Acknowledgments

I would like to express my sincere appreciation to my fantastic mentors, colleagues, friends, and family for their generous contributions to the work presented in this thesis and toward completing my Ph.D. degree.

Special mention goes to my enthusiastic adviser, Prof. Fei Sha. My Ph.D. has been an amazing five-year journey, and I thank Fei for his tremendous academic guidance and support, giving me many wonderful opportunities to explore and grow. As a mentor, Fei is knowledgeable and inspiring, always motivating me to think out of the box and challenge myself to the limit. As a research scientist, Fei has an extremely high standard and never stop pursuing impactful work, from which I can always derive “gradients” to improve. It was my great honor to work with him and follow his career path to become a faculty, and I am especially thankful for the excellent support and suggestions he has provided during my job hunting season.

Similarly, profound gratitude goes to my long-term collaborators and mentors Prof. Kristen Grauman and Dr. Boqing Gong. Kristen is a deep thinker, who can always point out key aspects to raise our work to the next level. She helped me set up a productive and solid path of research in my first half of Ph.D., and was enthusiastically supportive when I was on the job market. Boqing is my very best collaborator, friend, and teacher. It is he who step-by-step guided me on every aspect of doing research. I will never forget those paper deadlines we faced together and how we used to minimize the overlap of sleeping time to maintain progress. I am greatly inspired and motivated by watching him from being a hardworking senior Ph.D. to a productive young faculty. Like Fei and Kristen, Boqing offered tremendous support throughout my job search. Thank you.

I thank Prof. Winston Hsu and Prof. Jian-Jiun Ding, who led me into research and prepared me for the program when I was a M.S. student in NTU, Taiwan. My Ph.D. journey would not have started without them.

I would like to thank my dissertation defense committee members, Prof. Laurent Itti, Jason Lee, Panayiotis Georgiou, and Joseph Lim, for their interest, valuable time, and helpful comments. The same appreciation goes to my proposal committee members, Prof. Antonio Ortega, Haipeng Luo, and Meisam Razaviyayn. Especially Haipeng, Meisam, and Joseph set up mock interviews and job talks for me, greatly improving my preparation for job interviews. I also want to thank Lizsl De Leon and Jennifer Gerson for their professional services and excellent consultations at the CS department and Viterbi school to resolve many issues during my Ph.D.

My sincere gratitude also goes to Prof. Justin Solomon and Dominik Michels, and Dr. Hoifung Poon, Kris Quirk, and Xiaodong He, who provided me opportunities to collaborate on different projects, greatly expanding my research horizon and interest. The experience and skills I learned from them definitely facilitated my thesis work. I would specifically thank Justin for his help throughout my job search.

During my Ph.D., I was so fortunate to join the TEDS lab and have many talented and fun lab mates to work and live with. Kuan and Ali brought me many precious memories at my first half
of Ph.D.; the same appreciation goes to Dong, Yuan, Franziska, and Wenzhe. Thank Zhiyun for bringing me a great number of enjoyable moments at the lab, and thank Chao-Kai for many inspiring theoretical discussions. I also enjoyed chatting about lives with both of them. I want to give special thanks to Beer, Hexing (Frank), and Ke for their hardworking while we collaborated on zero-shot learning, visual question answering, and video summarization, respectively. Beer always works hard, thinks hard, plays hard, and sleeps hard. He cares a lot about people and everything but his daily routine. Frank has endless energy and passion for research and acquiring knowledge, and Ke is always humble and thoughtful. It was a tremendously productive period when working with them. Besides, I was so grateful to meet many new members at my final year, including Seb, Shariq, Aaron, Liyu, Ivy, Bowen, Chin-Cheng (Jeremy), Yuri, Yiming, Melissa, Han-Jia, and Marc. I learned so much from them, not only about research but about cultures and lifestyles. I also thank them for giving me opportunities to practice being a mentor. It was my great pleasure to discuss and work with so many gifted colleagues on diverse research problems, and thank all of them for helping me improve my job talks and defense presentation.

Finally, but by no means least, thanks to my family for all their love and encouragement. This thesis is dedicated to them. My parents are always supportive of me and my younger sister Yung-Hsuan for every decision we made, especially for studying abroad. It is amazing that we both studied and obtained the Ph.D. degree at USC, and attended the commencement in 2018. My Ph.D. life was with many up and down moments, and I really thank their company to share my happiness and depression. In the end, I would like to thank my loving, encouraging, patient, and supportive wife Feng-Ju (Claire). We have been life partners for seven years since our M.S. studies in Taiwan and it is she who brings me love and joy, teaches me how to pursue and fulfill dreams and never giving up, and makes me the most fortunate person. Thank you.
# Table of Contents

## Acknowledgments

## List of Tables

## List of Figures

## Abstract

## I Background

### 1 Introduction

1.1 Machine learning for intelligent systems ........................................ 3
1.2 Challenges in the wild ............................................................... 4
1.3 Transfer learning for intelligent systems ....................................... 5
1.4 Contributions and outline ......................................................... 7
1.5 Published work ........................................................................... 7
  1.5.1 Zero-shot learning ................................................................. 7
  1.5.2 Domain generalization for visual question answering .................. 7
  1.5.3 Other work ........................................................................... 8

## II Zero-shot Learning

### 2 Introduction to Zero-shot Learning

2.1 Definition .................................................................................. 10
  2.1.1 Notations ........................................................................... 11
  2.1.2 Problem formulations ......................................................... 12
  2.1.3 The framework of algorithms .............................................. 12
2.2 Challenges ................................................................................ 12
2.3 Contributions ............................................................................ 13
2.4 Outline of Part II ....................................................................... 13

### 3 Literature Survey on Zero-Shot Learning

3.1 Semantic representations .......................................................... 15
  3.1.1 Visual attributes ................................................................ 15
5.4.4 Which zero-shot learning approach is more robust to GZSL? ........ 47
5.5 Summary ........................................................................... 49

6 From Zero-Shot Learning to Conventional Supervised Learning 50
6.1 Comparisons among different learning paradigms .................... 50
6.2 Empirical studies ............................................................. 51
6.2.1 Overview ................................................................. 51
6.2.2 Setup ....................................................................... 52
6.2.3 Results ..................................................................... 52
6.3 Summary ........................................................................... 54

7 Improving Semantic Representations by Predicting Visual Exemplars (EXEM) 55
7.1 Approach ........................................................................ 55
7.1.1 Learning to predict the visual exemplars from the semantic representations 56
7.1.2 Zero-shot learning based on the predicted visual exemplars ........ 56
7.1.2.1 Predicted exemplars as training data ....................... 56
7.1.2.2 Predicted exemplars as the ideal semantic representations .. 56
7.1.3 Comparison to related approaches .................................. 57
7.2 Other details ................................................................. 57
7.3 Empirical studies ........................................................... 58
7.3.1 Setup ....................................................................... 58
7.3.2 Implementation details ............................................... 58
7.3.3 Predicted visual exemplars ........................................... 59
7.3.4 Results on the conventional setting ............................... 61
7.3.4.1 Main results .......................................................... 61
7.3.4.2 Large-scale zero-shot classification results ............. 62
7.3.4.3 Analysis .................................................................. 64
7.3.5 Results on the generalized setting ............................... 64
7.4 Summary ........................................................................... 65

III Domain Generalization for Visual Question Answering 66

8 Introduction to Visual Question Answering and Its Challenges 67
8.1 Review of existing work on Visual QA ................................. 68
8.1.1 Datasets ................................................................... 68
8.1.2 Evaluation ............................................................... 69
8.1.3 Algorithms .............................................................. 69
8.1.4 Analysis ................................................................. 69
8.2 Challenges ..................................................................... 70
8.3 Contributions and outline of Part III .................................. 70

9 Creating Better Visual Question AnsweringDatasets 73
9.1 Overview ....................................................................... 73
9.2 Related work .................................................................. 74
9.3 Analysis of decoy answers’ effects ................................... 75
List of Tables

4.1 Key characteristics of studied datasets ............................................. 32
4.2 Comparison between our results and the previously published results in multi-
way classification accuracies (in %) on the task of zero-shot learning. For each
dataset, the best is in red and the 2nd best is in blue. .......................... 34
4.3 Comparison between results by ConSE and our method on ImageNet. For both
types of metrics, the higher the better. ............................................ 34
4.4 Comparison between sample- and class-wise cross-validation for hyper-parameter
tuning on CUB (learning with the one-versus-other loss). ....................... 35
4.5 Detailed analysis of various methods: the effect of feature and attribute types on
multi-way classification accuracies (in %). Within each column, the best is in red
and the 2nd best is in blue. We cite both previously published results (numbers in
bold italics) and results from our implementations of those competing methods
(numbers in normal font) to enhance comparability and to ease analysis (see texts
for details). We use the shallow features provided by [106], [83], [142] for AwA,
CUB, SUN, respectively. ................................................................. 36
4.6 Effect of types of semantic representations on AwA. ............................ 37
4.7 Effect of learning semantic representations ........................................ 37
5.1 Classification accuracies (%) on conventional ZSL ($A_{U\rightarrow U}$), multi-class classi-
fication for seen classes ($A_{S\rightarrow S}$), and GZSL ($A_{S\rightarrow T}$ and $A_{U\rightarrow T}$), on AwA and
CUB. Significant drops are observed from $A_{U\rightarrow U}$ to $A_{U\rightarrow T}$. .............. 43
5.2 Comparison of performance measured in AUSUC between two cross-validation
strategies on AwA and CUB. One strategy is based on accuracies ($A_{S\rightarrow S}$ and
$A_{U\rightarrow U}$) and the other is based on AUSUC. See text for details. .................. 47
5.3 Performances measured in AUSUC of several methods for Generalized Zero-Shot
Learning on AwA and CUB. The higher the better (the upper bound is 1). ....... 48
5.4 Performances measured in AUSUC by different zero-shot learning approaches on
GZSL on ImageNet, using our method of calibrated stacking. .................... 48
6.1 Comparison of performances measured in AUSUC between GZSL (using word2vec
and G-attr) and multi-class classification on ImageNet-2K. Few-shot results
are averaged over 100 rounds. GZSL with G-attr improves upon GZSL with
word2vec significantly and quickly approaches multi-class classification per-
formance. .................................................................................... 53
6.2 Comparison of performances measured in AUSUC between GZSL with word2vec
and GZSL with G-attr on the full ImageNet with over 20,000 unseen classes.
Few-shot results are averaged over 20 rounds. .................................... 54
7.1 We compute the Euclidean distance matrix between the unseen classes based on semantic representations ($D_{a_u}$), predicted exemplars ($D_{\psi(a_u)}$), and real exemplars ($D_{v_u}$). Our method leads to $D_{\psi(a_u)}$ that is better correlated with $D_{v_u}$ than $D_{a_u}$ is. See text for more details. 

