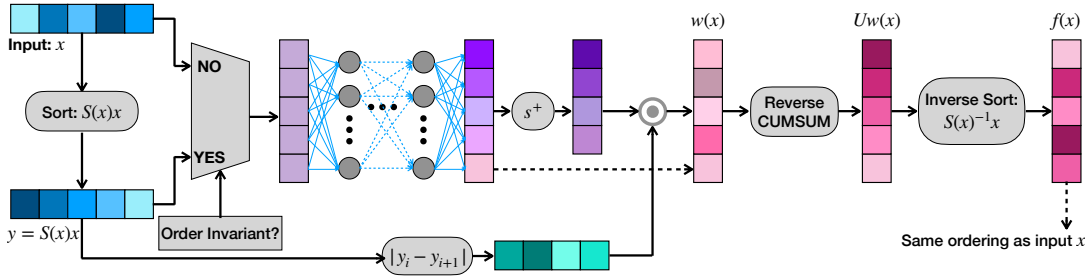


Figure 6.3: Flow graph of the intra order-preserving function. The vector $\mathbf{x} \in \mathbb{R}^n$ is the input to the graph. Function \mathbf{m} is estimated using a generic multi-layer neural network with non-linear activation for the hidden layers. The input to the network is sorted for learning order-preserving functions. We employ softplus activation function s^+ to impose strict positivity constraints.

difference is that the neural network that parameterizes \mathbf{m} receives instead the sorted input. Figure 6.3 illustrates the architecture of these models.

The diagonal intra order-preserving version in Theorem 3 is formed by learning an increasing function shared across all logit dimensions. We use the official implementation of proposed architecture in [Wehenkel and Louppe, 2019] that learns monotonic functions with unconstrained neural networks. The idea is to learn an increasing function $\bar{f}(x) : \mathbb{R} \rightarrow \mathbb{R}$ using a neural network, which can be realized by learning a strictly positive function $\bar{f}'(x)$ and a bias $\bar{f}(0) \in \mathbb{R}$ and constructing the desired function \bar{f} by the integral $\bar{f}(x) = \int_0^x \bar{f}'(t)dt + \bar{f}(0)$. In implementation, the derivative \bar{f}' is modeled by a generic neural network and the positiveness is enforced by using a proper activation function in the last layer. In the forward computation, the integral is approximated numerically using Clenshaw-Curtis quadrature [Clenshaw and Curtis, 1960] and the backward pass is performed by Leibniz integral rule to reduce memory footprint.

We note that, similar to our work, the concurrent work in [Zhang et al., 2020] also give special attention to order preserving transformations for calibration. However, their introduced functions are less expressive than the ones presented in this work.

6.5 Experiments

We evaluate the performance of intra order-preserving (OP), order-invariant intra order-preserving (OI), and diagonal intra order-preserving (DIAG) families in calibrating the output of various image classification deep networks and compare their results with the previous post-hoc calibration techniques.

Datasets. We use six different datasets: CIFAR-10,100 [Krizhevsky et al., 2009], SVHN [Netzer et al., 2011], CARS [Krause et al., 2013], BIRDS [Welinder et al., 2010], and ImageNet [Deng et al., 2009]. In these datasets, the number of classes vary from 10 to 1000. We evaluate the performance of different post-hoc calibration methods to calibrate ResNet [He et al., 2016], Wide ResNet [Zagoruyko and Komodakis, 2016],

DenseNet [Huang et al., 2017a], and PNASNet5 [Liu et al., 2018] networks. The size of the calibration and the test datasets, as well as the number of classes for each dataset, are shown in Table 6.1. We note that the calibration sets sizes are the same as the previous methods [Guo et al., 2017; Kull et al., 2019].

We follow the experiment protocol in [Kull et al., 2019] and use cross validation on the calibration dataset to find the best hyperparameters and architectures for all the methods. We found that [Kull et al., 2019] have improved their performance via averaging output predictions of models trained on different folds. We follow the same approach to have fair comparisons. Our criteria for selecting the best architecture is the NLL value. We perform 3 fold cross validation for ImageNet and 5 folds for all the other datasets. We limit our architecture to fully connected networks and vary the number of hidden layers as well as the size of each layer. We allow networks with up to 3 hidden layers in all the experiments. In CIFAR-10, SVHN, and CIFAR-100 with fewer classes, we test networks with $\{1, 2, 10, 20, 50, 100, 150\}$ units per layer and for the larger CARS, BIRDS, and ImageNet datasets, we allow a wider range of $\{2, 10, 20, 50, 100, 150, 500\}$ units per layer. We use the similar number of units for all the hidden layers to reduce the search space. We use ReLU activation for all middle hidden layers and Softplus on the last layer when strict positivity is desired. We utilize L-BFGS [Liu and Nocedal, 1989] for small scale optimization problems when the computational resources allow (temperature scaling and diagonal intra order-preserving (DIAG) methods on CIFAR and SVHN datasets) and use Adam [Kingma and Ba, 2014] optimizer for other experiments. Table 6.2 summarize cross validation learned hyperparameter for each method.

Although the functions learned in Table 6.2 are more complicated than linear transformations used in the baselines, they are not too complex to slow down computation as the calibration network size is negligible compared to the backbone network used in the experiments. In our experiments, all methods take less than 0.5 milliseconds/sample in forward path and their differences are negligible.

We use the pre-computed logits of these networks provided by [Kull et al., 2019] for CIFAR, SVHN, and ImageNet with DenseNet and ResNet³. In addition, we use the publicly available state-of-the-art models for PNASNet5-large and ResNet50

³https://github.com/markus93/NN_Calibration

Table 6.1: Statistics of the Evaluation Datasets.

Dataset	#classes	Calibration set size	Test dataset size
CIFAR-10	10	5000	10000
SVHN	10	6000	26032
CIFAR-100	100	5000	10000
CARS	196	4020	4020
BIRDS	200	2897	2897
ImageNet	1000	25000	25000

Table 6.2: Hyperparameters learned by cross validation. For DIAG, OI, OP, and UNCONSTRAINED we show the network architectures learned by cross validation. The number of units in each layer are represented by a sequence of numbers, e.g. (10, 20, 30, 40) represents a network with 10 input units, 20 and 30 units in the first and second hidden layers, respectively, and 40 output units. We perform multi-fold cross-validation and select the architecture with lowest NLL on validation set.

Dataset	Model	DIAG	OI	OP	UNCONSTRAINED
CIFAR10	ResNet 110	(1,10,10,1)	(10,150,150,10)	(10,2,2,10)	(10,500,10)
CIFAR10	Wide ResNet 32	(1,2,2,1)	(10,10,10,10)	(10,2,2,10)	(10,150,150,10)
CIFAR10	DenseNet 40	(1,2,2,1)	(10,50,50,100)	(10,2,2,10)	(10,150,150,10)
SVHN	ResNet 152 (SD)	(1,20,20,1)	(10,10,10,10)	(10,50,50,10)	(10,500,10)
CIFAR100	ResNet 110	(1,10,10,1)	(100,100,100,100)	(100,150,150,100)	(100,500,100)
CIFAR100	Wide ResNet 32	(1,1,1)	(100,100,100,100)	(100,2,2,100)	(100,500,100)
CIFAR100	DenseNet 40	(1,1,1)	(100,10,10,100)	(100,2,2,100)	(100,500,500,100)
CARS	ResNet 50 (pre)	(1,50,1)	(196,10,196)	(196,2,2,196)	(196,500,196)
CARS	ResNet 101 (pre)	(1,20,20,1)	(196,100,100,196)	(196,20,20,196)	(196,500,196)
CARS	ResNet 101	(1,50,1)	(196,50,50,196)	(196,100,100,196)	(196,500,196)
BIRDS	ResNet 50 (NTS)	(1,50,50,1)	(200,150,150,200)	(200,50,50,200)	(200,500,200)
ImageNet	ResNet 152	(1,10,10,1)	(1000,150,150,1000)	(1000,2,2,1000)	(1000,150,1000)
ImageNet	DenseNet 161	(1,10,1)	(1000,100,100,1000)	(1000,2,2,1000)	(1000,150,1000)
ImageNet	PNASNet5 large	(1,20,20,1)	(1000,50,50,1000)	(1000,100,100,1000)	(1000,100,1000)

NTSNet [Yang et al., 2018]⁴ for ImageNet and BIRDS datasets, respectively. Furthermore, we trained different ResNet type models on CARS dataset using the standard pytorch training script. The ResNet models with (pre) are initialized with pre-trained ImageNet weights. The logits and the source code of our method is released at <https://github.com/AmirooR/IntraOrderPreservingCalibration>.

The effect of weight regularization on different metrics for MS and DIR methods is illustrated in Figure 6.4. This shows that simply regularizing the off diagonal elements of a linear layer has limited expressiveness to achieve good calibration especially in the case that number of classes is large.

Baselines. We compare the proposed function structures with temperature scaling (TS) [Guo et al., 2017], Dirichlet calibration with off-diagonal regularization (DIR) [Kull et al., 2019], and matrix scaling with off-diagonal regularization (MS) [Kull et al., 2019] as they are the current best performing post-hoc calibration methods. We also present the results of the original uncalibrated models (Uncal.) for comparison. To show the effect of intra order-preserving regularization, we also show the results of applying unconstrained multi-layer neural network without intra order-preserving constraint (UNCONSTRAINED). In cross-validation, we tune the architecture as well as regularization weight of UNCONSTRAINED and order-preserving functions. As we are using the same logits as [Kull et al., 2019], we report their results directly on CIFAR-10, CIFAR-100, and SVHN. However, since they do not present the results for CARS, BIRDS, and ImageNet datasets, we report the results of their official implementation⁵ on these datasets.

⁴<https://github.com/osmr/imgclsmob/blob/master/pytorch/README.md>

⁵https://github.com/dirichletcal/experiments_dnn/

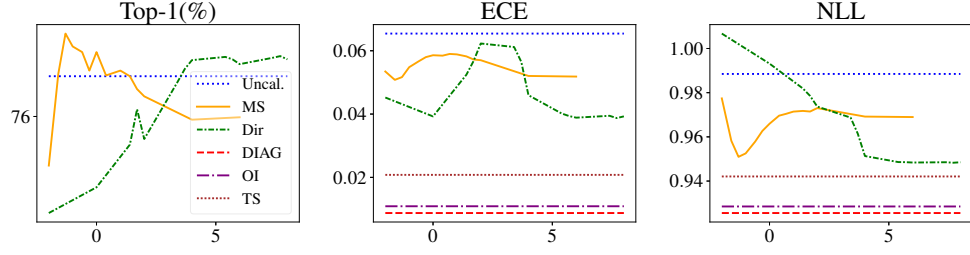


Figure 6.4: Accuracy, ECE, and NLL plots in MS and DIR for ResNet 152 on ImageNet with different regularization weights. In the plots, x-axis shows the log scale regularization and y-axis shows the accuracy, ECE, and NLL of different methods, respectively. The value of the bias regularizer is found by cross validation and kept constant for visualization purpose. Changing the bias regularizer has little effect on the final shape of the plots.

Table 6.3: ECE (with $M = 15$ bins) on various image classification datasets and models with different calibration methods. The subscript numbers represent the rank of the corresponding method on the given model/dataset. The accuracy of the uncalibrated model is shown in parentheses. The number in parentheses in DIR, MS, and UNCONSTRAINED methods show the change in accuracy for each method.

Dataset	Model	Uncal.	TS	DIR	MS	DIAG	OI	OP	UNCONSTRAINED
CIFAR10	ResNet 110	0.0475 ₈ (93.6%)	0.0113 ₅	0.0109 ₄ (−0.1%)	0.0106 ₃ (−0.1%)	0.0067 ₂	0.0061 ₁	0.0119 ₆	0.0170 ₇ (−0.4%)
CIFAR10	Wide ResNet 32	0.0451 ₈ (93.9%)	0.0078 ₄	0.0084 ₅ (+0.3%)	0.0073 ₂ (+0.3%)	0.0136 ₇	0.0064 ₁	0.0077 ₃	0.0097 ₆ (−0.1%)
CIFAR10	DenseNet 40	0.0550 ₈ (92.4%)	0.0095 ₂	0.0110 ₄ (+0.1%)	0.0099 ₃ (+0.1%)	0.0069 ₁	0.0116 ₅	0.0128 ₇	0.0125 ₆ (−0.5%)
SVHN	ResNet 152 (SD)	0.0086 ₈ (98.1%)	0.0061 ₅	0.0058 ₃ (+0.0%)	0.0060 ₄ (+0.0%)	0.0057 ₂	0.0116 ₆	0.0118 ₇	0.0015 ₁ (+0.0%)
CIFAR100	ResNet 110	0.1848 ₈ (71.5%)	0.0238 ₂	0.0282 ₅ (+0.2%)	0.0274 ₄ (+0.1%)	0.0507 ₇	0.0119 ₁	0.0253 ₃	0.0346 ₆ (−4.4%)
CIFAR100	Wide ResNet 32	0.1878 ₈ (73.8%)	0.0147 ₂	0.0189 ₅ (+0.1%)	0.0258 ₆ (+0.1%)	0.0172 ₃	0.0126 ₁	0.0173 ₄	0.0421 ₇ (−6.1%)
CIFAR100	DenseNet 40	0.2116 ₈ (70.0%)	0.0090 ₂	0.0114 ₄ (+0.1%)	0.0220 ₆ (+0.4%)	0.0075 ₁	0.0098 ₃	0.0154 ₅	0.0990 ₇ (−12.9%)
CARS	ResNet 50 (pre)	0.0239 ₇ (91.3%)	0.0144 ₃	0.0243 ₈ (+0.2%)	0.0186 ₆ (−0.3%)	0.0105 ₂	0.0103 ₁	0.0185 ₅	0.0182 ₄ (−3.5%)
CARS	ResNet 101 (pre)	0.0218 ₇ (92.2%)	0.0165 ₅	0.0225 ₈ (+0.0%)	0.0191 ₆ (−0.8%)	0.0102 ₁	0.0135 ₃	0.0125 ₂	0.0155 ₄ (−3.9%)
CARS	ResNet 101	0.0421 ₈ (85.2%)	0.0301 ₄	0.0245 ₃ (−0.3%)	0.0345 ₆ (−1.1%)	0.0206 ₁	0.0323 ₅	0.0358 ₇	0.0236 ₂ (−7.0%)
BIRDS	ResNet 50 (NTS)	0.0714 ₈ (87.4%)	0.0319 ₅	0.0486 ₆ (−0.2%)	0.0585 ₇ (−1.1%)	0.0188 ₂	0.0172 ₁	0.0292 ₄	0.0276 ₃ (−2.2%)
ImageNet	ResNet 152	0.0654 ₇ (76.2%)	0.0208 ₄	0.0452 ₅ (+0.1%)	0.0567 ₆ (+0.1%)	0.0087 ₁	0.0109 ₂	0.0167 ₃	0.1297 ₈ (−33.4%)
ImageNet	DenseNet 161	0.0572 ₇ (77.1%)	0.0198 ₄	0.0374 ₅ (+0.1%)	0.0443 ₆ (+0.4%)	0.0103 ₁	0.0123 ₂	0.0168 ₃	0.1380 ₈ (−28.1%)
ImageNet	PNASNet5 large	0.0610 ₇ (83.1%)	0.0713 ₈	0.0398 ₆ (+0.0%)	0.0217 ₄ (+0.3%)	0.0117 ₂	0.0084 ₁	0.0133 ₃	0.0316 ₅ (−4.8%)
Average Relative Error		1.00 ₈	0.42 ₄	0.49 ₅	0.50 ₆	0.27 ₁	0.33 ₂	0.41 ₃	0.66 ₇

6.5.1 Results

Table 6.3 summarizes the results of our calibration methods and other baselines in terms of ECE and presents the average relative error with respect to the uncalibrated model. Overall, DIAG has the lowest average relative error followed by OI among the models and datasets presented in Table 6.3. OI is the best-performing method in 7 out of 14 experiments including ResNet 110 and Wide ResNet 32 models on CIFAR datasets as well as state-of-the-art PNASNet5 large model. DIAG family’s relative average error is half the MS and DIR methods and 15% less compared to Temp. Scaling. Although DIR and MS were able to maintain the accuracy of the original models in most of the cases by imposing off diagonal regularization, order-preserving family could significantly outperform them regarding the ECE metric. Finally, we remark that learning an unconstrained multi-layer neural network does

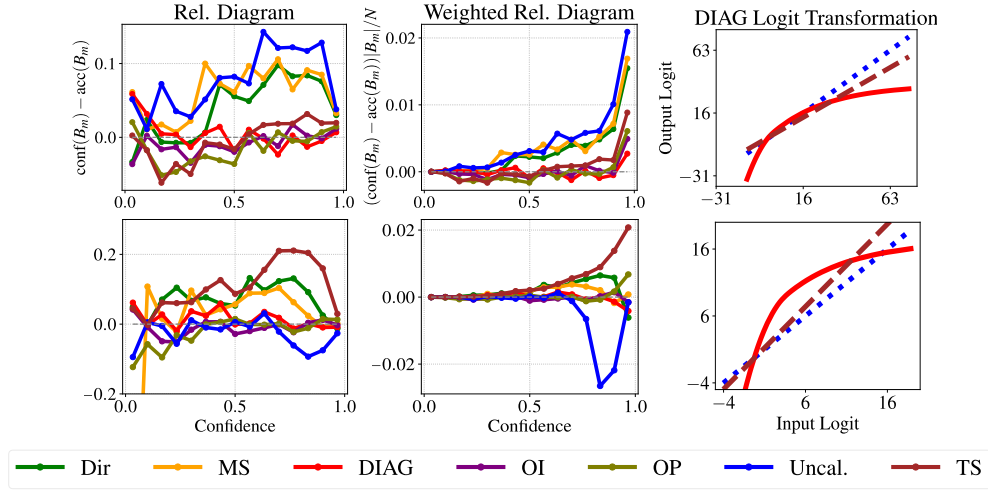


Figure 6.5: Performance evaluations of ResNet 152 (**Top Row**) and PNASNet5 large (**Bottom Row**) on ImageNet dataset. (**Left**) Reliability diagrams. As suggested by [Maddox et al., 2019] we show the difference between the estimated confidence and accuracy over $M = 15$ bins. The dashed grey lines represent the perfectly calibrated network at $y = 0$. Points above (below) the grey line show overconfident (underconfident) predictions in a bin. (**Middle**) Weighted reliability diagrams where bin values are weighted by data frequency distribution. Since the uncalibrated network has different distances to the perfect calibration in different bins, scaling by a single temperature will lead to a mix of underconfident and overconfident regions. Our order-preserving functions, on the other hand, have more flexibility to reduce the calibration error. (**Right**) Transformation learned by DIAG function compared to temperature scaling and uncalibrated model (identity map).

not exhibit a good calibration performance and drastically hurts the accuracy in some datasets as shown in the last column of Table 6.3.

Figure 6.5 illustrates the reliability diagrams of models trained on ResNet 152 (top row) and PNASNet5 large (bottom row). Weighted reliability diagrams are also presented to better indicate the differences regarding the ECE metric. Surprisingly, these diagrams show that the uncalibrated PNASNet5 large model is underconfident. The differences between the mappings learned by DIAG and temperature scaling on these models are illustrated on the right column. DIAG is capable of learning complex increasing functions while temperature scaling only scales all the logits. Compared with DIR and MS which learn a linear transformation, all intra order-preserving methods can learn non-linear transformations on the logits while decoupling accuracy from calibration of the predictions.

In addition to ECE, which considers the top prediction, we also measure the NLL, Marginal Calibration Error [Kumar et al., 2019], Classwise-ECE, and Berier score. As it is shown in Table 6.4, DIAG and OI have the best overall performance in terms of average relative error in most cases, while DIR is the top performing method in Classwise-ECE.

Table 6.4: Average relative error. Each entry shows the relative error compared to the uncalibrated model averaged over all the datasets. The subscripts represent the rank of the corresponding method on the given metric. See the Appendix for per dataset performance comparisons.

Evaluation Metric	Uncal.	TS	Dir	MS	DIAG	OI	OP
ECE	1.000 ₇	0.420 ₄	0.490 ₅	0.500 ₆	0.270 ₁	0.330 ₂	0.410 ₃
Debiased ECE [Kumar et al., 2019]	1.000 ₇	0.357 ₃	0.430 ₆	0.409 ₅	0.213 ₁	0.337 ₂	0.406 ₄
NLL	1.000 ₇	0.766 ₄	0.772 ₆	0.768 ₅	0.749 ₁	0.751 ₂	0.765 ₃
Marginal Calibration Error [Kumar et al., 2019]	1.000 ₇	0.750 ₃	0.735 ₂	0.996 ₆	0.725 ₁	0.778 ₄	0.898 ₅
Classwise-ECE	1.000 ₇	0.752 ₆	0.704 ₁	0.734 ₃	0.729 ₂	0.740 ₄	0.743 ₅
Brier	1.000 ₇	0.936 ₅	0.930 ₃	0.936 ₅	0.924 ₁	0.929 ₂	0.931 ₄

Table 6.5: Scores and rankings of different methods for Brier.

Dataset	Model	Uncal.	TS	Dir	MS	DIAG	OI	OP
CIFAR10	ResNet 110	0.01102 ₇	0.00979 ₆	0.00977 ₅	0.00976 ₄	0.00967 ₂	0.00963 ₁	0.00975 ₃
CIFAR10	Wide ResNet 32	0.01047 ₇	0.00924 ₄	0.00888 ₁	0.00889 ₂	0.00926 ₆	0.00921 ₃	0.00925 ₅
CIFAR10	DenseNet 40	0.01274 ₇	0.01100 ₄	0.01097 ₁	0.01097 ₁	0.01100 ₄	0.01110 ₆	0.01099 ₃
SVHN	ResNet 152 (SD)	0.00297 ₆	0.00291 ₁	0.00293 ₃	0.00298 ₇	0.00292 ₂	0.00293 ₃	0.00296 ₅
CIFAR100	ResNet 110	0.00453 ₇	0.00392 ₄	0.00391 ₃	0.00391 ₃	0.00393 ₅	0.00389 ₁	0.00390 ₂
CIFAR100	Wide ResNet 32	0.00432 ₇	0.00355 ₄	0.00354 ₂	0.00351 ₁	0.00355 ₄	0.00354 ₂	0.00355 ₄
CIFAR100	DenseNet 40	0.00491 ₇	0.00401 ₃	0.00400 ₁	0.00400 ₁	0.00401 ₃	0.00401 ₃	0.00402 ₆
CARS	ResNet 50 (pretrained)	0.000667 ₅	0.000666 ₄	0.000663 ₂	0.000679 ₇	0.000661 ₁	0.000664 ₃	0.000674 ₆
CARS	ResNet 101 (pretrained)	0.000626 ₆	0.000625 ₅	0.000623 ₃	0.000655 ₇	0.000622 ₂	0.000620 ₁	0.000623 ₃
CARS	ResNet 101	0.001131 ₆	0.001129 ₅	0.001123 ₃	0.001154 ₇	0.001118 ₁	0.001123 ₃	0.001119 ₂
BIRDS	ResNet 50 (NTSNet)	0.001035 ₆	0.000995 ₅	0.000988 ₄	0.001040 ₇	0.000977 ₃	0.000972 ₁	0.000974 ₂
ImageNet	ResNet 152	0.000338 ₇	0.000332 ₄	0.000333 ₆	0.000332 ₄	0.000329 ₁	0.000330 ₂	0.000331 ₃
ImageNet	DenseNet 161	0.000325 ₇	0.000321 ₄	0.000321 ₄	0.000318 ₁	0.000319 ₂	0.000320 ₃	0.000321 ₄
ImageNet	PNASNet5 large	0.000255 ₆	0.000261 ₇	0.000252 ₅	0.000247 ₃	0.000245 ₂	0.000244 ₁	0.000248 ₄
Average Relative Error		1.000 ₇	0.936 ₅	0.930 ₃	0.936 ₅	0.924 ₁	0.929 ₂	0.931 ₄

6.5.2 Ablation Studies and More Experiments

Calibration Set Size. In this experiment, we gradually increase the calibration set size from 10% to 100% of its original size to create smaller calibration subsets. Then, for each calibration subset, we train different post-hoc calibration methods and measure their accuracy, NLL, and ECE. The results are illustrated in Figure 6.6. In overall, the performance of non intra order-preserving methods, i.e. Dir and MS, are more sensitive to the size of the calibration set while intra order-preserving methods maintain the accuracy and are more stable in terms of NLL and ECE.

Brier Score, NLL, and Classwise-ECE. As shown in Table 6.5, our OI is the best method in 5 out of 14 models with respect to the Brier score. MS also wins in 4 models. However, it performs poorly on CARS and BIRDS datasets. Our DIAG has the best average relative error. Overall, both OI and DIAG perform well on this metric. The Dir is the third best method on this metric and is slightly worse than OI in average relative error.

Results of different methods regarding the NLL metric are shown in Table 6.6. MS is the best method when the number of classes is less than or equal to 100 on this metric. Its performance degrades as the number of classes grows. This is typically due to the excessive number of parameters introduced by this method. Surprisingly, TS is the best method in SVHN with ResNet 152 (SD) model but its performance is

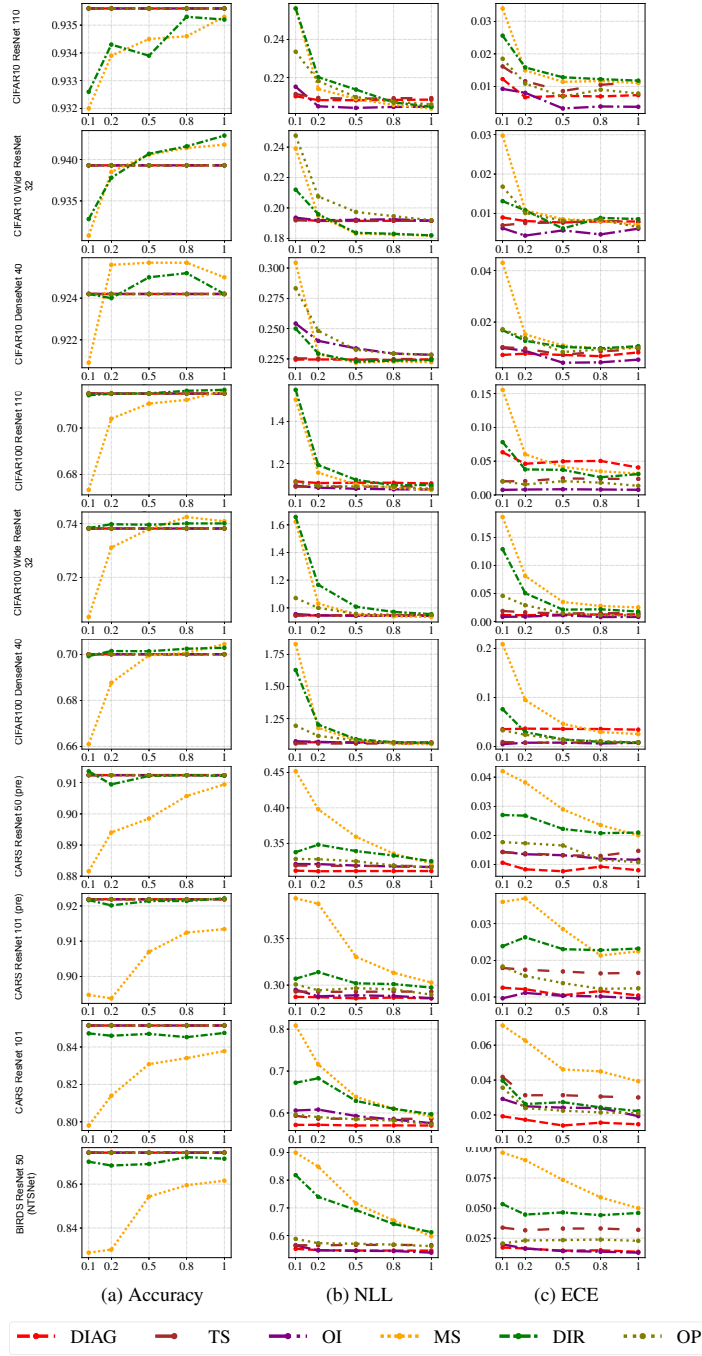


Figure 6.6: Accuracy, NLL, and ECE vs. calibration set size for CIFAR, CARS, BIRDS datasets. For each experiment, we use from 10% to 100% of the calibration set to train pos-hoc calibration functions and plot their accuracy, NLL, and ECE. Compared to DIR and MS, performance of the intra order-preserving methods (TS, DIAG, OI, and OP) degrades less with reducing the calibration set size.

Table 6.6: *NLL*.

Dataset	Model	Uncal.	TS	DIR	MS	DIAG	OI	OP
CIFAR10	ResNet 110	0.35827 ₇	0.20926 ₅	0.20511 ₃	0.20375 ₁	0.20674 ₄	0.20488 ₂	0.20954 ₆
CIFAR10	Wide ResNet 32	0.38170 ₇	0.19148 ₃	0.18203 ₂	0.18165 ₁	0.19221 ₅	0.19169 ₄	0.19332 ₆
CIFAR10	DenseNet 40	0.42821 ₇	0.22509 ₃	0.22371 ₂	0.22240 ₁	0.22551 ₄	0.23097 ₆	0.22798 ₅
SVHN	ResNet 152 (SD)	0.08542 ₇	0.07861 ₁	0.08038 ₅	0.08100 ₆	0.07887 ₂	0.07992 ₃	0.08010 ₄
CIFAR100	ResNet 110	1.69371 ₇	1.09169 ₄	1.09607 ₅	1.07370 ₁	1.10091 ₆	1.07966 ₂	1.08375 ₃
CIFAR100	Wide ResNet 32	1.80215 ₇	0.94453 ₃	0.95288 ₆	0.93273 ₁	0.94928 ₄	0.94312 ₂	0.95001 ₅
CIFAR100	DenseNet 40	2.01740 ₇	1.05713 ₂	1.05909 ₃	1.05084 ₁	1.05972 ₄	1.06127 ₅	1.07626 ₆
CARS	ResNet 50 (pretrained)	0.32993 ₇	0.31813 ₄	0.32381 ₆	0.31904 ₅	0.31234 ₁	0.31593 ₂	0.31793 ₃
CARS	ResNet 101 (pretrained)	0.30536 ₇	0.29329 ₃	0.29714 ₄	0.29788 ₅	0.28573 ₁	0.28897 ₂	0.30444 ₆
CARS	ResNet 101	0.61185 ₇	0.58619 ₄	0.59504 ₆	0.58683 ₅	0.57385 ₁	0.57774 ₂	0.58319 ₃
BIRDS	ResNet 50 (NTSNet)	0.74676 ₇	0.56569 ₄	0.61239 ₅	0.63055 ₆	0.54915 ₂	0.54508 ₁	0.56288 ₃
ImageNet	ResNet 152	0.98848 ₇	0.94208 ₄	0.95081 ₅	0.95786 ₆	0.92553 ₁	0.92850 ₂	0.93935 ₃
ImageNet	DenseNet 161	0.94395 ₇	0.90928 ₅	0.91214 ₆	0.90578 ₃	0.88937 ₁	0.89552 ₂	0.90632 ₄
ImageNet	PNASNet5 large	0.80240 ₇	0.75761 ₆	0.73955 ₅	0.71522 ₄	0.65550 ₁	0.65674 ₂	0.69595 ₃
Average Relative Error		1.000 ₇	0.766 ₄	0.772 ₆	0.768 ₅	0.749 ₁	0.751 ₂	0.765 ₃

very similar to the DIAG. The reason is that this model has a very high accuracy and the original model is actually already well calibrated. So, the single parameter TS would be enough to improve the calibration slightly. Our DIAG is the best method on datasets with larger number of classes and our OI is also comparable to it. Both these method have the best average ranking and DIAG has the best relative error on NLL.