7.2 Comparison between existing ZSL approaches in multi-way classification accuracies (in %) on four benchmark datasets. For each dataset, we mark the best in red and the second best in blue. *Italic* numbers denote *per-sample* accuracy instead of *per-class* accuracy. On ImageNet, we report results for both types of semantic representations: Word vectors (wv) and MDS embeddings derived from WordNet (hie). All the results are based on GoogLeNet features [175].

7.3 Comparison between existing ZSL approaches on ImageNet using *word vectors* of the class names as semantic representations. For both metrics (in %), the higher the better. The best is in red. The numbers of unseen classes are listed in parentheses. †: our implementation.

7.4 Comparison between existing ZSL approaches on ImageNet (with 20,842 unseen classes) using *MDS embeddings derived from WordNet* [123] as semantic representations. The higher, the better (in %). The best is in red.

7.5 Accuracy of *exem (1NN)* on AwA, CUB, and SUN when predicted exemplars are from original visual features (No PCA) and PCA-projected features (PCA with $d = 1024$ and $d = 500$).

7.6 Comparison between *exem (1NN)* with support vector regressors (SVR) and with 2-layer multi-layer perceptron (MLP) for predicting visual exemplars. Results on CUB are for the first split. Each number for MLP is an average over 3 random initialization.

7.7 Generalized ZSL results in Area Under Seen-Unseen accuracy Curve (AUSUC) on AwA, CUB, and SUN. For each dataset, we mark the best in red and the second best in blue. All approaches use GoogLeNet as the visual features and calibrated stacking to combine the scores for seen and unseen classes.

9.1 Accuracy of selecting the right answers out of 4 choices (%) on the Visual QA task on Visual7W.

9.2 Summary of Visual QA datasets.

9.3 Test accuracy (%) on Visual7W.

9.4 Accuracy (%) on the validation set in VQA.

9.5 Test accuracy (%) on qaVG.

9.6 Using models trained on qaVG to improve Visual7W and VQA (Accuracy in %).

9.7 Accuracy (%) on VQA−2014val, which contains 76,034 triplets.

9.8 Test accuracy (%) on COCOQA.

9.9 Test accuracy (%) on VQA2-2017val.

9.10 Test accuracy (%) on VQA2−2017val, which contains 134,813 triplets.

9.11 Test accuracy (%) on Visual7W, comparing different embeddings for questions and answers. The results are reported for the IoU +QoU-decoys.

9.12 Test accuracy (%) on Visual7W, comparing different random decoy strategies to our methods: (A) Orig + uniformly random decoys from unique correct answers, (B) Orig + weighted random decoys w.r.t. their frequencies, and All (Orig+IoU +QoU).
10.1 Results of *Name That Dataset!* ................................................. 89
10.2 Various Settings for cross-dataset Adaptation. Source domain always provide I, Q and A (T+D) while the target domain provides the same *only* during testing. .......................... 92
10.3 Domain adaptation (DA) results (in %) on *original* VQA [3] and Visual7W [230]. **Direct:** direct transfer without DA. [174]; CORAL. [183]; ADDA. **Within:** apply models trained on the target domain if supervised data is provided. (best DA result in bold) ........................................... 97
10.4 Domain adaptation (DA) results (in %) on *revised* VQA and Visual7W from Chapter 9. (best DA result in bold) ........................................... 97
10.5 DA results (in %) on *original* datasets, with target data sub-sampling by 1/16. FT: fine-tuning. (best DA result in bold) ........................................... 98
10.6 DA results (in %) on *revised* datasets, with target data sub-sampling by 1/16. FT: fine-tuning. (best DA result in bold) ........................................... 98
10.7 DA results (in %) on VQA and Visual7W (both original and revised) using a variant of the SMem model [203]. ........................................... 100
10.8 DA results (in %) on VQA and Visual7W (both original and revised) using a variant of the HieCoAtt model [122]. ........................................... 100
10.9 Transfer results (in %) across different datasets (the decoys are generated according to Chapter 9). The setting for domain adaptation (DA) is on [Q+T+D] using 1/16 of the training examples of the target domain. ........................................... 101
10.10 Domain adaptation (DA) results (in %) with or without the discriminative loss surrogate term ........................................... 102
10.11 OE results (VQA → COCOQA, sub-sampled by 1/16). ..................................... 103
10.12 Transfer results (in %) across datasets (the decoys are generated according to Chapter 9). The setting for domain adaptation (DA) is on [Q+T+D] using all the training examples of the target domain. ........................................... 103
11.1 Summary statistics of Visual QA datasets. ............................................. 111
11.2 The answer coverage of each dataset. ............................................. 112
11.3 Results (%) on Visual QA with different settings: open-ended (Top-K) and multiple-choice (MC) based for different datasets. The omitted ones are due to their missing in the corresponding work. ........................................... 114
11.4 The effect of negative sampling ($M = 3,000$) on fPMC. The number is the accuracy in each question type on VQA2 (val). ........................................... 115
11.5 Detailed analysis of different $\alpha(a, d)$ for weighted likelihood. The reported number is the accuracy on VQA2 (validation). ........................................... 116
11.6 The # of common answers across datasets (training set) ........................................... 116
11.7 Results of cross-dataset transfer using either classification-based models or our models (PMC) for Visual QA. ($f_\theta = SAN$) ........................................... 117
11.8 Transferring is improved on the VQA2 dataset without Yes/No answers (and the corresponding questions) ($f_\theta = SAN$). ........................................... 117
11.9 Analysis of cross dataset performance over Seen/Unseen answers using either CLS or PMC for Visual QA ........................................... 118
11.10 Results for the baseline method that fix answer embedding as GloVe. (We show results with SAN as $f_\theta(i, q)$). ........................................... 118
11.11 Efficiency study among CLS(MLP), uPMC(MLP) and fPMC(MLP). The reported numbers are the average inference time of a mini-batch of 128 (|T| = 1000). . . 120
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>In films that aim to render our future world, intelligent systems that can perform visual recognition, language understanding, and reasoning on top of the two in-formation play a significant role and can always catch our eyes.</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>The performance trend of machines on classifying among 1,000 common object in ILSVRC [157]. The numbers in bars are the top-5 error rates. The leading algorithms shown in the figure are SIFT + FVs [161], AlexNet [101], VGGNet [170], GoogLeNet [175], and ResNet [73].</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>The visual question answering (Visual QA) task [14]: given an image an intelligent system needs to answer questions related to the image.</td>
<td>3</td>
</tr>
<tr>
<td>1.4</td>
<td>An illustration on developing a Visual QA system, which involves collecting training data and performing (supervised) learning. The resulting system then can be applied to an environment similar to the training data; i.e., answering (recognizing) familiar questions (objects).</td>
<td>4</td>
</tr>
<tr>
<td>1.5</td>
<td>An illustration on how a learned system in Fig. 1.4 will fail in the wild to answer unfamiliar question or recognize unseen objects. “?” means the system refuses to answer.</td>
<td>5</td>
</tr>
<tr>
<td>1.6</td>
<td>An illustration of the long-tailed distribution on object categories in nature images (from the the SUN dataset [202]) [228]. The vertical axis corresponds to the number of examples. The blue curve in the inset shows a log-log plot, along with a best-fit line in red. This suggests that the distribution follows a long-tailed power law.</td>
<td>5</td>
</tr>
<tr>
<td>1.7</td>
<td>An illustration on developing a Visual QA system using transfer learning. The learned model can not only perform well in an environment similar to the training data, but transfer its abilities to the wild to further answering unfamiliar questions and recognize unseen objects.</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>We consider zero-shot learning for visual recognition by ignoring the question.</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>An illustration on the possibility of zero-shot learning (ZSL). Given two images and two categories, Okapi and Araripe Manakin, telling which image belongs to which class can be hard if we have not seen them before (i.e., a ZSL task). However, if we are further provided with the class semantic representations (e.g., Okapi has stripes and a black body), then the task becomes much simpler to us. This is because we can visually understand the semantic representation, probably learned from other animals we have seen before.</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Illustration on visual attributes and annotations for (a) animals and (b) human faces.</td>
<td>16</td>
</tr>
</tbody>
</table>
3.2 Illustrations of vector representation of words. (a) Two-dimensional PCA projection of the skip-gram vectors of countries and their capital cities [131]. (b) t-SNE visualization [184] of the skip-gram vectors [131] of visual object names [50].

3.3 Example annotations on bird and flower images [148].

3.4 The class taxonomies of animals [68] derived from WordNet [132, 46].

3.5 An illustration of the DAP model [105, 106]. The figure is from [105, 106] so that the notations are not the same as the ones defined in the thesis. In the figure \( \{a_1, \ldots, a_M\} \) corresponds to \( M \) attributes, \( \{y_1, \ldots, y_K\} \) corresponds to seen classes, and \( \{z_1, \ldots, z_L\} \) corresponds to unseen classes.