Finally, Table 6.7 compares different methods in Classwise-ECE. While there is no single winning method on Classwise-ECE when the number of classes is less than 200, DIR is the best method on this metric in ImageNet and in overall. In the next section, we discuss a hidden bias in Classwise-ECE metric that might become problematic. It seems Classwise-ECE might promote uncertainty in the output regardless of the actual accuracy of the model. This suggests there might be more investigation required for this metric and a practitioner should be cautious about these numbers.

6.5.2.1 Is Classwise-ECE a Proper Scoring Rule Calibration Metric?

It is known that ECE is not a proper scoring rule and thus there exist trivial solutions which yield optimal scores [Ovadia et al., 2019]. In this section, we show the same holds for Classwise-ECE metric. Classwise-ECE is “defined as the average gap across all classwise-reliability diagrams, weighted by the number of instances in each bin:

$$\text{Classwise-ECE} = \frac{1}{k} \sum_{j=1}^k \sum_{i=1}^m \frac{|B_{i,j}|}{n} |y_j(B_{i,j}) - \hat{p}_j(B_{i,j})| \quad (6.1)$$

where k , m , n are the numbers of classes, bins and instances, respectively, $|B_{i,j}|$ denotes the size of the bin, and $\hat{p}_j(B_{i,j})$ and $y_j(B_{i,j})$ denote the average prediction of class j probability and the actual proportion of class j in the bin $B_{i,j}$.” [Kull et al., 2019].

Table 6.7: Classwise ECE.

Dataset	Model	Uncal.	TS	DIR	MS	DIAG	OI	OP
CIFAR10	ResNet 110	0.09846 ₇	0.04344 ₅	0.03950 ₄	0.03615 ₂	0.03791 ₃	0.03454 ₁	0.04435 ₆
CIFAR10	Wide ResNet 32	0.09530 ₇	0.04775 ₄	0.02947 ₂	0.02921 ₁	0.05462 ₆	0.04747 ₃	0.04918 ₅
CIFAR10	DenseNet 40	0.11430 ₇	0.03977 ₄	0.03687 ₂	0.03678 ₁	0.03877 ₃	0.04575 ₅	0.05182 ₆
SVHN	ResNet 152 (SD)	0.01940 ₄	0.01849 ₂	0.01988 ₅	0.02088 ₆	0.01478 ₁	0.01858 ₃	0.02128 ₇
CIFAR100	ResNet 110	0.41644 ₇	0.20095 ₃	0.18639 ₁	0.20270 ₅	0.21966 ₆	0.19977 ₂	0.20237 ₄
CIFAR100	Wide ResNet 32	0.42027 ₇	0.18573 ₄	0.17951 ₁	0.17966 ₂	0.18636 ₅	0.19397 ₆	0.18484 ₃
CIFAR100	DenseNet 40	0.47026 ₇	0.18664 ₃	0.18630 ₂	0.19112 ₅	0.18614 ₁	0.19866 ₆	0.18752 ₄
CARS	ResNet 50 (pretrained)	0.17353 ₃	0.18513 ₇	0.17094 ₂	0.18312 ₆	0.16891 ₁	0.18217 ₅	0.17567 ₄
CARS	ResNet 101 (pretrained)	0.16503 ₄	0.17186 ₆	0.15914 ₂	0.17405 ₇	0.16692 ₅	0.16434 ₃	0.15509 ₁
CARS	ResNet 101	0.26300 ₂	0.27234 ₆	0.26333 ₃	0.27447 ₇	0.26594 ₅	0.26488 ₄	0.25097 ₁
BIRDS	ResNet 50 (NTSNet)	0.24901 ₃	0.26369 ₇	0.22920 ₁	0.25639 ₆	0.25073 ₅	0.25069 ₄	0.24031 ₂
ImageNet	ResNet 152	0.31846 ₇	0.30886 ₄	0.30061 ₁	0.30895 ₅	0.31372 ₆	0.30642 ₃	0.30081 ₂
ImageNet	DenseNet 161	0.30992 ₇	0.30309 ₅	0.29403 ₁	0.29807 ₂	0.30659 ₆	0.30248 ₄	0.29959 ₃
ImageNet	PNASNet5 large	0.31356 ₇	0.25587 ₆	0.23797 ₁	0.24283 ₂	0.25004 ₅	0.24634 ₄	0.24493 ₃
Average Relative Error		1.000 ₇	0.752 ₆	0.704 ₁	0.734 ₃	0.729 ₂	0.740 ₄	0.743 ₅

While the above definition of Classwise-ECE intuitively makes sense, we show that this metric fails to represent the quality of a predictor in a common degenerate case e.g. in a balanced dataset with k classes one could achieve a perfect Classwise-ECE by scaling down the logits with a large enough positive scalar. A large enough temperature value increases the uncertainty of the model and brings all the class probabilities close to $1/k$ while maintaining the accuracy of the model. As the result, in all the classwise-reliability diagrams every data point falls into the bin that contains confidence values around $1/k$. Since the dataset is balanced, the actual proportion of class j in that bin will also be $1/k$ so the model exhibits a perfect Classwise-ECE.

We remark that this problem does not happen with ECE, because ECE is computed with regard to the *accuracy* of the bins. While all the data points still fall inside the bin that contains the confidence value $1/k$, the accuracy of this bin would be equal to the accuracy of the model. Thus, there would be mismatch between the confidence and the accuracy of the bin, which results to a high ECE.

To validate this insight, we scale down the uncalibrated logit values by a large scalar number and see how it affects Classwise-ECE in Table 6.8. It shows this simple hack drastically improves the Classwise-ECE value of the uncalibrated models and outperforms the methods in Table 6.7 by large margin in most of the cases. Note that we can not achieve perfect Classwise-ECE because the datasets are not perfectly balanced.

We are concerned that this issue with Classwise-ECE might bias future work to lean towards merely increasing the uncertainty of predictions without actually calibrating the model in a meaningful way. To avoid this, Classwise-ECE metric should be always used with other proper scoring rule metrics (e.g., NLL or Brier) in evaluation. As we discuss in the next section, this issue would not happen when bins are dynamically chosen to ensure the number of data points in each bin remains equal.

Table 6.8: Temperature scaling effect on Classwise-ECE. A large temperature value improves the Classwise-ECE in most of the cases. The subscript numbers represent the rank compared to the values in Table 6.7. We remark that the purpose of this experiment is not to improve the performance but rather highlight the need for studying Classwise-ECE metric in the future works.

Dataset	Model	Uncal.	Uncal./1000
CIFAR10	ResNet 110	0.09846	0.00021 ₁
CIFAR10	Wide ResNet 32	0.09530	0.00126 ₁
CIFAR10	DenseNet 40	0.11430	0.00143 ₁
SVHN	ResNet 152 (SD)	0.01940	0.33123 ₈
CIFAR100	ResNet 110	0.41644	0.00080 ₁
CIFAR100	Wide ResNet 32	0.42027	0.00199 ₁
CIFAR100	DenseNet 40	0.47026	0.00282 ₁
CARS	ResNet 50 (pretrained)	0.17353	0.16048 ₁
CARS	ResNet 101 (pretrained)	0.16503	0.16108 ₃
CARS	ResNet 101	0.26300	0.15067 ₁
BIRDS	ResNet 50 (NTSNet)	0.24901	0.05831 ₁
ImageNet	ResNet 152	0.31846	0.11074 ₁
ImageNet	DenseNet 161	0.30992	0.11074 ₁
ImageNet	PNASNet5 large	0.31356	0.10960 ₁

6.5.2.2 Debiased ECE and a Fix to Classwise-ECE

We believe that the issue mentioned above is due to the binning scheme used in estimating Classwise-ECE which allows all the data points fall into a single bin. [Nixon et al., 2019] propose an adaptive binning scheme that guarantees the number of data points in each bin remains balanced; therefore, it does not exhibit the same issue as Classwise-ECE.

In addition to the binning scheme, [Kumar et al., 2019] introduce *debaised ECE* and multiclass *marginal calibration error* metrics that are debaised versions similar to the ECE and Classwise-ECE metrics, respectively. The idea is to subtract an approximate correction term to reduce the biased estimate of the metrics. For the completeness, we present *debaised ECE* and multiclass *marginal calibration error* for all the methods in Table 6.9 and Table 6.10, respectively. While the results in debaised ECE are similar to ECE, comparing the results in Table 6.7 and Table 6.10 shows DIAG is performing better in terms of multiclass marginal calibration error and outperforms DIR in average relative error.

Overall, although the intra order-preserving models are the winning methods among most of the ever-increasing calibration metrics, one should carefully pick the calibration method and the metric depending on their application.

6.6 Summary

In this chapter, we have introduced the family of intra order-preserving functions which retain the top- k predictions of any deep network when used as the post-hoc calibration function. We have proposed a new neural network architecture to represent these functions, and new regularization techniques based on order-invariant

Table 6.9: Debiased ECE [Kumar et al., 2019].

Dataset	Model	Uncal.	TS	Dir	MS	DIAG	OI	OP
CIFAR-10	ResNet 110	0.09070 ₇	0.01924 ₄	0.01927 ₅	0.01716 ₃	0.01573 ₂	0.00000 ₁	0.02282 ₆
CIFAR-10	Wide ResNet 32	0.08661 ₇	0.00809 ₂	0.00943 ₃	0.00958 ₄	0.01717 ₆	0.00194 ₁	0.01073 ₅
CIFAR-10	DenseNet 40	0.10340 ₇	0.01195 ₂	0.01228 ₃	0.01266 ₄	0.01130 ₁	0.02410 ₆	0.02309 ₅
SVHN	ResNet 152 (SD)	0.01922 ₅	0.00892 ₂	0.00979 ₄	0.00939 ₃	0.00617 ₁	0.02269 ₆	0.03429 ₇
CIFAR-100	ResNet 110	0.22699 ₇	0.02004 ₂	0.02842 ₃	0.03054 ₅	0.05596 ₆	0.00626 ₁	0.02903 ₄
CIFAR-100	Wide ResNet 32	0.24827 ₇	0.01031 ₂	0.01909 ₅	0.03018 ₆	0.01498 ₄	0.00545 ₁	0.01408 ₃
CIFAR-100	DenseNet 40	0.26523 ₇	0.00000 ₁	0.00000 ₁	0.02809 ₆	0.00000 ₁	0.00432 ₄	0.01265 ₅
CARS	ResNet 50 (pre)	0.02327 ₆	0.00900 ₂	0.02512 ₇	0.01605 ₄	0.00000 ₁	0.01611 ₅	0.01363 ₃
CARS	ResNet 101 (pre)	0.02181 ₆	0.01956 ₃	0.02419 ₇	0.01504 ₂	0.01964 ₄	0.02136 ₅	0.01271 ₁
CARS	ResNet 101	0.04280 ₅	0.02654 ₃	0.01518 ₁	0.03728 ₄	0.02542 ₂	0.04766 ₆	0.04821 ₇
BIRDS	ResNet 50 (NTS)	0.47117 ₇	0.04054 ₄	0.05545 ₅	0.07224 ₆	0.01518 ₁	0.01650 ₂	0.03104 ₃
ImageNet	ResNet 152	0.07745 ₇	0.02157 ₄	0.05247 ₅	0.06099 ₆	0.00066 ₁	0.00941 ₂	0.01804 ₃
ImageNet	DenseNet 161	0.06598 ₇	0.02008 ₄	0.04542 ₅	0.04888 ₆	0.00998 ₁	0.01158 ₂	0.01924 ₃
ImageNet	PNASNet5 large	0.06820 ₆	0.09620 ₇	0.05728 ₅	0.03580 ₄	0.01273 ₃	0.00713 ₁	0.01272 ₂
Average Relative Error		1.000 ₇	0.357 ₃	0.430 ₆	0.409 ₅	0.213 ₁	0.337 ₂	0.406 ₄

Table 6.10: Marginal Calibration Error [Kumar et al., 2019].

Dataset	Model	Uncal.	TS	Dir	MS	DIAG	OI	OP
CIFAR-10	ResNet 110	0.00859 ₇	0.00305 ₂	0.00371 ₆	0.00363 ₅	0.00346 ₄	0.00218 ₁	0.00336 ₃
CIFAR-10	Wide ResNet 32	0.01516 ₇	0.01408 ₃	0.00410 ₁	0.00432 ₂	0.01442 ₆	0.01416 ₅	0.01410 ₄
CIFAR-10	DenseNet 40	0.01132 ₇	0.00602 ₄	0.00417 ₁	0.00583 ₂	0.00601 ₃	0.00729 ₆	0.00686 ₅
SVHN	ResNet 152 (SD)	0.00227 ₂	0.00245 ₃	0.00426 ₅	0.00541 ₆	0.00178 ₁	0.00387 ₄	0.00691 ₇
CIFAR-100	ResNet 110	0.00315 ₇	0.00129 ₁	0.00185 ₅	0.00233 ₆	0.00144 ₃	0.00141 ₂	0.00151 ₄
CIFAR-100	Wide ResNet 32	0.00356 ₇	0.00266 ₄	0.00222 ₂	0.00199 ₁	0.00270 ₆	0.00268 ₅	0.00257 ₃
CIFAR-100	DenseNet 40	0.00417 ₇	0.00266 ₆	0.00222 ₁	0.00263 ₅	0.00261 ₄	0.00259 ₃	0.00234 ₂
CARS	ResNet 50 (pre)	0.00063 ₆	0.00058 ₂	0.00035 ₁	0.00090 ₇	0.00060 ₅	0.00059 ₃	0.00059 ₃
CARS	ResNet 101 (pre)	0.00043 ₃	0.00044 ₄	0.00044 ₄	0.00092 ₇	0.00041 ₂	0.00034 ₁	0.00046 ₆
CARS	ResNet 101	0.00114 ₁	0.00114 ₁	0.00173 ₆	0.00230 ₇	0.00114 ₁	0.00118 ₅	0.00117 ₄
BIRDS	ResNet 50 (NTS)	0.00934 ₇	0.00139 ₄	0.00148 ₆	0.00141 ₅	0.00138 ₃	0.00132 ₂	0.00130 ₁
ImageNet	ResNet 152	0.00040 ₆	0.00038 ₂	0.00034 ₁	0.00042 ₇	0.00038 ₂	0.00038 ₂	0.00038 ₂
ImageNet	DenseNet 161	0.00041 ₇	0.00039 ₃	0.00035 ₁	0.00038 ₂	0.00039 ₃	0.00039 ₃	0.00039 ₃
ImageNet	PNASNet5 large	0.00039 ₇	0.00032 ₆	0.00025 ₁	0.00028 ₂	0.00028 ₂	0.00029 ₄	0.00030 ₅
Average Relative Error		1.000 ₇	0.750 ₃	0.735 ₂	0.996 ₆	0.725 ₁	0.778 ₄	0.898 ₅

and diagonal structures. In short, calibrating neural network with this new family of functions generalizes many existing calibration techniques, with additional flexibility to express the post-hoc calibration function. The experimental results show the importance of learning within the intra order-preserving family as well as support the effectiveness of the proposed regularization in calibrating multiple classifiers on various datasets.

Conclusion

Deep neural networks have been involved in some of the most spectacular breakthroughs in the last few years. In this thesis, we have argued that there are a few limitations that we need to consider when training or deploying these models for end-users.

We started to investigate tasks where there are limited examples with weak annotations available to the model. In particular, we considered the task of few-shot common object localization, where the task is to localize the objects of a target class in a small collection of images. Using the multiple-instance learning terminology, we have formalized the task as having a small collection of bags where each bag is composed of multiple elements. Assuming a novel target class, we only know whether the bags in the collection contain elements from the target class (positive bags) or not (negative bag). The goal is to find the elements that are from the target class in the collection. We have formulated the problem as an energy minimization on a densely connected graphical model with unary and pairwise potentials. Each bag corresponds to a node in this graphical model, and the bag elements correspond to the node's labels. The pairwise potentials encourage selecting elements from the same classes, and the unary potentials support selecting elements that are not from the negative bag. We learned unary and pairwise potentials from a large dataset of known objects based on the relation network model. We have proposed a novel attention mechanism to aggregate the information in the negative bag as our unary potentials. Furthermore, we have developed a greedy inference algorithm that uses the fact that a common object across a set of bags is also a valid common object across any subset of those bags. Our inference method does not necessarily compute all pairwise potentials and is beneficial in the tasks where computing all pairwise potentials is the bottleneck. Our experiments have been a support for the effectiveness of our method and inference algorithm. Our method outperforms traditional multiple-instance learning algorithms in limited data settings. On the other hand, our inference method achieves comparable results to the well-known graphical model inference algorithms while significantly reducing the computational costs.

We later focused on the task of few-shot weakly supervised object detection. We decomposed the problem into few-shot common object localization and few-shot object detection. We have developed a probabilistic approach to the few-shot common object localization problem that directly works on the features extracted from

a trained object detector. We assumed each class is distributed based on the von Mises-Fisher distribution on the unit hyper-sphere and learned the parameters of the distributions and assignment of proposals to different classes jointly using the expectation-maximization (EM) algorithm. In contrast to our previous greedy inference method, this approach did not require hyperparameters or episodic training. We have extended common object localization to solve the few-shot weakly-supervised object detection problem. To this end, we can localize objects of novel classes in unseen query images.

We have studied knowledge transfer for weakly supervised object localization to reduce the cost of effort and time in annotating large-scale object detection datasets for novel classes. In this problem, the goal is to transfer knowledge from a source dataset of known objects with full bounding box annotations to find instances of novel objects in a target dataset with only image-level labels. Traditional multiple-instance learning approaches leverage objectness scores as knowledge transfer from the source dataset. In contrast, we discussed that using just objectness is a limited form of knowledge transfer and additionally transferred a pairwise similarity measure that compares pairs of proposals. Since the target dataset of novel objects was a large-scale dataset, we were less prone to overfitting than the few-shot common object localization problem. Hence, we adapted our pairwise and unary similarities using the target dataset. We learned the similarity functions and the assignment of proposals to different target classes jointly using alternating optimization. We have shown that the assignment problem becomes an integer linear program for a certain type of loss function and have introduced an efficient inference algorithm to solve the challenge of computing all pairwise similarities.

Lastly, we have investigated another limitation of deep neural networks in which they tend to output miscalibrated confidence scores. We employed post-hoc calibration to resolve this limitation and discussed that since the calibration set usually has a limited size, a general calibration function may overfit and decrease the accuracy of the trained model. To this end, we have introduced the family of the intra order-preserving functions as regularization in the space of calibration function to learn a general calibration function and at the same time maintain the accuracy of the trained model. We have presented two other sub-families based on diagonal and order-invariant functions that benefit from parameter sharing across different classes and are more effective in limited data scenarios. We have identified necessary and sufficient conditions for representing intra order-preserving functions, examined their differentiability, and suggested a novel neural network architecture that can express complex intra order-preserving functions by common neural network components. Our comprehensive experiments have confirmed the effectiveness of the proposed regularizations in the space of calibration functions.

7.1 Future Directions

In this section, we list several possible extensions to the methods discussed in this thesis.

Multiple diverse proposals for common objects. Current graphical model inference methods provide a single selection as their final result. In contrast, our greedy inference method is proposal-based inherently and can be modified to provide multiple diverse solutions to the problem. This feature can be useful in cases where there are multiple common objects in the collection. In addition, the top selected solution proposal may contain some falsely selected elements. So, providing multiple possible solution proposals can help the downstream tasks to have higher accuracy. For example, the similarity of multiple solution proposals to query regions can help select the best solution proposal and consequently increase the performance in the few-shot weakly supervised object detection task.

Universal Cross-Transformers for few-shot classification. The EM idea for common object localization applies to the few-shot classification problem as well. In a recent work [Doersch et al., 2020], Cross-Transformers are utilized to densely match regions between the query image and images in the support set in a single step. We can better localize the regions of interest in the support set with the EM iterates assuming a distribution over the common object present in images of the support set containing the same class. One might perceive the EM iterates as the recurrent iterates in the universal transformers [Dehghani et al., 2019]. Here, we ought to find a probabilistic view of the recurrence and self-attention in the universal transformers.

Incorporating semantic knowledge. In addition to pairwise similarity knowledge transfer, we can also leverage semantic knowledge such as word vectors and textual attributes to describe novel classes. Consolidating semantic knowledge with pairwise knowledge transfer unites weakly supervised and zero-shot learning methods and can boost the performance of respective models.

Other applications of intra order-preserving functions. We believe the applications of intra order-preserving family are not limited to network calibration. Other promising domains include, e.g., data compression, depth perception system calibration, and tone-mapping in images where tone-maps need to be monotonic. Exploring the applicability of intra order-preserving functions in other applications is an interesting future direction.

Appendix A

A.1 Modeling with Gaussian Distribution

In Section 4.3.4, we conduct experiment with Gaussian distribution used to model the common object distribution. We assume the common object distribution is Gaussian with mean θ and diagonal covariance matrix $\sigma^2 I$, i.e., $\mathbf{x} \sim \mathcal{N}(\theta, \sigma^2 I)$

$$p_{\theta_c}^+(\mathbf{x}) = \frac{1}{Z} \exp\left(-\frac{\|\mathbf{x} - \theta\|^2}{2\sigma^2}\right), \quad (\text{A.1})$$

and plug the distribution into Equation (4.9) to get the soft label update rule as

$$\mathbf{w}_{ik} = \frac{\exp\left(-\frac{\|F_{ik} - \theta\|^2}{2\sigma^2} - \log u_{\omega}^-(F_{ik})\right)}{\sum_{j=1}^P \exp\left(-\frac{\|F_{ij} - \theta\|^2}{2\sigma^2} - \log u_{\omega}^-(F_{ij})\right)}. \quad (\text{A.2})$$

E-step is computed by setting the derivative of Equation (4.11) w.r.t. θ to zero

$$\theta \leftarrow \frac{1}{M} \sum_{i=1}^M \sum_{k=1}^P \mathbf{w}_{ik} F_{ik} = \frac{1}{M} \sum_{i=1}^M \mathbf{w}_i^\top F_i, \quad (\text{A.3})$$

A.2 MI-SVM WSOD Baseline

To the best of our knowledge there is no WSOD algorithm for few-shot setting in the literature. However, WSOD with knowledge transfer methods [Rahimi et al., 2020a; Uijlings et al., 2018; Hoffman et al., 2016; Deselaers et al., 2010] are closely related to our work. In this section, we describe our slightly modified version of [Uijlings et al., 2018] introduced in Section 2.5.1 and discuss its differences to the proposed method. In Section 4.3, we empirically compare our work against [Uijlings et al., 2018]. We use highly efficient GPU solver in [Amos and Kolter, 2017] for SVM optimization. To have a fair comparison, we use the same Faster-RCNN model trained on the base classes as we used in our model to extract bounding box and feature proposals from all images. For the objectness model O , we learn a class-agnostic logistic regression model on the extracted feature.

The expectation and maximization steps in our method are analogous to re-localization and re-training steps in [Uijlings et al., 2018]. In MI-SVM, only the proposal with the highest score is labeled positive in the re-localization step while our COL method infers soft labels in the expectation step via an attention mechanism. Using soft labels could be beneficial as they reflect the uncertainty in choosing the common object.

Appendix B

B.1 Missing Proof for Linearity of Labels in Sigmoid Cross-entropy Loss Function

Let $\ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ be the sigmoid cross-entropy loss function

$$\ell(x, y) = -(1 - y) \log(1 - \sigma(x)) - y \log(\sigma(x)),$$

where $\sigma(x) = 1 / (1 + \exp(-x))$ is the sigmoid function. Then, $\ell(x, y) = \ell(x, 0) - yx$, for any $x \in \mathbb{R}$ and $y \in [0, 1]$.

Proof.

$$\begin{aligned} \ell(x, y) - \ell(x, 0) &= \left(-(1 - y) \log(1 - \sigma(x)) - y \log(\sigma(x)) \right) + \log(1 - \sigma(x)) \\ &= y \log(1 - \sigma(x)) - y \log(\sigma(x)) \\ &= -yx \end{aligned}$$

Last equality is derived using the fact that $\log(1 - \sigma(x)) - \log(\sigma(x)) = -x$ which can be easily verified by plugging in the sigmoid function. ■

Appendix C

C.1 Missing Proofs for Intra Order-Preserving Functions

Theorem 1. *A continuous function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is intra order-preserving, if and only if $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}U\mathbf{w}(\mathbf{x})$ with U being an upper-triangular matrix of ones and $\mathbf{w} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ being a continuous function such that*

- $\mathbf{w}_i(\mathbf{x}) = 0$, if $\mathbf{y}_i = \mathbf{y}_{i+1}$ and $i < n$,
- $\mathbf{w}_i(\mathbf{x}) > 0$, if $\mathbf{y}_i > \mathbf{y}_{i+1}$ and $i < n$,
- $\mathbf{w}_n(\mathbf{x})$ is arbitrary,

where $\mathbf{y} = S(\mathbf{x})\mathbf{x}$ is the sorted version of \mathbf{x} .

Proof of Theorem 1. (\rightarrow) For a continuous intra order-preserving function $\mathbf{f}(\mathbf{x})$, let $\mathbf{w}(\mathbf{x}) = U^{-1}S(\mathbf{x})\mathbf{f}(\mathbf{x})$. First we show \mathbf{w} is continuous. Because \mathbf{f} is intra order-preserving, it holds that $S(\mathbf{x}) = S(\mathbf{f}(\mathbf{x}))$. Let $\hat{\mathbf{f}}(\mathbf{x}) := S(\mathbf{f}(\mathbf{x}))\mathbf{f}(\mathbf{x})$ be the sorted version of $\mathbf{f}(\mathbf{x})$. The above implies $\mathbf{w}(\mathbf{x}) = U^{-1}\hat{\mathbf{f}}(\mathbf{x})$. By Lemma 1, we know $\hat{\mathbf{f}}$ is continuous and therefore \mathbf{w} is also continuous.

Lemma 1. *Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a continuous intra order-preserving function. $S(\mathbf{f}(\mathbf{x}))\mathbf{f}(\mathbf{x})$ is a continuous function.*

Next, we show that \mathbf{w} satisfies the properties listed in Theorem 1. As $\mathbf{w}(\mathbf{x}) = U^{-1}\hat{\mathbf{f}}(\mathbf{x})$, we can equivalently write \mathbf{w} as

$$\mathbf{w}_i(\mathbf{x}) = \begin{cases} \hat{\mathbf{f}}_i(\mathbf{x}) - \hat{\mathbf{f}}_{i+1}(\mathbf{x}) & 1 \leq i < n \\ \hat{\mathbf{f}}_n(\mathbf{x}) & i = n. \end{cases}$$

Since $\hat{\mathbf{f}}$ is the sorted version of \mathbf{f} , it holds that $\mathbf{w}_i(\mathbf{x}) \geq 0$ for $1 \leq i < n$. Also, by the definition of the order-preserving function, $\mathbf{w}_i(\mathbf{x})$ can be zero if and only if $\mathbf{y}_i = \mathbf{y}_{i+1}$, where $\mathbf{y} = S(\mathbf{x})\mathbf{x}$. These two arguments prove the necessary condition.

(\leftarrow) For a given $\mathbf{w}(\mathbf{x})$ satisfying the condition in the theorem statement, let $\mathbf{v}(\mathbf{x}) = U\mathbf{w}(\mathbf{x})$. Equivalently, we can write $\mathbf{v}_i(\mathbf{x}) = \sum_{j=0}^{n-i} \mathbf{w}_{n-j}(\mathbf{x})$ and $\mathbf{v}_i(\mathbf{x}) - \mathbf{v}_{i+1}(\mathbf{x}) = \mathbf{w}_i(\mathbf{x})$, $\forall i \in \llbracket n \rrbracket$. By construction of \mathbf{w} , one can conclude that $\mathbf{v}(\mathbf{x})$ is a sorted vector where

two consecutive elements $\mathbf{v}_i(\mathbf{x})$ and $\mathbf{v}_{i+1}(\mathbf{x})$ are equal if and only if $\mathbf{y}_i = \mathbf{y}_{i+1}$. Therefore, $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}\mathbf{v}(\mathbf{x})$ has the same ranking as \mathbf{x} . In other words, \mathbf{f} is an intra order-preserving function. The continuity of \mathbf{f} follows from the lemma below and the fact that \mathbf{v} is continuous when \mathbf{w} is continuous.

Lemma 2. *Let $\mathbf{v} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a continuous function in which $\mathbf{v}_i(\mathbf{x})$ and $\mathbf{v}_{i+1}(\mathbf{x})$ are equal if and only if $\mathbf{y}_i = \mathbf{y}_{i+1}$, where $\mathbf{y} = S(\mathbf{x})\mathbf{x}$. Then $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}\mathbf{v}(\mathbf{x})$ is a continuous function.*

■

C.1.0.1 Deferred Proofs of Lemmas

Proof of Lemma 1. Let $\mathbb{P}^n = \{P_1, \dots, P_K\}$ be the finite set of all possible $n \times n$ dimensional permutation matrices. For each $k \in [K]$, define the closed set $\mathbb{N}_k = \{\mathbf{x} : S(\mathbf{x})\mathbf{x} = P_k\mathbf{x}\}$. These sets are convex polyhedrons since each can be defined by a finite set of linear inequalities; in addition, they together form a covering set of \mathbb{R}^n . Note that $S(\mathbf{x}) = P_k$ is constant in the interior $\text{int}(\mathbb{N}_k)$, but $S(\mathbf{x})$ may change on the boundary $\partial(\mathbb{N}_k)$ which corresponds to points where a tie exists in elements of \mathbf{x} (for such a point $S(\mathbf{x}) \neq P_k$). Nonetheless, by definition of the set \mathbb{N}_k , we have $S(\mathbf{x})\mathbf{x} = P_k\mathbf{x}$ for all $\mathbf{x} \in \mathbb{N}_k$, which implies that $S(\mathbf{x})$ and P_k can only have different elements for indices where elements of \mathbf{x} are equal.

To prove that $\hat{\mathbf{f}}(\mathbf{x}) := S(\mathbf{f}(\mathbf{x}))\mathbf{f}(\mathbf{x})$ is continuous, we leverage the fact that $\hat{\mathbf{f}}(\mathbf{x}) = S(\mathbf{x})\mathbf{f}(\mathbf{x})$ for intra order-preserving \mathbf{f} . We will first show that $\hat{\mathbf{f}}(\mathbf{x}) = P_k\mathbf{f}(\mathbf{x})$ for $\mathbf{x} \in \mathbb{N}_k$ and any $k \in [K]$, which implies $\hat{\mathbf{f}}$ is continuous on \mathbb{N}_k when \mathbf{f} is continuous. To see this, consider an arbitrary $k \in [K]$. For $\mathbf{x} \in \text{int}(\mathbb{N}_k)$ in the interior, we have $S(\mathbf{x}) = P_k$ and therefore $\hat{\mathbf{f}}(\mathbf{x}) = P_k\mathbf{f}(\mathbf{x})$. For $\mathbf{x} \in \partial\mathbb{N}_k$ on the boundary, we have

$$\hat{\mathbf{f}}(\mathbf{x}) = S(\mathbf{x})\mathbf{f}(\mathbf{x}) = P_k\mathbf{f}(\mathbf{x}).$$

The last equality holds because the difference between $S(\mathbf{x})$ and P_k are only in the indices for which elements of \mathbf{x} are equal, and the order-preserving \mathbf{f} preserves exactly the same equalities. Thus, the differences between permutations $S(\mathbf{x})$ and P_k do not reflect in the products $S(\mathbf{x})\mathbf{f}(\mathbf{x})$ and $P_k\mathbf{f}(\mathbf{x})$.