3.6 An illustration of the SJE model [6, 5]. The figure is from [5] so that the notations are not the same as the ones defined in the thesis. In the figure, \( \theta(x) \) is equivalent to \( x \) in the thesis; \( \varphi(y_i) \) is equivalent to \( a_{y_i} \) in the thesis.

3.7 An illustration of the IAP model [105, 106]. The figure is from [105, 106] so that the notations are not the same as the ones defined in the thesis. In the figure \( \{a_1, \ldots, a_M\} \) corresponds to \( M \) attributes, \( \{y_1, \ldots, y_K\} \) corresponds to seen classes, and \( \{z_1, \ldots, z_L\} \) corresponds to unseen classes.

4.1 Illustration of our method SynC for zero-shot learning. Object classes live in two spaces. They are characterized in the semantic space with semantic representations (as) such as attributes and word vectors of their names. They are also represented as models for visual recognition (ws) in the model space. In both spaces, those classes form weighted graphs. The main idea behind our approach is that these two spaces should be aligned. In particular, the coordinates in the model space should be the projection of the graph vertices from the semantic space to the model space—preserving class relatedness encoded in the graph. We introduce adaptable phantom classes (b and v) to connect seen and unseen classes—classifiers for the phantom classes are bases for synthesizing classifiers for real classes. In particular, the synthesis takes the form of convex combination.

4.2 Data splitting for different cross-validation (CV) strategies: (a) the seen-unseen class splitting for zero-shot learning, (b) the sample-wise CV, (c) the class-wise CV.

4.3 We vary the number of phantom classes \( R \) as a percentage of the number of seen classes \( S \) and investigate how much that will affect classification accuracy (the vertical axis corresponds to the ratio with respect to the accuracy when \( R = S \)). The base classifiers are learned with Ours\(^{\text{o-vs-o}}\).

4.4 Qualitative results of our method (Ours\(^{\text{struct}}\)) on AwA. (Top) We list the 10 unseen class labels. (Middle) We show the top-5 images classified into each class, according to the decision values. Misclassified images are marked with red boundaries. (Bottom) We show the first misclassified image (according to the decision value) into each class and its ground-truth class label.

5.1 Comparisons of (a) conventional ZSL and (b) generalized ZSL in the testing phase—conventional ZSL assumes the absence of seen classes’ instances and only classifies test instances into one of the unseen classes. The notations follow those in Section 5.2.2.
5.2 We observed that seen classes usually give higher scores than unseen classes, even to an unseen class instance (e.g., a zebra image). We thus introduce a calibration factor $\gamma$, either to reduce the scores of seen classes or to increase those of unseen classes (cf. eq. (5.2)).

5.3 The Seen-Unseen accuracy Curve (SUC) obtained by varying $\gamma$ in the calibrated stacking classification rule eq. (5.2). The AUSUC summarizes the curve by computing the area under it. We use the method SynC$^0$ vs-o on the AwA dataset, and tune hyper-parameters as in Table 5.1. The red cross denotes the accuracies by direct stacking.

5.4 Comparison between several ZSL approaches on the task of GZSL for AwA and CUB.

5.5 Comparison between ConSE and SynC of their performances on the task of GZSL for ImageNet where the unseen classes are within 2 tree-hops from seen classes.

6.1 The comparison of zero-shot, few-shot, and conventional supervised learning (i.e., many shot-learning). For all the paradigms, categories of interest can be separated into two portions: one with many training examples per class; one with zero, few, or again many examples. For ZSL, the first (second) portion is called seen (unseen) classes, and extra class semantic representations $a_c$ are provided. In our SynC algorithm, we learn a mechanism $h$ to synthesize the classifier $w_c$ given the corresponding $a_c$. We can actually learn and apply the same mechanism to the other mechanisms if we have $a_c$: for example, constructing $a_c$ by average visual features.

6.2 We contrast the performances of GZSL to multi-class classifiers trained with labeled data from both seen and unseen classes on the dataset ImageNet-2K. GZSL uses WORD2VECTOR (in red) and the idealized visual features (G-attr) as semantic representations (in black).

7.1 Given the semantic information and visual features of the seen classes, our method learns a kernel-based regressor $\psi(\cdot)$ such that the semantic representation $a_c$ of class $c$ can predict well its class exemplar (center) $v_c$ that characterizes the clustering structure. The learned $\psi(\cdot)$ can be used to predict the visual feature vectors of the unseen classes for nearest-neighbor (NN) classification, or to improve the semantic representations for existing ZSL approaches.

7.2 t-SNE [184] visualization of randomly selected real images (crosses) and predicted visual exemplars (circles) for the unseen classes on (from left to right) AwA, CUB, SUN, and ImageNet. Different colors of symbols denote different unseen classes. Perfect predictions of visual features would result in well-aligned crosses and circles of the same color. Plots for CUB and SUN are based on their first splits. Plots for ImageNet are based on randomly selected 48 unseen classes from 2-hop and word vectors as semantic representations. Best viewed in color.

8.1 The visual question answering (Visual QA) task [14]: given an image an intelligent system needs to answer questions related to the image.
8.2 To learn a Visual QA system, we need to collect multiple question-answer pairs for an image (in black color). However, human language is extremely flexible—there can be exponentially many distinct questions or answers with respect to the vocabulary size and the text length. Moreover, people can have different language styles. As a consequence, there can be many ways of phrasing questions (or answers) of the same semantic meaning (in gray color). It is thus desirable to have a system capable of dealing with those unfamiliar language usage.

8.3 We experiment knowledge transfer across five popular datasets: VQA [14], Visual7W [230], Visual Genome (VG) [100], COCOQA [150], and VQA2 [67]. We train a model on one dataset and investigate how well it can perform on the others.

8.4 We introduced a framework by adapting the unfamiliar language usage (target domain) to what the learned Visual QA model has been trained on (source domain) so that we can re-use the model without re-training.

8.5 Denote $i$ as an image, $q$ as a question, and $c$ as a candidate answer, we aim to learn a scoring function $f(i, q, c)$ so that it gives a high score if $c$ is the target answer of the $(i, q)$ pair. We factorize $f(i, q, c)$ into $h(i, q)$ and $g(c)$, in which we can take advantage of existing joint embedding of vision and language for $h(i, q)$. Moreover, $g(c)$ can effectively captures the answer semantic ignored in many state-of-the-art models. The scoring function is learned to maximize the likelihood of outputting the target answer from a set of stochastically sampled candidates.

9.1 An illustration of how the shortcuts in the Visual7W dataset [230] should be remedied. In the original dataset, the correct answer “A train” is easily selected by a machine as it is far often used as the correct answer than the other decoy (negative) answers. (The numbers in the brackets are probability scores computed using eq. (9.2)). Our two procedures—QoU and IoU (cf. Section 9.4)—create alternative decoys such that both the correct answer and the decoys are highly likely by examining either the image or the question alone. In these cases, machines make mistakes unless they consider all information together. Thus, the alternative decoys suggested our procedures are better designed to gauge how well a learning algorithm can understand all information equally well.

9.2 Example image-question-target triplets from Visual7W, VQA, and VG, together with our IoU-decoys (A, B, C) and QoU-decoys (D, E, F). G is the target. Machine’s selections are denoted by green ticks (correct) or red crosses (wrong).

9.3 Ambiguous examples by our IoU-decoys (A, B, C) and QoU-decoys (D, E, F). G is the target. Ambiguous decoys F are marked.

10.1 An illustration of the dataset bias in visual question answering. Given the same image, Visual QA datasets like VQA [14] (right) and Visual7W [230] (left) provide different styles of questions, correct answers (red), and candidate answer sets, each can contributes to the bias to prevent cross-dataset generalization.

10.2 An illustration of the MLP-based model for multiple-choice Visual QA. Given an IQA triplet, we compute the $M(I, Q, C_k)$ score for each candidate answer $C_k$. The candidate answer that has the highest score is selected as the model’s answer.
10.3 Domain adaptation (DA) results (in %) with limited target data, under Setting [Q+T+D] with $\lambda = 0.1$. A sub-sampling rate $a$ means using $\frac{1}{2a}$ of the target data.

10.4 Qualitative comparison on different type of questions when transferring from VQA− to Visual7W (on the original datasets).

10.5 Results by varying $\lambda$ on the original VQA and Visual7W datasets, for both the [Q+T] and [Q+T+D] settings.

11.1 Conceptual diagram of our approach. We learn two embedding functions to transform image question pair $(i, q)$ and (possible) answer $a$ into a joint embedding space. The distance (by inner products) between the embedded $(i, q)$ and $a$ is then measured and the closest $a$ (in red) would be selected as the output answer.

11.2 Detailed analysis on the size of negative sampling to fPMC(MLP) and fPMC(SAN) at each mini-batch. The reported number is the accuracy on VQA2 (val).

11.3 t-SNE visualization. We randomly select 1000 answers from Visual7W and visualize them in the initial answer embedding and learned answer embeddings. Each answer is marked with different colors according to their question types. (e.g. when, how, who, where, why, what). To make the figure clear for reading, we randomly sub-sampled the text among those 1000 answers to visualize.

11.4 Inference time Vs. Mini-batch index. fPMC(MLP) and CLS(MLP) model are 10x faster than uPMC(MLP) (use PyTorch v0.2.0 + Titan XP + Cuda 8 + Cudnnv5).
Abstract

Developing intelligent systems for vision and language understanding has long been a crucial part that people dream about the future. In the past few years, with the accessibility to large-scale data and the advance of machine learning algorithms, vision and language understanding has had significant progress for constrained environments. However, it remains challenging for unconstrained environments in the wild where the intelligent system needs to tackle unseen objects and unfamiliar language usage that it has not been trained on. Transfer learning, which aims to transfer and adapt the learned knowledge from the training environment to a different but related test environment has thus emerged as a promising framework to remedy the difficulty.

In my thesis, I focus on two challenging paradigms of transfer learning: zero-shot learning and domain adaptation. I will begin with zero-shot learning (ZSL), which aims to expand the learned knowledge from seen objects, of which we have training data, to unseen objects, of which we have no training data. I will present an algorithm SynC that can construct the classifier of any object class given its semantic representation, even without training data, followed by a comprehensive study on how to apply it to different environments. The study further suggests directions to improve the semantic representation, leading to an algorithm EXEM that can widely benefit existing ZSL algorithms.

I will then describe an adaptive visual question answering (Visual QA) framework that builds upon the insight of zero-shot learning and can further adapt its knowledge to new environments given limited information. Along our work we also revisit and revise existing Visual QA datasets so as to ensure that a learned model can faithfully comprehend and reason both the visual and language information, rather than relying on incidental statistics to perform the task.

For both zero-shot learning for object recognition and domain adaptation for visual question answering, we conduct extensive empirical studies on multiple (large-scale) datasets and experimental settings to demonstrate the superior performance and applicability of our proposed algorithms toward developing intelligent systems in the wild.
Part I

Background
Chapter 1

Introduction

Intelligent systems have long played the main role in our dreams about the future. While “intelligence” can be defined in many different ways to include the capacity for logic, understanding, self-awareness, learning, reasoning, planning, and problem solving, to we human beings the most useful intelligent systems in our daily lives will be those that can fluently interact with us and the environment via visual perception and natural language. Specifically, in films that aim to render our future world, most intelligent systems are equipped with the abilities to visually recognize the environment, understand human language, and reason on top of both sources of information (see Fig. 1.1). These systems can perform those abilities not only in environment they have been familiar with (e.g., at home or office, and with familiar people), but also in the wild—to interact with new environment and even update themselves to get familiar with it, just like we humans do.

Figure 1.1: In films that aim to render our future world, intelligent systems that can perform visual recognition, language understanding, and reasoning on top of the two information play a significant role and can always catch our eyes.
Beyond just dreaming, it has taken us a great amount of time and effort toward developing intelligent systems, with several milestones being achieved in the past few years. One striking success is on visual object recognition, in which an intelligent system (or machine) needs to tell the object category pictured in an image. In the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [157], where there are 1,000 common object categories, a machine can achieve a 4% top-5 error rate, better than 5% by humans (see Fig. 1.2). We also have question answering systems like Siri and Amazon Alexa that can interact with humans via natural language.

Built upon these technologies, now we are looking more into systems that can handle multi-modal information, such as one that can perform visual question answering (Visual QA), in which given a visual input (e.g., an image) a machine needs to answer related questions (see Fig. 1.3). Seen essentially as a form of (visual) Turing test that artificial intelligence should strive to achieve, Visual QA has attracted a lot of attention lately. Just in the past three years, promising progresses have been made. For example, on the VQA dataset [14] where human attains accuracy of 88.5%, the state-of-the-art model on the multiple-choice task can already achieved 71.4% [227].

1.1 Machine learning for intelligent systems

Many of those progresses are attributed to the availability of large-scale training data and the advance of machine learning algorithms. For example, to develop a Visual QA system, we need
Figure 1.4: An illustration on developing a Visual QA system, which involves collecting training data and performing (supervised) learning. The resulting system then can be applied to an environment similar to the training data; i.e., answering (recognizing) familiar questions (objects).

to first collect training data (e.g., many image-question-answer triplets). Then we design the system’s model architecture and perform (supervised) learning to determine the parameters of the system. If everything goes well—collecting sufficient and high-quality data, designing a suitable architecture and algorithm, and learning the model till converge—the resulting system should perform well in a test environment similar to where the training data is collected.

### 1.2 Challenges in the wild

A system constructed in such a way, however, may not perform well in the wild to answer unfamiliar questions or recognize unseen objects. For instance, the system learned in Fig. 1.4 will fail if given an image of a zebra. The system has never seen zebra before so the best answer it can guess will be a horse. On the other hand, if the system is given an unfamiliar question like “What is the creature called?”. Then even if the image is about familiar objects like horse, the system might refuse to give any answer. (See Fig. 1.5 for an illustration.) Therefore, in order to develop a system that can work in the wild, we must resolve the above two challenges.

A straightforward solution, following the conventional machine learning pipeline as depicted in Fig. 1.4, is to re-collect training data to cover those unseen and unfamiliar instances and re-learn the systems from scratch. This method, however, is practically tedious and costly. There are exponentially many possible questions and a huge amount of object categories (depending on the

---

1The availability of large-scale data allows learning powerful models like deep neural networks [101, 175, 170, 73] and enables the learned model to generalize well to the test data (sampled from the same distribution of the training data). For instance, the models for ILSVRC are trained using one million labeled images [38].
Figure 1.5: An illustration on how a learned system in Fig. 1.4 will fail in the wild to answer unfamiliar question or recognize unseen objects. “?” means the system refuses to answer.

Moreover, many object categories in natural images follow the so-called long-tail distribution [160, 228]: in contrast to common objects such as household items, they do not occur frequently enough for us to collect and label a large set of representative images. (See Fig. 1.6 for an illustration.) Last but not the least, re-training models by ignoring the existing ones is simply time and computationally consuming. A more efficient and extensible solution is thus desirable.

Figure 1.6: An illustration of the long-tailed distribution on object categories in nature images (from the SUN dataset [202]) [228]. The vertical axis corresponds to the number of examples. The blue curve in the inset shows a log-log plot, along with a best-fit line in red. This suggests that the distribution follows a long-tailed power law.

1.3 Transfer learning for intelligent systems

In my thesis, I am dedicated to developing intelligent systems using the concept of transfer learning [139], which seeks to properly transfer data, knowledge, or models from related (training) environments to the test environment. In our case, this amounts to designing transfer learning algorithms so that the learned model can not only perform well in an environment similar to the
training data, but transfer its abilities to the wild (with the help of external knowledge, or a limited amount of data from the wild) to further answering unfamiliar questions and recognize unseen objects. Fig. 1.7 gives an illustration.

To begin with, we should note that the two challenges are fundamentally different. According to the training data depicted in Fig. 1.7 (on the top), we call zebra an unseen object, and we require the system not only to recognize it (i.e., be able to tell the category) but also to differentiate it from visually-similar objects like horse. On contrast, we call “What is the creature called?” an unfamiliar question because it semantically means the same as “What is the animal?”. In this case, we would like the system to associate the two questions so as to apply its already learned knowledge (i.e., generate the answer “a horse”).

We thus formulate them via two different paradigms of transfer learning and develop algorithms accordingly. On one end, we view recognizing unseen object classes as a zero-shot learning (ZSL)\(^2\) problem [138, 105]—in which no training data of (parts of) the classes of interest in the test environment is available. ZSL thus aims to expand and transfer the classifiers (or more abstractly, learned discriminative knowledge) and label space from seen classes, of which we have access to labeled training data, to unseen ones using external class semantic representations.

On the other end, we view answering unfamiliar questions as a domain adaptation (DA) problem [64, 62]—the statistical distributions of training and test data are not identical but related. Domain adaptation thus aims to bridge (or reduce) the distribution difference so that the learned knowledge from the training data can be applied to the test one.

\(^2\)The term “shot” corresponds to the number of training examples for a certain category.
1.4 Contributions and outline

My thesis provides a comprehensive set of insights and techniques to improve zero-shot learning (ZSL) for applications in the wild—from effectively leveraging the semantic representations in relating classes \textbf{(algorithm design)}, to revisiting and revising the ZSL setting \textbf{(evaluation metric)}, and to unifying ZSL with few-shot learning and applying the insights to improve semantic representation \textbf{(connection to other paradigms)}.

My thesis further provides a series of analysis and techniques to improve knowledge transfer across domains for visual question answering (Visual QA)—from revisiting and revising existing datasets \textbf{(dataset design)}, to mitigating domain mismatch while ensuring consistency among modalities \textbf{(algorithm design)}, and to developing a probabilistic framework on leveraging answer semantics to account for out-of-vocabulary answers \textbf{(ZSL for Visual QA)}.

The remaining of the thesis is organized as follows: \textbf{Part II} on zero-shot learning, \textbf{Part III} on domain generalization for visual question answering, and \textbf{Part IV} on the conclusion.

1.5 Published work

1.5.1 Zero-shot learning

Chapter 4 corresponds to our CVPR 2016 paper [26]:


Chapter 5 and Chapter 6 correspond to our ECCV 2016 paper [28]:


Chapter 7 corresponds to our ICCV 2017 paper [27]:


1.5.2 Domain generalization for visual question answering

Chapter 9 corresponds to our NAACL 2018 paper [30]:


Chapter 10 corresponds to our CVPR 2018 paper [31]:


Chapter 11 corresponds to our CVPR 2018 paper [78]:

1.5.3 Other work

Besides the publications relevant to my thesis, we have also published other research accomplishments in NIPS 2014 [61], ICML 2015 [32], UAI 2015 [29], CVPR 2016 [216], and ECCV 2016 [217].


*: Equal contributions
Part II

Zero-shot Learning
Chapter 2

Introduction to Zero-shot Learning

In this part, we will focus on zero-shot learning (ZSL). Built upon the flow chart in Fig 1.7, we make an assumption: the question is always “What is the animal (or object, scene, etc.)?” We thus can ignore the question, leading to a visual recognition task (see Fig. 2.1). We will reconsider different questions in Part III.

Figure 2.1: We consider zero-shot learning for visual recognition by ignoring the question.