Next, we show that $\hat{\mathbf{f}}(\mathbf{x}) = P_k\mathbf{f}(\mathbf{x}) = P_{k'}\mathbf{f}(\mathbf{x})$ for $\mathbf{x} \in \partial\mathbb{N}_k \cap \partial\mathbb{N}_{k'}$. While $P_k \neq P_{k'}$, the intersection $\partial\mathbb{N}_k \cap \partial\mathbb{N}_{k'}$ contains exactly points \mathbf{x} such that the index differences in P_k and $P_{k'}$ correspond to same value in \mathbf{x} . Because \mathbf{f} is order-preserving, by an argument similar to the previous step, we have $P_k\mathbf{f}(\mathbf{x}) = P_{k'}\mathbf{f}(\mathbf{x})$ for $\mathbf{x} \in \partial\mathbb{N}_k \cap \partial\mathbb{N}_{k'}$.

Together these two steps and the fact that $\{\mathbb{N}_k\}$ is covering set on \mathbb{R}^n show that $\hat{\mathbf{f}}$ is a piece-wise continuous function on \mathbb{R}^n when \mathbf{f} is continuous on \mathbb{R}^n . ■

Proof of Lemma 2. In order to show the continuity of $\mathbf{f}(\mathbf{x})$, we use a similar argument as in Lemma 1 (see therein for notation definitions). For any $k \in [K]$, it is also trivial to show that \mathbf{f} is continuous over the open set $\text{int}(\mathbb{N}_k)$ since $\mathbf{f}(\mathbf{x}) = P_k^{-1}\mathbf{v}(\mathbf{x})$. We use the same argument as Lemma 1 to show it is also a continuous for any point

$$\mathbf{x} \in \partial(\mathbb{N}_k)$$

$$\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}\mathbf{v}(\mathbf{x}) = P_k^{-1}\mathbf{v}(\mathbf{x}).$$

The last equality holds because P_k^{-1} and $S(\mathbf{x})^{-1}$ can only have different elements among elements of $\mathbf{y} = S(\mathbf{x})\mathbf{x}$ with equal values, and \mathbf{v} preserves exactly these equalities in \mathbf{y} . Finally, the proof can be completed by piecing the results of different \mathbb{N}_k together. ■

C.1.1 Proof of Theorem 2, Order-invariant Functions

Theorem 2. *A continuous, intra order-preserving function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is order-invariant, if and only if $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}U\mathbf{w}(\mathbf{y})$, where U , \mathbf{w} , and \mathbf{y} are in Theorem 1.*

To prove Theorem 2, we first study the properties of order invariant functions in Appendix C.1.1.1. We will provide necessary and sufficient conditions to describe order invariant functions, like what we did in Theorem 1 for intra order-preserving functions. Finally, we combine these insights and Theorem 1 to prove Theorem 2 in Appendix C.1.1.2.

C.1.1.1 Properties of Order Invariant Functions

The goal of this section is to prove the below theorem, which characterizes the representation of order invariant functions using the concept of equality-preserving.

Definition 6. We say a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is *equality-preserving*, if $\mathbf{f}_i(\mathbf{x}) = \mathbf{f}_j(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x}_i = \mathbf{x}_j$ for some $i, j \in \llbracket n \rrbracket$

Theorem 4. *A function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is order-invariant, if and only if $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}\bar{\mathbf{f}}(S(\mathbf{x})\mathbf{x})$ for some function $\bar{\mathbf{f}} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ that is equality-preserving on the domain $\{\mathbf{y} : \mathbf{y} = S(\mathbf{x})\mathbf{x}, \text{ for } \mathbf{x} \in \mathbb{R}^n\}$.*

Theorem 4 shows an order invariant function can be expressed in terms of some equality-preserving function. In fact, every order invariant function is equality-preserving.

Proposition 1. *Any order-invariant function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is equality-preserving.*

Proof. Let $P_{ij} \in \mathbb{P}^n$ denote the permutation matrix that only swaps i^{th} and j^{th} elements of the input vector; i.e. $\mathbf{y} = P_{ij}\mathbf{x} \Rightarrow \mathbf{y}_i = \mathbf{x}_j, \mathbf{y}_j = \mathbf{x}_i, \mathbf{y}_k = \mathbf{x}_k, \forall \mathbf{x} \in \mathbb{R}^n, i, j, k \in \llbracket n \rrbracket$, and $k \neq i, j$. Thus, for an order-invariant function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and any $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x}_i = \mathbf{x}_j$, we have

$$\mathbf{f}(P_{ij}\mathbf{x}) = P_{ij}\mathbf{f}(\mathbf{x}) \Rightarrow \mathbf{f}_i(P_{ij}\mathbf{x}) = \mathbf{f}_j(\mathbf{x}) \Rightarrow \mathbf{f}_i(\mathbf{x}) = \mathbf{f}_j(\mathbf{x}) \quad (\because P_{ij}\mathbf{x} = \mathbf{x} \text{ for } \mathbf{x} \text{ such that } \mathbf{x}_i = \mathbf{x}_j).$$
■

We are almost ready to prove Theorem 4. We just need one more technical lemma, whose proof is deferred to the end of this section.

Lemma 3. For any $P \in \mathbb{P}^n$ and an equality-preserving $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $S(\mathbf{x})\mathbf{f}(\mathbf{x}) = S(P\mathbf{x})P\mathbf{f}(\mathbf{x})$.

Proof of Theorem 4. (\rightarrow) For an order-invariant function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, we have $\mathbf{f}(P\mathbf{x}) = P\mathbf{f}(\mathbf{x})$ by Definition 3 for any $P \in \mathbb{P}^n$. Take $P = S(\mathbf{x})$. We then have the equality $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}\mathbf{f}(S(\mathbf{x})\mathbf{x})$. This is an admissible representation because, by Proposition 1, \mathbf{f} is equality-preserving.

(\leftarrow) Let $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}\bar{\mathbf{f}}(S(\mathbf{x})\mathbf{x})$ for some equality-preserving function $\bar{\mathbf{f}}$. First, because $\bar{\mathbf{f}}$ is equality preserving and \mathbf{f} is constructed through the sorting function S , we notice that $\mathbf{f}(\mathbf{x})$ is equality-preserving. Next, we show \mathbf{f} is also order invariant:

$$\begin{aligned}
 \mathbf{f}(P\mathbf{x}) &= S(P\mathbf{x})^{-1}\bar{\mathbf{f}}(S(P\mathbf{x})P\mathbf{x}) \\
 &= S(P\mathbf{x})^{-1}\bar{\mathbf{f}}(S(\mathbf{x})\mathbf{x}) \quad (\because S(P\mathbf{x})P\mathbf{x} = S(\mathbf{x})\mathbf{x} \text{ by choosing } \mathbf{f}(\mathbf{x}) = \mathbf{x} \text{ in Lemma 3}) \\
 &= S(P\mathbf{x})^{-1}S(\mathbf{x})\mathbf{f}(\mathbf{x}) \quad (\because \text{definition of } \mathbf{f}(\mathbf{x})) \\
 &= S(P\mathbf{x})^{-1}S(P\mathbf{x})P\mathbf{f}(\mathbf{x}) \quad (\because \text{Lemma 3}) \\
 &= P\mathbf{f}(\mathbf{x}).
 \end{aligned}$$

■

C.1.1.2 Main Proof

Proof of Theorem 2. (\rightarrow) From Theorem 1 we can write $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}U\mathbf{w}(\mathbf{x})$. On the other hand, from Theorem 4 we can write $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}\bar{\mathbf{f}}(\mathbf{y})$ for some equality-preserving function $\bar{\mathbf{f}}$. Using both we can identify $\mathbf{w}(\mathbf{x}) = U^{-1}\bar{\mathbf{f}}(\mathbf{y})$ which implies that \mathbf{w} is only a function of the sorted input \mathbf{y} and can be equivalently written as $\mathbf{w}(\mathbf{y})$.

(\leftarrow) For \mathbf{w} with the properties in the theorem statement, the function $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}U\mathbf{w}(\mathbf{y})$ satisfies the conditions of Theorem 1; therefore \mathbf{f} is intra order-preserving. To show \mathbf{f} is also order-invariant, we write $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}\bar{\mathbf{f}}(\mathbf{y})$ where $\bar{\mathbf{f}}(\mathbf{y}) = U\mathbf{w}(\mathbf{y})$. Because $\bar{\mathbf{f}}_i(\mathbf{y}) = \sum_{j=0}^{n-i} \mathbf{w}_{n-j}(\mathbf{x})$, we can derive with the definition of \mathbf{w} that

$$\mathbf{y}_i = \mathbf{y}_{i+1} \Rightarrow \mathbf{w}_i(\mathbf{y}) = 0 \Rightarrow \bar{\mathbf{f}}_i(\mathbf{x}) = \bar{\mathbf{f}}_{i+1}(\mathbf{x}).$$

That is, $\bar{\mathbf{f}}(\mathbf{y})$ is equality-preserving on the domain of sorted inputs. Thus, \mathbf{f} is also order-invariant. ■

C.1.1.3 Deferred Proof of Lemmas

Proof of Lemma 3. To prove the statement, we first notice a fact that $S(\mathbf{x}) = S(P\mathbf{x})P$, for any $P \in \mathbb{P}^n$ and $\mathbf{x} \in \mathbb{X} := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}_i \neq \mathbf{x}_j, \forall i, j \in \llbracket n \rrbracket, i \neq j\}$. Therefore, for $\mathbf{x} \in \mathbb{X}$, we have $S(\mathbf{x})\mathbf{f}(\mathbf{x}) = S(P\mathbf{x})P\mathbf{f}(\mathbf{x})$.

Otherwise, consider some $\mathbf{x} \in \mathbb{R}^n \setminus \mathbb{X}$. Without loss of generality¹, we may consider $n > 2$ and \mathbf{x} such that $\mathbf{x}_1 = \mathbf{x}_2 > \mathbf{x}_k$ for all $k > 2$; because \mathbf{f} is equality-preserving, we have $\mathbf{f}_1(\mathbf{x}) = \mathbf{f}_2(\mathbf{x})$.

To prove the desired equality, we will introduce some extra notations. We use subscript $i:j$ to extract contiguous parts of a vector, e.g. $\mathbf{x}_{2:n} = [\mathbf{x}_2, \dots, \mathbf{x}_n]$ and $\mathbf{f}_{2:n}(\mathbf{x}) = [\mathbf{f}_2(\mathbf{x}), \dots, \mathbf{f}_n(\mathbf{x})]$ (by our construction of \mathbf{x} , $\mathbf{x}_{2:n}$ is a vector where each element is unique.) In addition, without loss of generality, suppose $P \in \mathbb{P}^n$ shifts index 1 to some index $i \in \llbracket n \rrbracket$; we define $\bar{P} \in \{0, 1\}^{n-1 \times n-1}$ by removing the 1st column and the i th row of P (which is also a permutation matrix). Using this notion, we can partition $S(\bar{P}\mathbf{x}_{2:n}) \in \{0, 1\}^{n-1 \times n-1}$ as

$$S(\bar{P}\mathbf{x}_{2:n}) = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

where $B_1 \in \mathbb{R}^{1 \times i-1}$, $B_2 \in \mathbb{R}^{1 \times n-i}$, $B_3 \in \mathbb{R}^{n-2 \times i-1}$, and $B_4 \in \mathbb{R}^{n-2 \times n-i}$. This would imply that $S(P\mathbf{x}) \in \{0, 1\}^{n \times n}$ can be written as one of followings

$$\begin{bmatrix} e_i^\top & & \\ B_1 & 0 & B_2 \\ B_3 & 0 & B_4 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} B_1 & 0 & B_2 \\ & e_i^\top & \\ B_3 & 0 & B_4 \end{bmatrix} \quad (\text{C.1})$$

where e_i is the i th canonical basis.

To prove the statement, let $\mathbf{y} = P\mathbf{f}(\mathbf{x})$. By the definition of \bar{P} , we can also write \mathbf{y} as

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_{1:i-1} \\ \mathbf{y}_i \\ \mathbf{y}_{i+1:n} \end{bmatrix} = \begin{bmatrix} (\bar{P}\mathbf{f}_{2:n}(\mathbf{x}))_{1:i-1} \\ \mathbf{f}_1(\mathbf{x}) \\ (\bar{P}\mathbf{f}_{2:n}(\mathbf{x}))_{i:n-1} \end{bmatrix} \quad (\text{C.2})$$

Let us consider the first case in (C.1). We have

$$S(P\mathbf{x})P\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{y}_i \\ B_1\mathbf{y}_{1:i-1} + B_2\mathbf{y}_{i+1:n} \\ B_3\mathbf{y}_{1:i-1} + B_4\mathbf{y}_{i+1:n} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_i \\ S(\bar{P}\mathbf{x}_{2:n})\bar{P}\mathbf{f}_{2:n}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1(\mathbf{x}) \\ S(\mathbf{x}_{2:n})\mathbf{f}_{2:n}(\mathbf{x}) \end{bmatrix} = S(\mathbf{x})\mathbf{f}(\mathbf{x})$$

where the second equality follows from (C.2), the third from the fact we proved at the beginning for the set \mathbb{X} , and the last equality is due to the assumption $\mathbf{x}_1 = \mathbf{x}_2 > \mathbf{x}_k$ and the equality-preserving property that $\mathbf{f}_1(\mathbf{x}) = \mathbf{f}_2(\mathbf{x})$. For the second case in (C.1), based on the same reasoning above, we can show

$$S(P\mathbf{x})P\mathbf{f}(\mathbf{x}) = \begin{bmatrix} (S(\mathbf{x}_{2:n})\mathbf{f}_{2:n}(\mathbf{x}))_1 \\ \mathbf{f}_1(\mathbf{x}) \\ (S(\mathbf{x}_{2:n})\mathbf{f}_{2:n}(\mathbf{x}))_{2:n-1} \end{bmatrix},$$

Because $\mathbf{x}_1 = \mathbf{x}_2$, we have $(S(\mathbf{x}_{2:n})\mathbf{f}_{2:n}(\mathbf{x}))_1 = \mathbf{f}_1(\mathbf{x}) = \mathbf{f}_2(\mathbf{x})$. Thus, $S(P\mathbf{x})P\mathbf{f}(\mathbf{x}) =$

¹This choice is only for convenience of writing the indices.

$S(\mathbf{x})\mathbf{x}$. ■

C.1.2 Proof of Theorem 3, Diagonal Functions

Theorem 3. *A continuous, intra order-preserving function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is diagonal, if and only if $\mathbf{f}(\mathbf{x}) = [\bar{f}(\mathbf{x}_1), \dots, \bar{f}(\mathbf{x}_n)]$ for some continuous and increasing function $\bar{f} : \mathbb{R} \rightarrow \mathbb{R}$.*

We first prove some properties of *diagonal* intra order-preserving functions, which will be used to prove Theorem 3.

Proposition 2. *Any intra order-preserving function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is equality-preserving.*

Proof. This can be seen directly from the definition of intra order-preserving functions. ■

Corollary 2. *The following statements are equivalent*

1. A function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is diagonal and equality-preserving.
2. $\mathbf{f}(\mathbf{x}) = [\bar{f}(\mathbf{x}_1), \dots, \bar{f}(\mathbf{x}_n)]$ for some $\bar{f} : \mathbb{R} \rightarrow \mathbb{R}$.
3. A function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is diagonal and order-invariant.

Proof. (1 \rightarrow 2) Let $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}_1), \dots, f_n(\mathbf{x}_n)]$ be a diagonal and equality-preserving function. One can conclude that $\mathbf{f}_1(x) = \dots = \mathbf{f}_n(x)$ for all $x \in \mathbb{R}$.