In contrast to conventional supervised learning for visual recognition, where both the training and test data (e.g., images and the corresponding category labels) are assumed to come from the same distribution, zero-shot learning (ZSL) distinguishes between two types of classes: seen classes, of which we have access to labeled training data, and unseen ones, of which no training data are available. ZSL then aims to transfer and adapt the classifiers (or more abstractly, learned discriminative knowledge) and label space from seen classes to unseen ones.

To this end, we need to address two key interwoven issues [138]: (1) how to relate unseen classes to seen ones and (2) how to attain discriminative performance on the unseen classes even though we do not have their labeled data. Existing literature assumes the availability of class semantic representations for both types of classes, e.g., human-annotated attributes of classes [44, 105, 140], word vectors of class names [131, 130, 143], textual descriptions of each class [148], and hierarchical class taxonomies [132, 46]. Such representations provide the cue to relate classes
Figure 2.2: An illustration on the possibility of zero-shot learning (ZSL). Given two images and two categories, Okapi and Araripe Manakin, telling which image belongs to which class can be hard if we have not seen them before (i.e., a ZSL task). However, if we are further provided with the class semantic representations (e.g., Okapi has stripes and a black body), then the task becomes much simpler to us. This is because we can visually understand the semantic representation, probably learned from other animals we have seen before.

and design algorithms to recognize unseen ones. (See Fig. 2.2 for an illustration.) The learning problem of ZSL can thus be formulated as follows.

### 2.1 Definition

#### 2.1.1 Notations

Denote by $S = \{1, 2, \cdots, S\}$ the label space of *seen* classes and $U = \{S + 1, \cdots, S + U\}$ the label space of *unseen* classes. We use $T = S \cup U$ to represent the union of the two sets of classes. We then denote by $P_S(y)$ the distribution on $S$, $P_U(y)$ the distribution on $U$, $P_T(y) = \alpha \times P_S(Y) + (1 - \alpha) \times P_U(Y)$ the distribution on $T$ (with $1 > \alpha > 0$), and $P_{X|Y}(x|y)$ the conditional feature distribution on $x \in \mathbb{R}^D$ given $y \in T$ where $D$ is the dimensionality of features. Finally, we denote by $a_c \in \mathcal{A}$ the semantic representation of class $c \in T$.

\footnote{Some existing approaches assume the availability of similarity $s_{i,j}$ between a pair of classes $i$ and $j$. In this case, we can assume $s_{i,j} = s(a_i, a_j)$, where $s(\cdot, \cdot)$ is a certain similarity measure and $a_i$ and $a_j$ are derived accordingly.}
2.1.2 Problem formulations

In ZSL, we are given the training data \( D_{tr} = \{(x_n \in \mathbb{R}^D, y_n)\}_{n=1}^N \), where \((x_n \in \mathbb{R}^D, y_n)\) is i.i.d. sampled from \( P_S(y)P_{X|Y}(x|y) \). That is, the label space of \( D_{tr} \) is \( S = \{1, 2, \ldots, S\} \). Additionally, we are given for each \( c \in S \) the corresponding \( a_c \). The goal of ZSL is to learn from \( D_{tr} \) and \( \{a_c\}_{c=1}^S \) so that in testing, given \( \{a_c\}_{c=S+1}^{S+U} \) that corresponds to the unseen classes \( c \in U \), we can further recognize instances of the unseen classes.

According to how the test instances are generated, ZSL can be categorized into conventional ZSL and generalized ZSL:

- Conventional ZSL: A test instance \((x, y)\) is sampled from \( P_U(y)P_{X|Y}(x|y) \). That is, test instances come only from unseen classes \( U \) and we only classify them among \( U \), implying the absence of seen classes’ instances in the test environment.

- Generalized ZSL: A test instance \((x, y)\) is sampled from \( P_T(y)P_{X|Y}(x|y) \). That is, test instances can come from both seen and unseen classes. The label space is thus the union of them (i.e., \( T \)).

So far, most of the existing work focuses on the conventional setting.

2.1.3 The framework of algorithms

Most of the ZSL algorithms, although not shown obviously at the first glance, aim to learn a scoring function \( f(a, x): A \times X \mapsto \mathbb{R} \) so that we can assign label \( \hat{y} \) to the instance \( x \) by

\[
\hat{y} = \arg\max_{c \in U} f(a_c, x) \quad \text{(for conventional ZSL)},
\]

\[
\hat{y} = \arg\max_{c \in T} f(a_c, x) \quad \text{(for generalized ZSL)}.
\]

Some recently published work take an alternative way of thinking, aiming to generate instances (images or visual features) of the unseen classes given the semantic representations. Conventional supervised learning algorithms then can be applied to train classifiers. We will provide more details on these algorithms in Chapter 3.

2.2 Challenges

According to the problem formulations and the framework of algorithms presented above, ZSL has several challenges categorized as follows.

Class semantic representations While different forms of class semantic representations have been exploited and compared in existing literature, it remains unclear what ideal semantic representations will be and how to improve existing ones or design better ones accordingly.

\footnote{Some existing approaches assume the availability of \( \{a_c\}_{c=S+1}^{S+U} \) in training, or perform training once \( \{a_c\}_{c=S+1}^{S+U} \) is given. These approaches may need to store the training data or retrain the models when unseen classes change.}
Algorithms  Designing algorithms is the main focus of ZSL research in the literature. The challenges are on how to effectively leverage the given semantic representations to relate classes and how to define and learn the scoring function $f(a, x)$ as presented in Section 2.1.3.

Experimental settings  As mentioned in Section 2.1, most of the existing work focuses on the conventional setting, in which the test environment only contains instances of unseen categories. In real-world applications, categories that have available training data are likely the commonly-seen ones. It is thus unrealistic to assume their absence in the test environment. It is important to investigate how the existing work can be applied to the generalized setting. In our studies (Chapter 5), we show that naively combining the scoring functions of seen and unseen classes as in Section 2.1.3 leads to poor performance for the generalized setting.

The performance gap to supervised learning  While much effort has been committed to ZSL and the result on benchmarked datasets has been significantly improved in the past few years, it remains unclear if the state-of-the-art performance is good enough compared to training classifiers with labeled data of all the classes of interest. In the worst case, if the performance gap between these two paradigms (i.e., zero-shot learning vs. supervised learning) is large, it may imply that we should put more effort on collecting and labeling data instead. Or, the existing literature may miss essential factors in exploiting semantic representations or designing algorithms.

Theoretical foundations  Last but not the least, compared to conventional supervised learning that has solid theoretical foundations on performance guarantee, zero-shot learning so far has no such notions, making itself a rather ad-hoc or empirical topic in machine learning.

[41, 167, 107, 53] have also pointed out other challenges including hubness and domain shift.

2.3 Contributions

We provide a comprehensive set of insights and techniques to improve zero-shot learning—from a principled algorithm to effectively leveraging the semantic representations in relating classes (Chapter 4), to revisiting and revising the ZSL settings and evaluation metrics (Chapter 5), to inverting the gap between ZSL and conventional supervised learning as well as suggesting the ideal form of semantic representations (Chapter 6), and to improving the class semantic representations by incorporating visual domain knowledge (Chapter 7).

2.4 Outline of Part II

The remaining of this part is organized as follows:

Chapter 3 is a survey on zero-shot learning. We present the class semantic representations, algorithms, and settings of the existing work in the literature, and discuss related tasks to zero-shot learning.
Chapter 4 presents our synthesized classifiers (SynC) to construct the classifier of any class given its semantic representation for ZSL.

Chapter 5 presents our studies on generalized ZSL, together with an effective calibration framework to balance recognizing seen and unseen classes as well as a metric called Area Under Seen and Unseen Curve (AUSUC) to characterize such a trade-off.

Chapter 6 presents the relationship and investigate the performance gap among zero-shot learning, few-shot learning, and conventional supervised learning. We show that, by designing the semantic representations in a certain way, the performance gap can be largely reduced.

Chapter 7 builds upon Chapter 6 and introduces a novel approach to improve semantic representations by learning a mapping from the original representations to the average visual features.
Chapter 3

Literature Survey on Zero-Shot Learning

Zero-shot learning (ZSL) has attracted significant attention in recent years from computer vision [106, 197, 105, 128, 129, 5, 6, 189], machine learning [171, 156, 83, 136, 75, 91, 172, 138, 207, 112], and artificial intelligence [178, 146, 185]. The major focus is on the classification problem, the one formulated in Chapter 2, while some others work on reinforcement learning [136, 75], imitation learning [141], generative models [91, 149], and visual question answering and captioning [178, 146, 185]. In this chapter, we provide a survey on zero-shot learning for classification\(^1\)—including semantic representations, algorithm design, and relations to other learning paradigms.

3.1 Semantic representations

Semantic representations are the essential information in performing zero-shot learning—without them we have no guidance on how to apply, transfer, or adapt the knowledge learned from the training labeled data to the test environment whose label space has a significant non-overlapping with the training data. Since the goal of zero-shot learning for classification is to classify instances of the classes that have no training data, a situation likely results from rare observations or high cost to collect and annotate data, the semantic representations should be extracted from a different resources or modalities from the data. For example, in visual object recognition, the labeled data are images and their corresponding class labels. Therefore, the semantic representations are usually derived from textual descriptions of the classes. In the following we review the semantic representations that have been developed and exploited for visual recognition.

3.1.1 Visual attributes

Visual attributes are properties (e.g., shapes, materials, colors, textures, parts) commonly observable from the appearances of objects [44, 105, 140, 47, 187, 52] (or scenes [142], human faces [103, 102], etc.) that have human-designated names (e.g., “cylindrical”, “furry”, “red”,

\(^1\)We specifically focus on image-based visual recognition.
“stripped”, “four-legged”). See Fig. 3.1 for an illustration. A good dictionary of visual attributes should contain vocabularies that (1) collectively can concisely and discriminatively describe an object\(^2\) and (2) individually are shared among several objects. These properties make visual attributes a compelling way to visually and semantically represent an object (or object class) and measure the similarity (or difference) among objects. There have been extensive work in the literature of computer vision on how to design a good dictionary and detect attributes from object appearances, and how attributes can benefit visual recognition or other applications [44, 140, 47, 102, 118, 206, 19, 212, 145].

For zero-shot learning, we directly take the pre-defined dictionary and the ground-truth attribute annotations on the class level (mostly done by humans, especially domain experts) as the class semantic representations [105, 187, 142, 223]. We note that it is nearly unavoidable of such human efforts—since we have no labeled images for the unseen classes, we can only rely on the semantic understanding or visual experience by humans to annotate attributes for those classes. This fact makes visual attributes a less practical and attractive way for zero-shot learning when accounting for a massive number of unseen classes. Nevertheless, visual attributes so far have been the most popular semantic representations in existing work.

### 3.1.2 Vector representations (word vectors) of class names

How to represent a word (or phrases) has long been a core task in natural language processing—good representations should faithfully describe the semantic similarity and difference among words. The vector representations (also known as word vectors) learned from the word co-occurrence statistics from large-scale ontologies (such as Wikipedia or news corpora) have been shown as a concise and powerful way to represent words [131, 130, 143]. See Fig. 3.2 for an

---

\(^2\)An object can be described by a vector with the dictionary size as the dimensionality. Each entry means the existence (\([0, 1]\)) or the relative strength or observation probability (\(\mathbb{R}\) or \([0, 1]\)) of the corresponding attribute.
(a) On countries and their capital cities [131].

(b) On visual object names [50].

Figure 3.2: Illustrations of vector representation of words. (a) Two-dimensional PCA projection of the skip-gram vectors of countries and their capital cities [131]. (b) t-SNE visualization [184] of the skip-gram vectors [131] of visual object names [50].

illustration. For zero-shot learning on visual recognition, as long as the class names do show up in the ontologies, we can automatically learn and extract the corresponding word vectors to be the class semantic representations\(^3\). Compared to visual attributes, word vectors of class names require much less human efforts and are more suitable for large-scale tasks. However, since they are learned from the word co-occurrence (or other objectives defined solely on the ontologies) without explicitly taking visual information into account, they may not describe the visual similarity among objects as good as visual attributes.

Word vectors have been used as semantic representations in much recent work [135, 50, 172, 55, 54, 9, 37, 191]. See [6] for a comparison on word vectors learned using different objectives.

3.1.3 Textural descriptions of classes

In stead of treating class names as words in ontologies, we can derive class semantic representations from the textual descriptions of the classes; e.g., from the Wikipedia page of classes. This idea has been used in [111, 42, 144, 148, 8]. In [148], the authors specifically collect for each image of birds or flowers a short textual description (see Fig. 3.3). To represent the textual descriptions, existing work uses the bag of word representation (e.g., term frequency-inverse document frequency feature vectors) [42, 111, 144, 229] or encodes the text sequences by recurrent neural networks (RNN) or convolutional neural networks (CNN) [148]. One exception is [8], which discovers visual terms from documents and represents each class in a similar way to visual attributes.

\(^3\)For class names that contain multiple words (e.g., “baseball player”), we can treat the whole names as new words (e.g., “baseball_player”) and re-learn the word vectors.
3.1.4 Hierarchical class taxonomies

Hierarchical class taxonomies, such as WordNet [132, 46] or domain-specific taxonomies for animals and plants, provide another source of information to relate classes. See Fig. 3.4 for an illustration. In [123], Lu computes the shortest paths between class names on the WordNet hierarchy, transforms the path lengths into similarities, and perform multidimensional scaling (MDS) to obtain the class semantic representations. [5] constructs a binary vector of the size of the total number of nodes in the hierarchy to represent each leaf class \( j \) — the \( i \)-th element is 1 if the corresponding node is the leaf class \( j \) (i.e., \( i = j \)) or its ancestor; otherwise 0. [6] constructs a real-valued vector of the same size as in [5]; the elements encode the similarities (computed from the hierarchy) between a leaf class to all the nodes. In [7], Al-Halah and Stiefelhagen proposed a hierarchical attribute transfer method that combines visual attributes and class taxonomies for zero-shot learning. [155, 154] also consider extracting semantic representations or relationship among classes from the hierarchy. We note that the class taxonomies are usually constructed by human experts. Therefore, they might suffer from the same difficulty as visual attributes.

3.1.5 Other forms of semantic representations

There are also other forms of semantic representations developed in existing literature. [155, 154, 127] use the search hit counts or text snippets from World Wide Web. [43, 4] utilize the part information (e.g., part descriptions or localizations) to obtain high-quality semantic representations for fine-grained classes like bird species, while [92, 120] investigates the use of gazes and similes. Knowledge bases (or graphs) have also been used recently to model relationship among classes [109, 191]. Finally, [84, 166] specifically work on how to combine multiple semantic representations for enhanced performance.
3.2 Algorithms for conventional ZSL

With the class semantic representations as well as the labeled training data of seen classes, zero-shot learning algorithms are then designed to leverage such information so as to obtain discriminative performance on unseen classes. According to how the semantic representations are being used, existing algorithms can roughly be categorized into (1) embedding-based methods and (2) similarity-based methods. In this section we survey several representative algorithms of each category. We will also discuss some methods that may not be identified as either category.

We note that some methods require the similarity (or relatedness) between pairs of classes as the semantic cue to relate classes (i.e., \( s_{ij} \) for classes \( i \) and \( j \)). By assuming that there exists a certain similarity measure \( s(\cdot, \cdot) \) and class semantic representations \( \{a_c\}_{c=1}^{S+U} \) such that \( s_{ij} = s(a_i, a_j) \forall i, j \in T \), we can still view those methods as taking \( \{a_c\}_{c=1}^{S+U} \) as the semantic cues to perform zero-shot learning.

Without loss of generality, in the following we treat \( a_c \) as either a binary or real-valued vector.

3.2.1 Embedding-based methods

In the embedding-based approaches, one first maps the input image representation \( x \) to the semantic representation space \( \mathcal{A} \), and then infers the class label in this space by various similarity (or relatedness) measures to the unseen classes’ semantic representations [5, 6, 50, 51, 53, 97, 106, 116, 135, 172, 192], essentially a two-stage procedure. In other words, denote by \( m : \mathcal{X} \rightarrow \mathcal{A} \) such a mapping, and denote by \( s(\cdot, \cdot) \) the similarity measure on \( \mathcal{A} \), embedding-based approaches predicts the label for the input feature vector \( x \) by

\[
\hat{y} = \arg \max_{c \in U} s(m(x), a_c). \tag{3.1}
\]

Note that \( s(\cdot, \cdot) \) can be asymmetric. Approaches of this category are different by how to define \( s(\cdot, \cdot) \) and how to learn the mapping \( m(\cdot) \).

The concept behind this category of approaches is that the semantic representations, together with the measure \( s(\cdot, \cdot) \), can capture the similarities among both seen and unseen classes. Moreover, each element of the representations has certain meaning that shares across multiple categories. For example, if the representations are attributes (i.e., each entry of \( a_c \) corresponds to the existence of a certain attribute or not), there should be multiple classes sharing one attribute. Therefore, even we do not have labeled images of unseen classes in training, we can still recognize those classes by detecting if the test image has certain attributes—the attributes detectors can be learned from images of seen classes.

**Direct attribute prediction (DAP)**  DAP [105, 106] assumes that the \( K \) dimensional class semantic representations are binary vectors, and builds for each element \( a[k] \) (i.e., \( k \)-th attribute)
a probabilistic detector \( p(a[k]|x) \). We use \( a \) here as a random vector, and \( a_c \) the vector corresponding to class \( c \). DAP then defines the posterior \( p(c|x) \) on class label \( c \) given \( x \) as

\[
p(c|x) = \sum_{a \in \{0,1\}^K} p(c|a)p(a|x) = \frac{p(c) \prod_{k=1}^K p(a_c[k]|x)}{p(a_c)}.
\]  

where normally the prior \( p(c) \) is set as uniform over unseen classes, and \( p(a_c) = \prod_{k=1}^K p(a_c[k]) \) can be estimated empirically from the training data. The \( k \)-th detector \( p(a[k]|x) \) can be learned by training a logistic regression to classify images of the seen classes that have the \( k \)-th attribute from images of those that do not have the \( k \)-th attribute. The maximum a posteriori (MAP) rule is then applied to assign the class label to \( x \):

\[
\hat{y} = \arg \max_{c \in U} p(c|x) = \arg \max_{c \in U} \prod_{k=1}^K \frac{p(a_c[k]|x)}{p(a_c[k])}.
\]  

See Fig. 3.5 for an illustration.

In terms of Eq. (3.1), in DAP we have

\[
m(x) = [p(a[1] = 1|x), \cdots , p(a[K] = 1|x)]^T, \quad s(m(x), a_c) = \prod_{k=1}^K \frac{m(x)[k]a_c[k](1 - m(x)[k])(1 - a_c[k])}{p(a_c[k])}.
\]  

\[\text{The assumption } p(a_{c'}|c) = 1 \text{ if } c' = c, \text{ otherwise } = 0, \text{ is imposed.}\]
DAP is among the first few algorithms for zero-shot learning and has a probabilistic interpretation. However, it cannot be directly applied to real-valued semantic representations. Moreover, DAP learns $p(a[k]|x)$ to minimize the error of predicting attributes, which may not guarantee good performance on predicting class labels. See [192, 83, 7] for extensions on DAP to incorporate correlations among attributes, to account for the unreliability of attribute predictions, and to define better decision rules based on hierarchical attribute transfer.