(2 \rightarrow 3) Let $\mathbf{u} = P\mathbf{x}$ for some permutation matrix $P \in \mathbb{P}^n$. Then $\mathbf{f}(P\mathbf{x}) = [\bar{f}(\mathbf{u}_1), \dots, \bar{f}(\mathbf{u}_n)] = P[\bar{f}(\mathbf{x}_1), \dots, \bar{f}(\mathbf{x}_n)] = P\mathbf{f}(\mathbf{x})$.

(3 \rightarrow 1) True by Proposition 1. ■

Proof of Theorem 3. (\rightarrow) By Proposition 2, an intra order-preserving function \mathbf{f} is also equality-preserving. Therefore, by Corollary 2 it can be represented in the form $\mathbf{f}(\mathbf{x}) = [\bar{f}(\mathbf{x}_1), \dots, \bar{f}(\mathbf{x}_n)]$ for some $\bar{f} : \mathbb{R} \rightarrow \mathbb{R}$. Furthermore, because $\mathbf{f}(\mathbf{x})$ is intra order-preserving, for any $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{x}_1 > \mathbf{x}_2$, it satisfies $\mathbf{f}_1(\mathbf{x}_1) > \mathbf{f}_2(\mathbf{x}_2)$; that is, $\bar{f}(\mathbf{x}_1) > \bar{f}(\mathbf{x}_2)$. Therefore, \bar{f} is an increasing function. Continuity is inherited naturally.

(\leftarrow) Because $\mathbf{f}_i(\mathbf{x}) = \bar{f}(\mathbf{x}_i)$ and \bar{f} is an increasing function, it follows that \mathbf{f} is intra order-preserving

$$\mathbf{x}_i = \mathbf{x}_j \Rightarrow \mathbf{f}_i(\mathbf{x}) = \mathbf{f}_j(\mathbf{x}) \quad \text{and} \quad \mathbf{x}_i > \mathbf{x}_j \Rightarrow \mathbf{f}_i(\mathbf{x}) > \mathbf{f}_j(\mathbf{x}).$$
■

Finally, we prove that diagonal intra order-preserving functions are also order-invariant. This fact was mentioned in the Chapter 6 without a proof.

Corollary 3. *A diagonal intra order-preserving function is also order-invariant.*

Proof. Intra order-preserving functions are equality-preserving by Proposition 2. By Corollary 2 an diagonal equality-preserving function is order-invariant. ■

C.2 Continuity and Differentiability of the Proposed Architecture

In this section, we discuss properties of the function $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}UD(\mathbf{y})\mathbf{m}(\mathbf{x})$. In order to learn the parameters of \mathbf{m} with a first order optimization algorithm, it is important for \mathbf{f} to be differentiable with respect to the parameters of \mathbf{m} . This condition holds in general, since the only potential sources of non-differentiable \mathbf{f} , $S(\mathbf{x})^{-1}$ and \mathbf{y} are constant with respect to the parameters of \mathbf{m} . Thus, if \mathbf{m} is differentiable with respect to its parameters, \mathbf{f} is also differentiable with respect to the parameters of \mathbf{m} .

Next, we discuss continuity and differentiability of $\mathbf{f}(\mathbf{x})$ with respect the *input* \mathbf{x} . These properties are important when the input to function f is first processed by a trainable function \mathbf{g} (i.e. the final output is computed as $\mathbf{f} \circ \mathbf{g}(\mathbf{x})$). This is not the case in post-hoc calibration considered in Chapter 6, since the classifier \mathbf{g} here is not being trained in the calibration phase.

We show below that when $\mathbf{w}(\mathbf{x}) = D(\mathbf{y})\mathbf{m}(\mathbf{x})$ satisfies the requirements in Theorem 1, the function $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}UD(\mathbf{y})\mathbf{m}(\mathbf{x})$ is a continuous intra order-preserving function.

Corollary 4. *Let $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be a continuous function where $\sigma(0) = 0$ and strictly positive on $\mathbb{R} \setminus \{0\}$, and let \mathbf{m} be a continuous function where $\mathbf{m}_i(\mathbf{x}) > 0$ for $i < n$, and arbitrary for $\mathbf{m}_d(\mathbf{y})$. Let $D(\mathbf{y})$ denote a diagonal matrix with entries $D_{ii} = \sigma(\mathbf{y}_i - \mathbf{y}_{i+1})$ for $i < n$ and $D_{nn} = 1$. Then $\mathbf{w}(\mathbf{x}) = D(\mathbf{y})\mathbf{m}(\mathbf{x})$ is a continuous function and satisfies the following conditions*

- $\mathbf{w}_i(\mathbf{x}) = 0$, for $i < n$ and $\mathbf{y}_i = \mathbf{y}_{i+1}$
- $\mathbf{w}_i(\mathbf{x}) > 0$, for $i < n$ and $\mathbf{y}_i > \mathbf{y}_{i+1}$
- $\mathbf{w}_n(\mathbf{x})$ is arbitrary,

where $\mathbf{y} = S(\mathbf{x})\mathbf{x}$ is the sorted version of \mathbf{x} .

Proof. First, because $\mathbf{y} = S(\mathbf{x})\mathbf{x}$ is a continuous function (by Lemma 1 with $\mathbf{f}(\mathbf{x}) = \mathbf{x}$), $\mathbf{w}(\mathbf{x}) = D(\mathbf{y})\mathbf{m}(\mathbf{x})$ is also a continuous function. Second, because $\|\mathbf{x}\| < \infty$, we have $\mathbf{m}(\mathbf{x}) < \infty$ due to continuity. Therefore, it follows that $\mathbf{w}_i(\mathbf{x}) = \sigma(\mathbf{y}_i - \mathbf{y}_{i+1})\mathbf{m}_i(\mathbf{x})$ satisfies all the listed conditions. ■

To understand the differentiability of \mathbf{f} , we first see that \mathbf{f} may not be differentiable at a point where there is a tie among some elements of the input vector.

Corollary 5. *For \mathbf{w} in Corollary 4, there exists differentiable functions \mathbf{m} and σ such that $\mathbf{f}(\mathbf{x}) = S(\mathbf{x})^{-1}U\mathbf{w}(\mathbf{x})$ is not differentiable globally on \mathbb{R}^n .*

Proof. For the counter example, let $\mathbf{m} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ be a constant function $\mathbf{m}(\mathbf{x}) = [1, 1, 1]^\top$, and $\sigma(a) = a^2$. It is easy to verify that they both satisfy the conditions in Corollary 4 and are differentiable. We show that the partial derivative $\frac{\partial \mathbf{f}_1(\mathbf{x})}{\partial x_3}$ does not

exists at $\mathbf{x} = [2, 1, 1]^\top$. With few simple steps one could see $\mathbf{f}_1(\mathbf{x} + \alpha \mathbf{e}_3)$ for $\alpha \in (-\infty, 1]$ is

$$\mathbf{f}_1(\mathbf{x} + \alpha \mathbf{e}_3) = \begin{cases} \sigma(1) + \sigma(-\alpha) + 1 & \alpha \leq 0 \\ \sigma(1 - \alpha) + \sigma(\alpha) + 1 & 0 < \alpha \leq 1 \end{cases} \quad (\text{C.3})$$

Though this function is continuous, the left and right derivatives are not equal at $\alpha = 0$ so the function is not differentiable at $\mathbf{x} = [2, 1, 1]^\top$. ■

The above example shows that \mathbf{f} may not be differentiable for tied inputs. On the other hand, it is straightforward to see function \mathbf{f} is differentiable at points where there is no tie. More precisely, for the points with tie in the input vector, we show the function \mathbf{f} is B-differentiable, which is a weaker condition than the usual (Frechét) differentiability.

Definition 7. [Facchinei and Pang, 2007] A function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is said to be *B(ouligand)-differentiable* at a point $\mathbf{x} \in \mathbb{R}^n$, if \mathbf{f} is Lipschitz continuous in the neighborhood of \mathbf{x} and directionally differentiable at \mathbf{x} .

Proposition 3. For $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ in Theorem 1, let $\mathbf{w}(\mathbf{x})$ be as defined in Corollary 4. If σ and \mathbf{m} are continuously differentiable, then \mathbf{f} is B-differentiable on \mathbb{R}^n .

Proof. Let $\mathbb{P}^n = \{P_1, \dots, P_K\}$ be the finite set of all possible $n \times n$ dimensional permutation matrices. For each $k \in [K]$, define the closed set $\mathbb{N}_k = \{\mathbf{x} : S(\mathbf{x})\mathbf{x} = P_k\mathbf{x}\}$. These sets are convex polyhedrons since each can be defined by a finite set of linear inequalities; in addition, they together form a covering set of \mathbb{R}^n .

If there is no tie in elements of vector \mathbf{x} , then $\mathbf{x} \in \text{int}(\mathbb{N}_k)$ for some $k \in [K]$. Since the sorting function $S(\mathbf{x})$ has the constant value P_k in a small enough neighborhood of \mathbf{x} , the function \mathbf{f} is continuously differentiable (and therefore B-differentiable) at \mathbf{x} .

Next we show that, for any point $\mathbf{x} \in \mathbb{R}^n$ with some tied elements, the directional derivative of \mathbf{f} along an arbitrary direction $\mathbf{d} \in \mathbb{R}^n$ exists. For such \mathbf{x} and \mathbf{d} , there exists a $k \in [K]$ and a small enough $\delta > 0$ such that $\mathbf{x}, \mathbf{x} + \epsilon \mathbf{d} \in \mathbb{N}_k$ for all $0 \leq \epsilon \leq \delta$. Therefore, we have $\mathbf{f}(\mathbf{x}') = \hat{\mathbf{f}}(\mathbf{x}')$ for all $\mathbf{x}' \in [\mathbf{x}, \mathbf{x} + \delta \mathbf{d}]$, where $\hat{\mathbf{f}}_k(\mathbf{x}) = P_k^{-1} U D(P_k \mathbf{x}) \mathbf{m}(\mathbf{x})$. Let $\hat{\mathbf{f}}'_k(\mathbf{x}; \mathbf{d})$ denote the directional derivative of $\hat{\mathbf{f}}_k$ at \mathbf{x} along \mathbf{d} . By the equality of $\hat{\mathbf{f}}_k$ and \mathbf{f} in $[\mathbf{x}, \mathbf{x} + \delta \mathbf{d}]$, we conclude that the directional derivative $\mathbf{f}'(\mathbf{x}; \mathbf{d})$ exists and is equal to $\hat{\mathbf{f}}'_k(\mathbf{x}; \mathbf{d})$.

Finally, we note that \mathbf{f} is Lipschitz continuous, since it is composed by pieces of Lipschitz continuous functions $\hat{\mathbf{f}}_k$ for $k \in [K]$ (implied by the continuous differentiability assumption on σ and \mathbf{m}). Thus, \mathbf{f} is B-differentiable. ■

C.3 Reliability Diagrams

In Figure 6.5 of Chapter 6, we show the reliability diagrams and diagonal functions learned by TS and DIAG in ResNet 152 and PNASNet5 large on ImageNet dataset. Figure C.1 and C.2 illustrate the reliability diagrams for different calibration algorithms in all the models. In general DIAG method outperforms other methods in calibration in most of the regions. OP and OI methods also achieve good calibration

performance on this dataset and are slightly better than temperature scaling, while MS and DIR methods do not reduce the calibration error as much.

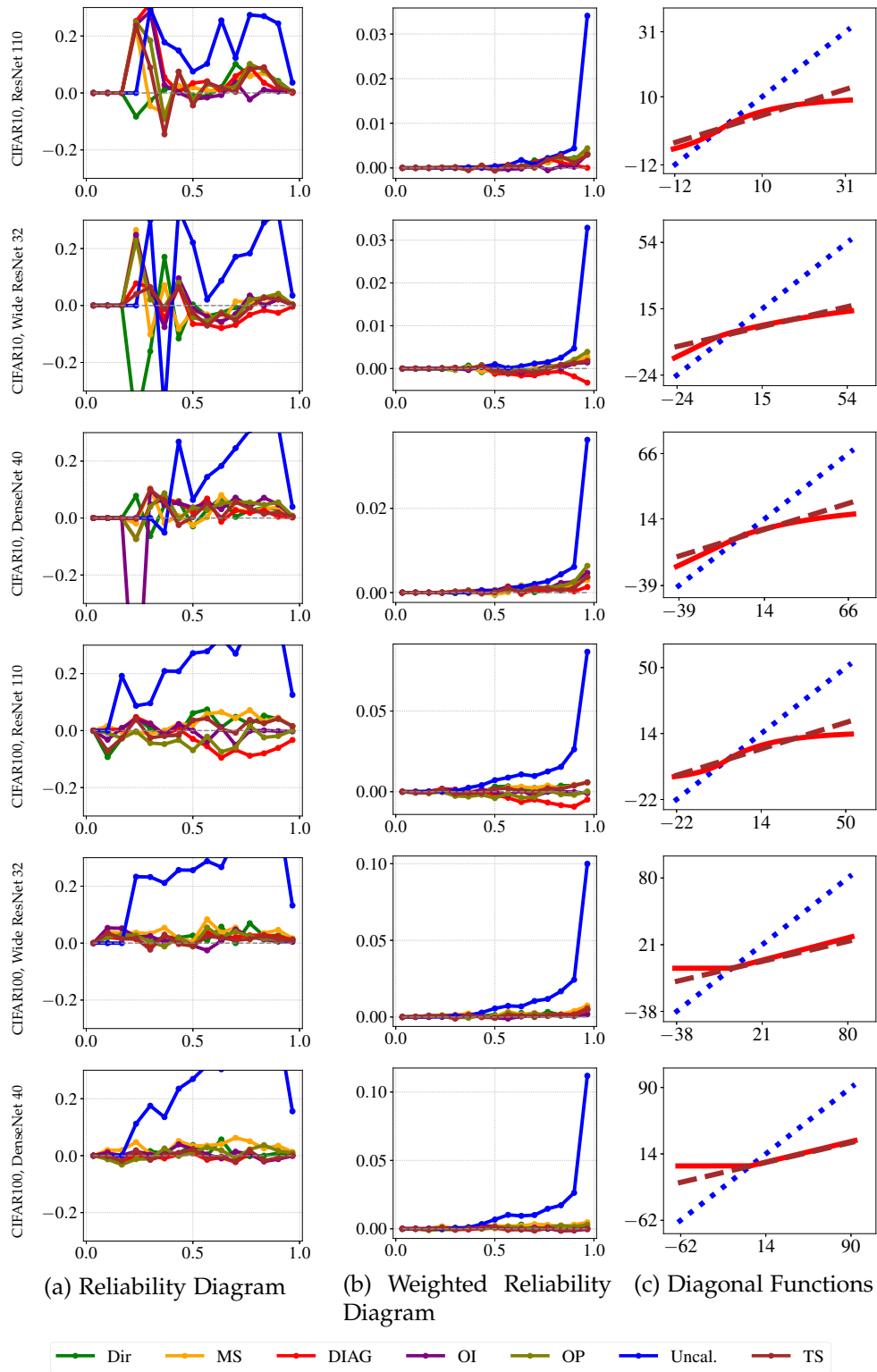


Figure C.1: Reliability diagrams and learned diagonal functions. See Figure 6.5 for the explanation of each diagram and axis.