**Structured Joint embedding (SJE)** SJE [6] learns a linear mapping $W \in \mathbb{R}^{D \times K}$ from the input image features $x$ to the semantic representation space (i.e., $m(x) = W^\top x$). It then measures the similarity between the image and the class $c$ by the inner product between the mapped features $W^\top x$ and $a_c$ (i.e., $x^\top W a_c$). The class decision rule is thus

$$\hat{y} = \arg \max_{c \in U} x^\top W a_c. \tag{3.8}$$

SJE applies the structured SVM formulation to learn $W$ from seen classes’ data

$$\min_W \sum_n \max(0, \max_{c \in S} \Delta(c, y_n) + x_n^\top W a_c - x_n^\top W a_{y_n}) + \lambda \Omega(W), \tag{3.9}$$

where $\Omega(\cdot)$ is a certain regularizer on $W$. In [6], $\Delta(c, y_n) = 1[c \neq y_n]$ and $\lambda = 0$, and stochastic gradient descent with early stopping is applied to optimize $W$.

SJE, compared to DAP, can be applied to real-valued semantic representations, and $W$ is learned to optimize the class prediction loss on the seen classes’ data. There are several other methods that share such advantages as SJE, including but not limited to [5, 50, 172, 156, 199, 54, 98, 115, 204, 133, 114, 215]. The main difference of these methods is on how to define the mapping $m(\cdot)$, the similarity, the loss function, and the regularization term. For example, [5] adapts the WSABIE loss [193] that is originally proposed for ranking. [172] minimizes the $\ell_2$ distance between $m(x_n)$ and $a_{y_n}$, where $m(\cdot)$ is modeled by a multi-layer perceptron (MLP). [24] considers learning the metric for minimizing Mahalanobis distance. [156] adopts a regression loss and a special regularizer that jointly lead to a close-form solution for $W$. [98, 33] exploit the reconstruction loss (inspired by auto-encoders) as a regularizer. [199, 43, 4] learn multiple $W$s at the same time to boost the performance.
Other methods  Eq. (3.1) can be extended to

\[ \hat{y} = \arg \max_{c \in \mathcal{U}} s(m(x), g(a_c)), \]  

(3.10)

where \( m : \mathcal{X} \rightarrow \mathcal{B} \) and \( g : \mathcal{A} \rightarrow \mathcal{B} \) are learnable mappings that map the image features \( x \) and the semantic representation \( a \) into a joint embedding space \( \mathcal{B} \), in which the similarity can be faithfully measured or noise in \( x, a \) can be suppressed. This extension has been considered in [207, 111, 221, 144, 148, 40, 86]. The mappings are mostly learned to minimize the classification error on the training data. The loss function or mapping forms need carefully design to prevent over-fitting to seen classes and poor generalization to unseen classes. For example, [144] learns linear mappings and impose a sparse constraint so as to robustly extract informative words from the documents to describe a class.

3.2.2 Similarity-based methods

In the similarity-based approaches, in contrast, one builds the classifiers for unseen classes by relating them to seen ones via class-wise similarities [55, 60, 127, 154, 155, 135, 105, 106]. Denote by \( h_c : \mathcal{X} \rightarrow \mathbb{R} \) or \([0, 1]\) the scoring function (classifier) for each seen class \( c \in \mathcal{S} \). Similarity-based approaches construct the classifier of an unseen class \( u \in \mathcal{U} \) by defining or learning a mapping \( \phi \) so that \( h_u = \phi(a_u, \{a_c\}_{c=1}^{\mathcal{S}}, \{h_c(\cdot)\}_{c=1}^{\mathcal{S}}) \). The concept behind this category of approaches is that the class semantic representations can convey the relatedness among classifiers, especially between the seen classes’ classifiers and the unseen ones.

One common drawback of the similarity-based approaches is that the learned seen class classifiers are not optimized to transfer the discriminative knowledge among classes.

Indirect attribute prediction (IAP)  IAP [105, 106] is very similar to DAP: they both estimate \( p(a[k]|x) \) for the \( K \) attributes and then apply the MAP rule to assign class labels to \( x \). The main difference is on how to estimate \( p(a[k]|x) \). Instead of training \( K \) binary classifiers directly as in DAP, IAP first trains the probabilistic classifier \( p(c|x) \) for \( c \in \mathcal{S} \) from the seen classes’ data. This can be done by learning a softmax classifier. DAP then obtains \( p(a[k]|x) \) by the following formula,

\[ p(a[k]|x) = \sum_{c \in \mathcal{S}} p(a[k]|c)p(c|x) = \sum_{c \in \mathcal{S}} 1[a[k] = a_c[k]]p(c|x). \]  

(3.11)

See Fig. 3.7 for an illustration.

IAP enjoys the same probabilistic interpretation as DAP. However, it also suffers from the same disadvantage—the learned \( p(c|x) \) can not guarantee good classification performance at the final MAP step.

Convex combination of semantic embedding (ConSE)  ConSE [135], similar to IAP, makes use of pre-trained classifiers \( p(c|x) \) for seen classes and their probabilistic outputs. ConSE uses
Figure 3.7: An illustration of the IAP model [105, 106]. The figure is from [105, 106] so that the notations are not the same as the ones defined in the thesis. In the figure $\{a_1, \cdots, a_M\}$ corresponds to $M$ attributes, $\{y_1, \cdots, y_K\}$ corresponds to seen classes, and $\{z_1, \cdots, z_L\}$ corresponds to unseen classes.

\[ p(c|x) \] to infer the semantic embeddings of $x$, and then classifies it into the unseen class using the same rule as in Eq. (3.1),

\[
\hat{y} = \arg\max_{c \in U} s(m(x), a_c) = \arg\max_{c \in U} \sum_{c \in S} a_c p(c|x), a_c). 
\] (3.12)

In [135], the cosine similarity is used for $s(\cdot, \cdot)$.

**Co-occurrence statistics (COSTA)** COSTA [127] constructs classifier $h_u$ for an unseen class $u$ by

\[
h_u(x) = \sum_{c \in S} s(a_u, a_c) h_c(x). \] (3.13)

COSTA considers several co-occurrence statistics to estimate $s_{ij} = s(a_i, a_j)$. This way of classifier construction is also adopted in [154, 155],

\[
h_u(x) = \frac{1}{T} \sum_{c \in S_T^{(u)}} h_c(x), \] (3.14)

where $S_T^{(u)}$ is a subset of $S$ containing those that have the $T$ highest similarities to $u$.

**Other methods** Elhoseiny et al. [42] proposed to learn a mapping from $A$ to $H$ (the hypothesis space of classifiers). This is, in theory, the most straightforward way to perform zero-shot learning. In practice, however, we only have $S$ pairs of data $\{(a_c, h_c)\}_{c=1}^{S}$ to learn such a mapping. Therefore, how to regularize in the learning process becomes extremely crucial.
Fu et al. [55] proposed a similar idea to IAP and ConSE, first getting \( p(c|x) \) to represent \( x \). They then applied a novel absorbing Markov chain process (AMP) on the graph relating seen and unseen classes (the edge weights are based on \( s(\cdot, \cdot) \)) to predict the class label for \( x \).

3.2.3 Other approaches

**Predicting visual instances of unseen classes** The main issue that leads to the need of zero-shot learning is the lack of (labeled) training data for unseen classes. One idea, beyond embedding-based and similarity-based methods, is to generate synthetic images (or the corresponding visual features) for each unseen class according to its semantic representation—if we can have labeled data for unseen classes, we can then apply conventional supervised learning techniques to learn classifiers for unseen classes.

This idea has gradually become popular for ZSL. According to the ability to generate multiple instances per class or not, methods can be separated into predicting visual exemplars [218, 189, 13, 119, 224] or predicting visual instances [149, 121, 25, 200, 190, 188, 229, 15, 225, 69]. Guo et al. [70] proposed to weightedly transfer labeled examples from seen classes to unseen ones, so that unseen classes can have pseudo labeled data.

**Predicting attributes from word vectors** [9, 37] consider a setting called unsupervised zero-shot learning, where attribute annotations are provided only for seen classes. They propose to leverage word vectors (for both types of classes) to explicitly or implicitly predict the attributes for unseen classes before performing zero-shot learning using attributes. The underlying belief is that attributes provide better semantic information than word vectors for object recognition.

3.3 Algorithms for generalized ZSL

There has been very little work on generalized zero-shot learning. [50, 135, 128, 176] allow the label space of their classifiers to include seen classes but they only test on the data from the unseen classes. [172] proposes a two-stage approach that first determines whether a test data point is from a seen or unseen class, and then apply the corresponding classifiers. However, their experiments are limited to only 2 or 6 unseen classes. In the domain of action recognition, [57] investigates the generalized setting with only up to 3 seen classes. [42] and [111] focus on training a zero-shot binary classifier for each unseen class (against seen ones)—it is not clear how to distinguish multiple unseen classes from the seen ones. Finally, open set recognition [162, 163, 81] considers testing on both types of classes, but treating the unseen ones as a single outlier class. In the following, we describe the methods in [172], which is the most relevant one to ours.

Socher et al. [172] propose a two-stage zero-shot learning approach that first predicts whether an image is of seen or unseen classes and then accordingly applies the corresponding classifiers. The first stage is based on the idea of novelty detection and assigns a high novelty score if it is unlikely for the data point to come from seen classes. They experiment with two novelty detection strategies: Gaussian and LoOP models [99]. The main idea is to assign a novelty score \( N(x) \) to each sample \( x \). With this novelty score, the final prediction rule becomes

\[
\hat{y} = \begin{cases} 
\arg \max_{c \in S} f(a_c, x), & \text{if } N(x) \leq -\gamma, \\
\arg \max_{c \in U} f(a_c, x), & \text{if } N(x) > -\gamma.
\end{cases}
\] (3.15)
where $-\gamma$ is the novelty threshold. The scores above this threshold indicate belonging to unseen classes. To estimate $N(x)$, for the Gaussian model, data points in seen classes are first modeled with a Gaussian mixture model. The novelty score of a data point is then its negative log probability value under this mixture model. Alternatively, the novelty score can be estimated using the Local Outlier Probabilities (LoOP) model [99]. The idea there is to compute the distances of $x$ to its nearest seen classes. Such distances are then converted to an outlier probability, interpreted as the likelihood of $x$ being from unseen classes.