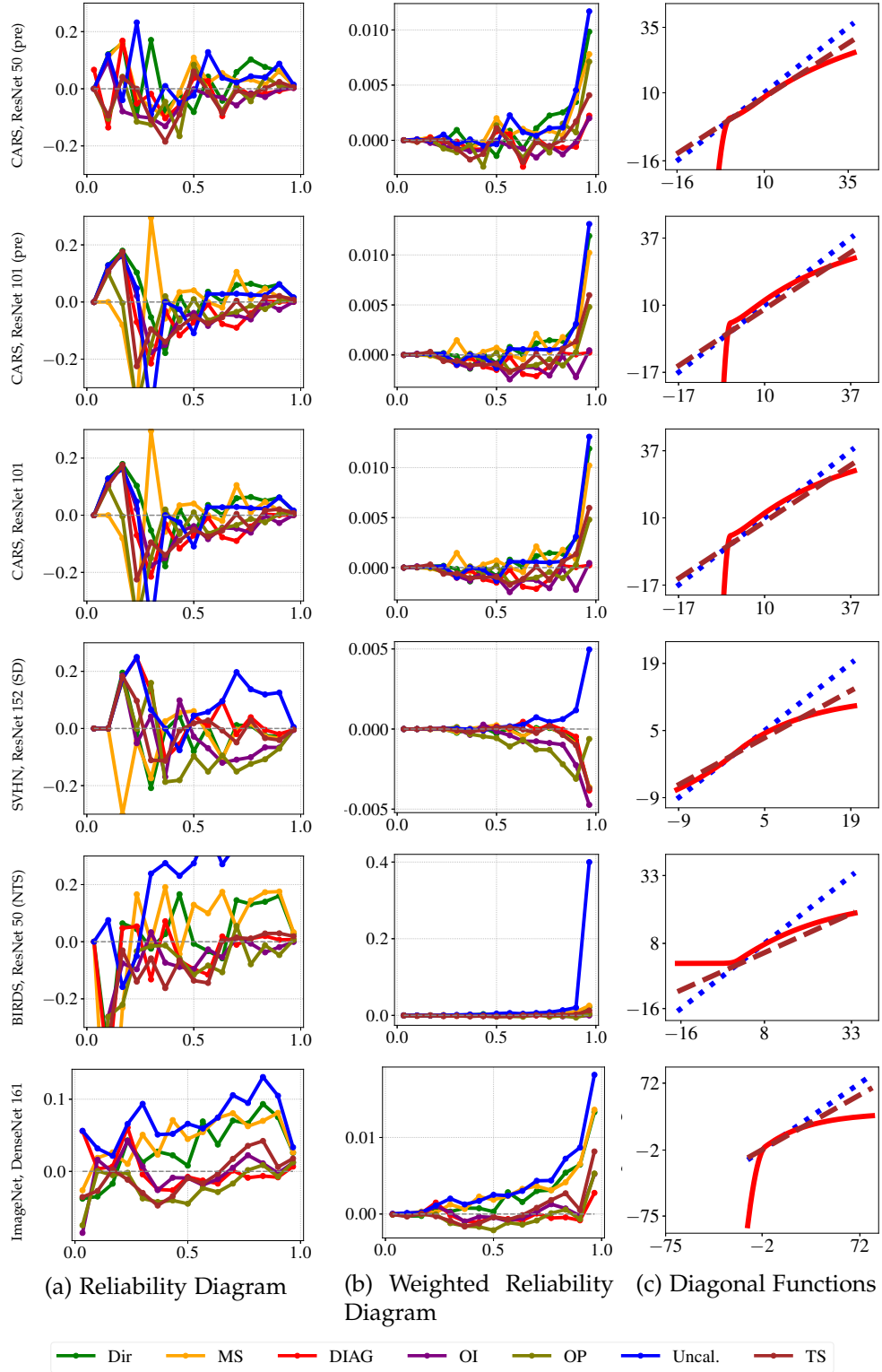


Figure C.2: Reliability diagrams and learned diagonal functions. See Figure 6.5 for the explanation of each diagram and axis.

Bibliography

- ADAMS, A.; BAEK, J.; AND DAVIS, M. A., 2010. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, vol. 29, 753–762. Wiley Online Library. (cited on page 79)
- AJANTHAN, T.; DESMAISON, A.; BUNEL, R.; SALZMANN, M.; TORR, P. H.; AND PAWAN KUMAR, M., 2017. Efficient linear programming for dense crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3298–3306. (cited on page 79)
- AMOS, B. AND KOLTER, J. Z., 2017. OptNet: Differentiable optimization as a layer in neural networks. In *ICML*. (cited on page 113)
- ANDRES, B.; BEIER, T.; AND KAPPES, J. H., 2012. OpenGM: A C++ library for discrete graphical models. *CoRR*, abs/1206.0111 (2012). (cited on page 31)
- ANDREWS, S.; TSOCHANTARIDIS, I.; AND HOFMANN, T., 2003. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems*, 577–584. (cited on pages 11, 12, 31, 33, 72, 74, and 75)
- ARUN, A.; JAWAHAR, C.; AND KUMAR, M. P., 2019. Dissimilarity coefficient based weakly supervised object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9432–9441. (cited on page 71)
- BABENKO, B.; YANG, M.-H.; AND BELONGIE, S., 2009. Visual tracking with online multiple instance learning. In *CVPR*, 983–990. IEEE. (cited on page 22)
- BANERJEE, A.; DHILLON, I. S.; GHOSH, J.; SRA, S.; AND RIDGEWAY, G., 2005. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, (2005). (cited on pages 47, 51, 52, and 53)
- BANSAL, A.; SIKKA, K.; SHARMA, G.; CHELLAPPA, R.; AND DIVAKARAN, A., 2018. Zero-shot object detection. In *Proceedings of the European Conference on Computer Vision*, 384–400. (cited on pages 34 and 81)
- BERGTHOLDT, M.; KAPPES, J.; SCHMIDT, S.; AND SCHNÖRR, C., 2010. A study of parts-based object class detection using complete graphs. *IJCV*, 87, 1-2 (2010), 93. (cited on pages 27, 31, 33, and 78)
- BESAG, J., 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 48, 3 (1986), 259–279. (cited on pages 6, 73, and 78)

- BILEN, H.; PEDERSOLI, M.; AND TUYTELAARS, T., 2014. Weakly supervised object detection with posterior regularization. In *British Machine Vision Conference*, vol. 3. (cited on pages 71 and 72)
- BILEN, H.; PEDERSOLI, M.; AND TUYTELAARS, T., 2015. Weakly supervised object detection with convex clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1081–1089. (cited on pages 16 and 35)
- BILEN, H. AND VEDALDI, A., 2016. Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2846–2854. (cited on pages 16 and 72)
- BLAKE, A.; KOHLI, P.; AND ROTHER, C., 2011. *Markov random fields for vision and image processing*. Mit Press. (cited on page 79)
- BRIER, G. W., 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78, 1 (1950), 1–3. (cited on pages 18 and 92)
- BUNESCU, R. C. AND MOONEY, R. J., 2007. Multiple instance learning for sparse positive bags. In *ICML*, 105–112. (cited on pages 31 and 33)
- CAO, Z.; SIMON, T.; WEI, S.-E.; AND SHEIKH, Y., 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*. (cited on page 89)
- CARUANA, R.; LOU, Y.; GEHRKE, J.; KOCH, P.; STURM, M.; AND ELHADAD, N., 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *ACM SIGKDD*, 1721–1730. (cited on page 89)
- CINBIS, R. G.; VERBEEK, J.; AND SCHMID, C., 2016. Weakly supervised object localization with multi-fold multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 39, 1 (2016), 189–203. (cited on pages 16, 35, 71, 72, 73, 74, and 83)
- CLENSHAW, C. W. AND CURTIS, A. R., 1960. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2, 1 (1960), 197–205. (cited on page 98)
- DEGROOT, M. H. AND FIENBERG, S. E., 1983. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32, 1-2 (1983), 12–22. (cited on page 18)
- DEHGHANI, M.; GOUWS, S.; VINYALS, O.; USZKOREIT, J.; AND KAISER, L., 2019. Universal transformers. In *ICLR*. (cited on page 111)
- DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; AND FEI-FEI, L., 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255. (cited on pages 44 and 98)

-
- DESELAERS, T.; ALEXE, B.; AND FERRARI, V., 2010. Localizing objects while learning their appearance. In *Proceedings of the European Conference on Computer Vision*, 452–466. Springer. (cited on pages 16, 35, 55, 71, 72, 73, 81, 88, and 113)
- DESELAERS, T.; ALEXE, B.; AND FERRARI, V., 2012. Weakly supervised localization and learning with generic knowledge. *International Journal of Computer Vision*, 100, 3 (2012), 275–293. (cited on page 31)
- DESELAERS, T. AND FERRARI, V., 2010. A conditional random field for multiple-instance learning. In *ICML*, 287–294. (cited on pages 23 and 31)
- DOERSCH, C.; GUPTA, A.; AND ZISSERMAN, A., 2020. Crosstransformers: spatially-aware few-shot transfer. *NeurIPS*, (2020). (cited on pages 13 and 111)
- EVERINGHAM, M.; VAN GOOL, L.; WILLIAMS, C. K. I.; WINN, J.; AND ZISSERMAN, A., 2007. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>. (cited on pages 44 and 56)
- FACCHINEI, F. AND PANG, J.-S., 2007. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media. (cited on page 124)
- FAKTOR, A. AND IRANI, M., 2013. Co-segmentation by composition. In *ICCV*, 1297–1304. (cited on page 22)
- FINN, C.; ABBEEL, P.; AND LEVINE, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. (cited on pages 13 and 27)
- FU, H.; XU, D.; ZHANG, B.; AND LIN, S., 2014. Object-based multiple foreground video co-segmentation. In *CVPR*, 3166–3173. (cited on page 22)
- GAL, Y. AND GHAHRAMANI, Z., 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 1050–1059. (cited on page 19)
- GAO, J.; WANG, J.; DAI, S.; LI, L.-J.; AND NEVATIA, R., 2019. Note-rcnn: Noise tolerant ensemble rcnn for semi-supervised object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 9508–9517. (cited on pages 16 and 71)
- GIDARIS, S. AND KOMODAKIS, N., 2018. Dynamic few-shot visual learning without forgetting. In *CVPR*, 4367–4375. (cited on pages 46 and 54)
- GIRSHICK, R., 2015. Fast r-cnn. In *ICCV*, 1440–1448. (cited on pages 35 and 89)
- GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; AND 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*, 580–587. (cited on page 83)

- GOKBERK CINBIS, R.; VERBEEK, J.; AND SCHMID, C., 2014. Multi-fold mil training for weakly supervised object localization. In *CVPR*. (cited on page 49)
- GUILLAUMIN, M. AND FERRARI, V., 2012. Large-scale knowledge transfer for object localization in imagenet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3202–3209. IEEE. (cited on pages 16, 72, and 75)
- GUO, C.; PLEISS, G.; SUN, Y.; AND WEINBERGER, K. Q., 2017. On calibration of modern neural networks. In *ICML*, 1321–1330. JMLR. org. (cited on pages 18, 90, 91, 92, 93, 95, 97, 99, and 100)
- HAJIMIRSADEGHI, H.; LI, J.; MORI, G.; ZAKI, M.; AND SAYED, T., 2013. Multiple instance learning by discriminative training of markov networks. In *UAI*, 262–271. (cited on page 23)
- HAYDER, Z.; HE, X.; AND SALZMANN, M., 2015. Structural kernel learning for large scale multiclass object co-detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2632–2640. IEEE. (cited on page 16)
- HAYDER, Z.; SALZMANN, M.; AND HE, X., 2014. Object co-detection via efficient inference in a fully-connected crf. In *Proceedings of the European Conference on Computer Vision*, 330–345. Springer. (cited on page 16)
- HE, K.; GKIOXARI, G.; DOLLÁR, P.; AND GIRSHICK, R., 2017. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, 2961–2969. (cited on page 71)
- HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. (cited on pages 35, 57, 81, 89, and 98)
- HOFFMAN, J.; PATHAK, D.; DARRELL, T.; AND SAENKO, K., 2015. Detector discovery in the wild: Joint multiple instance and representation learning. *CVPR*, (2015), 2883–2891. (cited on pages 31 and 33)
- HOFFMAN, J.; PATHAK, D.; TZENG, E.; LONG, J.; GUADARRAMA, S.; DARRELL, T.; AND SAENKO, K., 2016. Large scale visual recognition through adaptation using joint representation and multiple instance learning. *The Journal of Machine Learning Research*, 17, 1 (2016), 4954–4984. (cited on pages 16, 44, 55, 59, 72, 80, 83, 84, and 113)
- HSU, K.-J.; TSAI, C.-C.; LIN, Y.-Y.; QIAN, X.; AND CHUANG, Y.-Y., 2018. Unsupervised cnn-based co-saliency detection with graphical optimization. In *ECCV*. (cited on page 22)
- HU, T.; METTES, P.; HUANG, J.-H.; AND SNOEK, C. G., 2019. Silco: Show a few images, localize the common object. In *ICCV*. (cited on pages 17, 44, 55, and 56)
- HUANG, G.; LIU, Z.; VAN DER MAATEN, L.; AND WEINBERGER, K. Q., 2017a. Densely connected convolutional networks. In *CVPR*, 4700–4708. (cited on page 99)

-
- HUANG, J.; RATHOD, V.; SUN, C.; ZHU, M.; KORATTIKARA, A.; FATHI, A.; FISCHER, I.; WOJNA, Z.; SONG, Y.; GUADARRAMA, S.; ET AL., 2017b. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 7310–7311. (cited on pages 35 and 36)
- ILSE, M.; TOMCZAK, J.; AND WELLING, M., 2018. Attention-based deep multiple instance learning. In *ICML*, 2127–2136. (cited on pages 31, 33, and 50)
- JIANG, X.; OSL, M.; KIM, J.; AND OHNO-MACHADO, L., 2012. Calibrating predictive model estimates to support personalized medicine. *Journal of the American Medical Informatics Association*, 19, 2 (2012), 263–274. (cited on page 89)
- KANG, B.; LIU, Z.; WANG, X.; YU, F.; FENG, J.; AND DARRELL, T. R., 2019. Few-shot object detection via feature reweighting. In *ICCV*. (cited on page 56)
- KENDALL, A. AND GAL, Y., 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 5574–5584. (cited on page 90)
- KIM, D.; LEE, G.; JEONG, J.; AND KWAK, N., 2020. Tell me what they’re holding: Weakly-supervised object detection with transferable knowledge from human-object interaction. In *AAAI*. (cited on page 42)
- KINGMA, D. P. AND BA, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, (2014). (cited on page 99)
- KOLMOGOROV, V., 2006. Convergent tree-reweighted message passing for energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 28, 10 (2006), 1568–1583. (cited on pages 27, 31, 33, 73, 78, and 82)
- KRÄHENBÜHL, P. AND KOLTUN, V., 2011. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems*, 109–117. (cited on page 79)
- KRAUSE, J.; STARK, M.; DENG, J.; AND FEI-FEI, L., 2013. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia. (cited on page 98)
- KRIZHEVSKY, A.; HINTON, G.; ET AL., 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto. (cited on page 98)
- KRIZHEVSKY, A.; SUTSKEVER, I.; AND HINTON, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1097–1105. (cited on pages 81 and 83)
- KULL, M.; NIETO, M. P.; KÄNGSEPP, M.; SILVA FILHO, T.; SONG, H.; AND FLACH, P., 2019. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with Dirichlet calibration. In *NeurIPS*, 12295–12305. (cited on pages 18, 90, 91, 92, 93, 97, 99, 100, and 105)

-
- KULL, M.; SILVA FILHO, T.; AND FLACH, P., 2017a. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Artificial Intelligence and Statistics*, 623–631. (cited on pages 18, 90, and 93)
- KULL, M.; SILVA FILHO, T. M.; FLACH, P.; ET AL., 2017b. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. *Electronic Journal of Statistics*, 11, 2 (2017), 5052–5080. (cited on pages 18 and 90)
- KUMAR, A.; LIANG, P.; AND MA, T., 2019. Verified uncertainty calibration. In *NeurIPS*. (cited on pages xxii, 18, 92, 102, 103, 107, and 108)
- KUMAR, A.; SARAWAGI, S.; AND JAIN, U., 2018. Trainable calibration measures for neural networks from kernel mean embeddings. In *ICML*, 2805–2814. (cited on page 19)
- KUMAR, M. P.; PACKER, B.; AND KOLLER, D., 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, 1189–1197. (cited on page 16)
- LEE, K.; MAJI, S.; RAVICHANDRAN, A.; AND SOATTO, S., 2019. Meta-learning with differentiable convex optimization. In *CVPR*. (cited on page 44)
- LI, W.; JAFARI, O. H.; AND ROTHER, C., 2018. Deep object co-segmentation. In *ACCV*. (cited on page 17)
- LI, Y.; LIU, L.; SHEN, C.; AND VAN DEN HENGEL, A., 2016. Image co-localization by mimicking a good detector’s confidence score distribution. In *ECCV*, 19–34. (cited on page 17)
- LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; AND DOLLÁR, P., 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988. (cited on page 72)
- LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; AND ZITNICK, C. L., 2014. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, 740–755. (cited on pages 34, 44, 56, 73, and 81)
- LIU, C.; ZOPH, B.; NEUMANN, M.; SHLENS, J.; HUA, W.; LI, L.; FEI-FEI, L.; YUILLE, A. L.; HUANG, J.; AND MURPHY, K., 2018. Progressive neural architecture search. In *ECCV*. (cited on page 99)
- LIU, D. C. AND NOCEDAL, J., 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45, 1-3 (1989), 503–528. (cited on page 99)
- LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; AND BERG, A. C., 2016. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, 21–37. Springer. (cited on pages 72 and 83)

-
- LIU, Y.; LEE, J.; PARK, M.; KIM, S.; YANG, E.; HWANG, S. J.; AND YANG, Y., 2019. Learning to propagate labels: Transductive propagation network for few-shot learning. *ICLR*, (2019). (cited on page 13)
- LU, X.; WANG, W.; MA, C.; SHEN, J.; SHAO, L.; AND PORIKLI, F., 2019. See more, know more: Unsupervised video object segmentation with co-attention siamese networks. In *CVPR*. (cited on page 42)
- MADDOX, W. J.; IZMAILOV, P.; GARIPPOV, T.; VETROV, D. P.; AND WILSON, A. G., 2019. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*, 13132–13143. (cited on pages xx, 19, 90, and 102)
- MARDIA, K. V. AND JUPP, P. E., 2009. *Directional statistics*. John Wiley & Sons. (cited on page 52)
- MARON, O. AND LOZANO-PÉREZ, T., 1998. A framework for multiple-instance learning. In *NIPS*, 570–576. (cited on page 50)
- MISHRA, N.; ROHANINEJAD, M.; CHEN, X.; AND ABBEEL, P., 2018. A simple neural attentive meta-learner. In *ICLR*. (cited on page 13)
- MOZAFARI, A. S.; GOMES, H. S.; LEÃO, W.; AND GAGNÉ, C., 2019. Unsupervised temperature scaling: Post-processing unsupervised calibration of deep models decisions. In *ICML Workshop*. (cited on page 89)
- MÜLLER, R.; KORNBLITH, S.; AND HINTON, G. E., 2019. When does label smoothing help? In *NeurIPS*, 4696–4705. (cited on page 19)
- MURPHY, A. H. AND WINKLER, R. L., 1977. Reliability of subjective probability forecasts of precipitation and temperature. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 26, 1 (1977), 41–47. (cited on page 18)
- NAEINI, M. P.; COOPER, G.; AND HAUSKRECHT, M., 2015. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*. (cited on page 92)
- NETZER, Y.; WANG, T.; COATES, A.; BISSACCO, A.; WU, B.; AND NG, A. Y., 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf. (cited on page 98)
- NGUYEN, M. H.; TORRESANI, L.; DE LA TORRE, F.; AND ROTHER, C., 2009. Weakly supervised discriminative localization and classification: a joint learning process. In *ICCV*. (cited on page 49)
- NIXON, J.; DUSENBERRY, M.; ZHANG, L.; JERFEL, G.; AND TRAN, D., 2019. Measuring calibration in deep learning. *arXiv preprint arXiv:1904.01685*, (2019). (cited on pages 92 and 107)

-
- ORTEGA, J. M. AND RHEINBOLDT, W. C., 1970. *Iterative solution of nonlinear equations in several variables*, vol. 30. Siam. (cited on page 75)
- OVADIA, Y.; FERTIG, E.; REN, J.; NADO, Z.; SCULLEY, D.; NOWOZIN, S.; DILLON, J.; LAKSHMINARAYANAN, B.; AND SNOEK, J., 2019. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*. (cited on page 105)
- PEREYRA, G.; TUCKER, G.; CHOROWSKI, J.; KAISER, Ł.; AND HINTON, G., 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, (2017). (cited on page 19)
- PEREZ-RUA, J.-M.; ZHU, X.; HOSPEDALES, T. M.; AND XIANG, T., 2020. Incremental few-shot object detection. In *CVPR*. (cited on pages 42 and 56)
- PLATT, J., 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10, 3 (1999), 61–74. (cited on pages 18 and 90)
- QI, H.; BROWN, M.; AND LOWE, D. G., 2018. Low-shot learning with imprinted weights. In *CVPR*. (cited on pages 46 and 54)
- RAHIMI, A.; SHABAN, A.; AJANTHAN, T.; HARTLEY, R.; AND BOOTS, B., 2020a. Pairwise similarity knowledge transfer for weakly supervised object localization. In *ECCV*. (cited on pages 7, 48, 49, 55, and 113)
- RAHIMI, A.; SHABAN, A.; CHENG, C.-A.; HARTLEY, R.; AND BOOTS, B., 2020b. Intra order-preserving functions for calibration of multi-class neural networks. In *NeurIPS*. (cited on page 7)
- RAVI, S. AND LAROCHELLE, H., 2017. Optimization as a model for few-shot learning. In *ICLR*. (cited on pages xxi, 13, and 31)
- REDMON, J.; DIVVALA, S.; GIRSHICK, R.; AND FARHADI, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788. (cited on page 71)
- REN, S.; HE, K.; GIRSHICK, R.; AND SUN, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 91–99. (cited on pages xv, 14, 15, 45, 80, 81, 83, and 89)
- ROCHAN, M. AND WANG, Y., 2015. Weakly supervised localization of novel objects using appearance transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4315–4324. (cited on pages 16, 72, and 75)
- RODRÍGUEZ, P.; LARADJI, I.; DROUIN, A.; AND LACOSTE, A., 2020. Embedding propagation: Smoother manifold for few-shot classification. *ECCV*, (2020). (cited on page 13)

-
- RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; AND FEI-FEI, L., 2015. Imagenet large scale visual recognition challenge. *IJCV*, 115, 3 (2015), 211–252. (cited on pages 34, 73, and 83)
- SANTORO, A.; BARTUNOV, S.; BOTVINICK, M.; WIERSTRA, D.; AND LILICRAP, T., 2016. Meta-learning with memory-augmented neural networks. In *ICML*, 1842–1850. (cited on page 13)
- SAVCHYNSKYI, B. ET AL., 2019. Discrete graphical models – An optimization perspective. *Foundations and Trends® in Computer Graphics and Vision*, 11, 3-4 (2019), 160–429. (cited on pages 78 and 80)
- SCHRIJVER, A., 1998. *Theory of linear and integer programming*. John Wiley & Sons. (cited on page 77)
- SEO, S.; SEO, P. H.; AND HAN, B., 2019. Learning for single-shot confidence calibration in deep neural networks through stochastic inferences. In *CVPR*, 9030–9038. (cited on pages 19 and 97)
- SHABAN, A.; RAHIMI, A.; AJANTHAN, T.; BOOTS, B.; AND HARTLEY, R., 2022. Few-shot weakly-supervised object detection via directional statistics. In *WACV*. (cited on page 7)
- SHABAN, A.; RAHIMI, A.; BANSAL, S.; GOULD, S.; BOOTS, B.; AND HARTLEY, R., 2019. Learning to find common objects across few image collections. In *Proceedings of the IEEE International Conference on Computer Vision*, 5117–5126. (cited on pages xvi, 7, 16, 22, 23, 24, 25, 26, 27, 30, 31, 33, 35, 37, 38, 39, 44, and 81)
- SHI, M.; CAESAR, H.; AND FERRARI, V., 2017. Weakly supervised object localization using things and stuff transfer. *ICCV*, (2017), 3401–3410. (cited on page 35)
- SIAM, M.; DORAISWAMY, N.; ORESHKIN, B. N.; YAO, H.; AND JAGERSAND, M., 2020. Weakly supervised few-shot object segmentation using co-attention with visual and semantic embeddings. *IJCAI*, (2020). (cited on page 17)
- SIMONYAN, K. AND ZISSERMAN, A., 2015. Very deep convolutional networks for large-scale image recognition. *ICLR*, (2015). (cited on page 56)
- SINGH, B.; NAJIBI, M.; AND DAVIS, L. S., 2018. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, 9310–9320. (cited on page 72)
- SNELL, J.; SWERSKY, K.; AND ZEMEL, R., 2017. Prototypical networks for few-shot learning. In *NIPS*, 4077–4087. (cited on pages xv, 13, 14, 27, 44, 46, 48, and 49)
- SUNG, F.; YANG, Y.; ZHANG, L.; XIANG, T.; TORR, P. H.; AND HOSPEDALES, T. M., 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1199–1208. (cited on pages xv, 13, 14, 23, and 25)

-
- SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V.; AND ALEMI, A. A., 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*. (cited on page 83)
- SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; AND WOJNA, Z., 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826. (cited on page 83)
- TANG, K.; JOULIN, A.; LI, L.-J.; AND FEI-FEI, L., 2014. Co-localization in real-world images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1464–1471. (cited on pages 16 and 72)
- TANG, P.; WANG, X.; BAI, S.; SHEN, W.; BAI, X.; LIU, W.; AND YUILLE, A., 2018. Pcl: Proposal cluster learning for weakly supervised object detection. *IEEE transactions on pattern analysis and machine intelligence*, 42, 1 (2018), 176–191. (cited on page 71)
- TANG, P.; WANG, X.; BAI, X.; AND LIU, W., 2017. Multiple instance detection network with online instance classifier refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (cited on page 71)
- TANG, Y.; WANG, J.; GAO, B.; DELLANDRÉA, E.; GAIZAUSKAS, R.; AND CHEN, L., 2016. Large scale semi-supervised object detection using visual and semantic knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2119–2128. (cited on pages 80 and 83)
- THULASIDASAN, S.; CHENNUPATI, G.; BILMES, J. A.; BHATTACHARYA, T.; AND MICHALAK, S., 2019. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, 13888–13899. (cited on pages 18 and 97)
- TIAN, Y.; WANG, Y.; KRISHNAN, D.; TENENBAUM, J. B.; AND ISOLA, P., 2020. Rethinking few-shot image classification: a good embedding is all you need? In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV* 16, 266–282. Springer. (cited on pages 4 and 13)
- TUKEY, J. W., 1977. *Exploratory data analysis*. Addison-Wesley Series in Behavioral Science. Addison-Wesley, Reading, MA. (cited on pages 54 and 59)
- UIJLINGS, J.; POPOV, S.; AND FERRARI, V., 2018. Revisiting knowledge transfer for training object class detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1101–1110. (cited on pages 16, 17, 35, 44, 49, 55, 56, 58, 59, 71, 72, 73, 75, 80, 82, 83, 84, 113, and 114)
- UIJLINGS, J. R.; VAN DE SANDE, K. E.; GEVERS, T.; AND SMEULDERS, A. W., 2013. Selective search for object recognition. *International Journal of Computer Vision*, 104, 2 (2013), 154–171. (cited on page 83)
- VAN DER MAATEN, L. AND HINTON, G., 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9, 86 (2008), 2579–2605. <http://jmlr.org/papers/v9/vandermaaten08a.html>. (cited on page 46)

-
- VICENTE, S.; ROTHER, C.; AND KOLMOGOROV, V., 2011. Object cosegmentation. In *CVPR*, 2217–2224. (cited on pages 2, 17, 22, 23, and 31)
- VINYALS, O.; BLUNDELL, C.; LILLICRAP, T.; WIERSTRA, D.; ET AL., 2016. Matching networks for one shot learning. In *NIPS*, 3630–3638. (cited on pages 13 and 31)
- WAN, F.; LIU, C.; KE, W.; JI, X.; JIAO, J.; AND YE, Q., 2019. C-mil: Continuation multiple instance learning for weakly supervised object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2199–2208. (cited on pages 16, 71, 72, and 73)
- WAN, F.; WEI, P.; JIAO, J.; HAN, Z.; AND YE, Q., 2018. Min-entropy latent model for weakly supervised object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1297–1306. (cited on page 16)
- WANG, F.; XIANG, X.; CHENG, J.; AND YUILLE, A. L., 2017. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, 1041–1049. (cited on page 47)
- WANG, X.; HUANG, T. E.; DARRELL, T.; GONZALEZ, J. E.; AND YU, F., 2020. Frustratingly simple few-shot object detection. *ICML*, (2020). (cited on pages 4, 42, 54, and 56)
- WANG, Y.-X.; RAMANAN, D.; AND HEBERT, M., 2019. Meta-learning to detect rare objects. In *ICCV*. (cited on page 42)
- WEHENKEL, A. AND LOUPPE, G., 2019. Unconstrained monotonic neural networks. In *NeurIPS*, 1543–1553. (cited on page 98)
- WEISS, Y. AND FREEMAN, W. T., 2001. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47, 2 (2001), 736–744. (cited on pages 27 and 78)
- WELINDER, P.; BRANSON, S.; MITA, T.; WAH, C.; SCHROFF, F.; BELONGIE, S.; AND PERONA, P., 2010. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology. (cited on page 98)
- WINKLER, R. L. AND MURPHY, A. H., 1968. “Good” probability assessors. *Journal of applied Meteorology*, 7, 5 (1968), 751–758. (cited on page 18)
- XIAO, Y. AND MARLET, R., 2020. Few-shot object detection and viewpoint estimation for objects in the wild. *ECCV*, (2020). (cited on pages 42 and 56)
- XING, C.; ARIK, S.; ZHANG, Z.; AND PFISTER, T., 2020. Distance-based learning from errors for confidence calibration. In *ICLR*. (cited on page 97)
- YAN, X.; CHEN, Z.; XU, A.; WANG, X.; LIANG, X.; AND LIN, L., 2019. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *ICCV*. (cited on page 56)

- YANG, P.; HU, V. T.; METTES, P.; AND SNOEK, C. G., 2020. Localizing the common action among a few videos. In *European Conference on Computer Vision*, 505–521. Springer. (cited on page 2)
- YANG, S.; LIU, L.; AND XU, M., 2021. Free lunch for few-shot learning: Distribution calibration. *ICLR*, (2021). (cited on pages 13 and 54)
- YANG, Z.; LUO, T.; WANG, D.; HU, Z.; GAO, J.; AND WANG, L., 2018. Learning to navigate for fine-grained classification. In *ECCV*. (cited on page 100)
- YUN, S.; HAN, D.; OH, S. J.; CHUN, S.; CHOE, J.; AND YOO, Y., 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 6023–6032. (cited on page 18)
- ZADROZNY, B. AND ELKAN, C., 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, vol. 1, 609–616. Citeseer. (cited on pages 18 and 90)
- ZADROZNY, B. AND ELKAN, C., 2002. Transforming classifier scores into accurate multiclass probability estimates. In *ACM SIGKDD*, 694–699. (cited on pages 18 and 90)
- ZAGORUYKO, S. AND KOMODAKIS, N., 2016. Wide residual networks. In *BMVC*. (cited on pages 31 and 98)
- ZHANG, D.; HAN, J.; LI, C.; AND WANG, J., 2015. Co-saliency detection via looking deep and wide. In *CVPR*, 2994–3002. (cited on pages 17 and 22)
- ZHANG, H.; CISSÉ, M.; DAUPHIN, Y. N.; AND LOPEZ-PAZ, D., 2018. Mixup: Beyond empirical risk minimization. In *ICLR*. (cited on page 18)
- ZHANG, J.; KAILKHURA, B.; AND HAN, T., 2020. Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *ICML*. (cited on page 98)
- ZHU, Y.; ZHOU, Y.; YE, Q.; QIU, Q.; AND JIAO, J., 2017. Soft proposal networks for weakly supervised object localization. In *Proceedings of the IEEE International Conference on Computer Vision*, 1841–1850. (cited on page 16)