Recently Lee et al. [110] propose a hierarchical novelty detector to improve the performance.

### 3.4 Related tasks to zero-shot learning

In this section we present and discuss related tasks to the problem formulations of zero-shot learning described in Chapter 2.

#### 3.4.1 Transductive and semi-supervised zero-shot learning

[153, 51, 53, 97, 169, 222, 210, 134, 205, 173, 71] focus on the transductive setting, where they have access to unlabeled test data from unseen classes during the training stage. [113, 115] works on the semi-supervised setting, where a portion of unlabeled data (not used for testing) from unseen classes are available at training. For both settings, the unlabeled data from unseen classes can be used to refined the embedding function $m(\cdot)$ (c.f. Section 3.2.1) or the semantic representations. One key difference between the two settings is that in the transductive setting, the test data are given as a whole (i.e., we can exploit certain properties like smoothness among the test data to perform joint prediction).

#### 3.4.2 Zero-shot learning as the prior for active learning

Gavves et al. [60] consider the active learning problem—how to pick unlabeled data for acquiring annotations so as to efficiently obtain supervised learning signal. For classes that previously have no labeled data, they propose to use the classifiers constructed by zero-shot learning as the prior knowledge and develop several informative measures to rank instances for querying annotations.

#### 3.4.3 Few-shot learning

Few-shot learning [151, 176, 45, 104, 20, 171] considers the case where we only have for each class of interest few labeled training examples (e.g., one-shot learning means each class has only one labeled training example). Similar to zero-shot learning, few-shot learning usually assumes the availability of either (1) sufficient labeled training data for a set of common classes or (2) class semantic representations. This information enables inferring the data variation of those few-shot classes for constructing robust classifiers. In Chapter 5, we will discuss how zero-shot learning algorithms can be applied to one- or few-shot learning.
Chapter 4

Synthesize Classifier (SynC) for Zero-Shot Learning

In this chapter, we introduce Synthesized Classifiers (SynC), a state-of-the-art zero-shot learning algorithm for visual recognition. SynC effectively leverages the class semantic representations to relate classes so that the discriminative knowledge (i.e., the classifiers or models) learned from the seen classes can be transferred to the unseen ones. In contrast to the embedding- and similarity-based approaches, we aim to learn to predict the (linear) classifier of a class given its semantic representation. We describe the main idea first, followed by the details of the algorithm.

4.1 Main idea

Given class semantic representations, zero-shot learning aims to accurately recognize instances of the unseen classes by associating them to the seen ones. We tackle this challenge with ideas from manifold learning [16, 76], converging to a two-pronged approach. We view object classes in a semantic space as a weighted graph where the nodes correspond to object class names and the weights of the edges represent how they are related (according to the semantic representations). On the other end, we view models for recognizing visual images of those classes as if they live in a space of models. In particular, the parameters for each object model are nothing but coordinates in this model space whose geometric configuration also reflects the relatedness among objects. Fig. 4.1 illustrates this idea conceptually.

But how do we align the semantic space and the model space? The semantic space coordinates of objects are designated or derived based on class semantic representations that do not directly examine visual appearances at the lowest level, while the model space concerns itself largely for recognizing low-level visual features. To align them, we view the coordinates in the model space as the projection of the vertices on the graph from the semantic space—there is a wealth of literature on manifold learning for computing (low-dimensional) Euclidean space embeddings from the weighted graph, for example, the well-known algorithm of Laplacian eigenmaps [16].

To adapt the embeddings (or the coordinates in the model space) to data, we introduce a set of phantom object classes—the coordinates of these classes in both the semantic space and the model space are adjustable and optimized such that the resulting model for the real object classes achieve the best performance in discriminative tasks. However, as their names imply, those phantom classes do not correspond to and are not optimized to recognize any real classes directly. For mathematical convenience, we parameterize the weighted graph in the semantic space with the phantom classes in such a way that the model for any real class is a convex combination of the
coordinates of those phantom classes. In other words, the “models” for the phantom classes can also be interpreted as bases (classifiers) in a dictionary from which a large number of classifiers for real classes can be synthesized via convex combinations. In particular, when we need to construct a classifier for an unseen class, we will compute the convex combination coefficients from this class’s semantic space coordinates and use them to form the corresponding classifier.

4.2 Approach

4.2.1 Notations

We focus on linear classifiers in the visual feature space $\mathbb{R}^D$ that assign a label $\hat{y}$ to a data point $x$ by

$$\hat{y} = \arg \max_c w_c^T x,$$  \hspace{1cm} (4.1)

where $w_c \in \mathbb{R}^D$, although our approach can be readily extended to nonlinear settings by the kernel trick [165].

4.2.2 Manifold learning with phantom classes

We introduce a set of phantom classes associated with semantic representations $b_r, r = 1, 2, \ldots, R$. We stress that they are phantom as they themselves do not correspond to any real objects—they are introduced to increase the modeling flexibility, as shown below.
The real and phantom classes form a weighted bipartite graph, with the weights defined as

$$s_{cr} = \frac{\exp\{-d(a_c, b_r)\}}{\sum_{r=1}^{R} \exp\{-d(a_c, b_r)\}}$$

(4.2)

to correlate a real class \(c\) and a phantom class \(r\), where

$$d(a_c, b_r) = (a_c - b_r)^\top \Sigma^{-1} (a_c - b_r),$$

(4.3)

and \(\Sigma^{-1}\) is a parameter that can be learned from data, modeling the correlation among semantic representations. For simplicity, we set \(\Sigma = \sigma^2 I\) and tune the scalar free hyper-parameter \(\sigma\) by cross-validation.

The specific form of defining the weights is motivated by several manifold learning methods such as SNE [76]. In particular, \(s_{cr}\) can be interpreted as the conditional probability of observing class \(r\) in the neighborhood of class \(c\). However, other forms can be explored.

In the model space, each real class is associated with a classifier \(w_c\) and the phantom class \(r\) is associated with a virtual classifier \(v_r\). We align the semantic and the model spaces by viewing \(w_c\) (or \(v_r\)) as the embedding of the weighted graph. In particular, we appeal to the idea behind Laplacian eigenmaps [16], which seeks the embedding that maintains the graph structure as much as possible; equally, the distortion error

$$\min_{w_c, v_r} \|w_c - \sum_{r=1}^{R} s_{cr} v_r\|^2_2$$

is minimized. This objective has an analytical solution

$$w_c = \sum_{r=1}^{R} s_{cr} v_r, \quad \forall c \in T = \{1, 2, \cdots, S + U\}$$

(4.4)

In other words, the solution gives rise to the idea of synthesizing classifiers from those virtual classifiers \(v_r\). For conceptual clarity, from now on we refer to \(v_r\) as base classifiers in a dictionary from which new classifiers can be synthesized. We identify several advantages. First, we could construct an infinite number of classifiers as long as we know how to compute \(s_{cr}\). Second, by making \(R \ll S\), the formulation can significantly reduce the learning cost as we only need to learn \(R\) base classifiers.

### 4.2.3 Learning phantom classes

**Learning base classifiers** We learn the base classifiers \(\{v_r\}_{r=1}^{R}\) from the training data (of the seen classes only). We experiment with two settings. To learn one-versus-other classifiers, we optimize,
where $\ell(x, y; w) = \max(0, 1 - yw^\top x)^2$ is the squared hinge loss. The indicator $I_{y_n, c} \in \{-1, 1\}$ denotes whether or not $y_n = c$. It is easy to show that is a convex formulation in $v_r$ and can be efficiently solved, at the same computational cost as training $R$ one-versus-other linear classifiers. Alternatively, we apply the Crammer-Singer multi-class SVM loss [35], given by

$$
\ell_{cs}(x_n, y_n; \{w_c\}_{c=1}^S) = \max(0, \max_{c \in S - \{y_n\}} \Delta(c, y_n) + w_c^\top x_n - w_{y_n}^\top x_n),
$$

We have the standard Crammer-Singer loss when the structured loss $\Delta(c, y_n) = 1$ if $c \neq y_n$, which, however, ignores the semantic relatedness between classes. We additionally use the $\ell_2$ distance for the structured loss $\Delta(c, y_n) = \|a_c - a_{y_n}\|_2^2$ to exploit the class relatedness in our experiments. These two learning settings have separate strengths and weaknesses in empirical studies.

**Learning semantic representations for phantom classes** The weighted graph eq. (4.2) is also parameterized by adaptable embeddings of the phantom classes $b_r$. For this work, however, for simplicity, we assume that each of them is a sparse linear combination of the seen classes’ semantic representations:

$$
b_r = \sum_{c=1}^S \beta_{rc} a_c, \forall r \in \{1, \cdots, R\},
$$

Thus, to optimize those embeddings, we solve the following optimization problem

$$
\begin{align*}
\min_{\{v_r\}_{r=1}^R, \{\beta_{rc}\}_{r,c=1}^R} & \sum_{r=1}^R \sum_{c=1}^S \sum_{n=1}^N \ell(x_n, I_{y_n, c}; w_c) + \frac{\lambda}{2} \sum_{r=1}^R \sum_{c=1}^S \|w_c\|_2^2, \\
\text{s.t.} & \; w_c = \sum_{r=1}^R s_{cr} v_r, \; \forall c \in T = \{1, \cdots, S\},
\end{align*}
$$

where $\lambda$ is a predefined scalar equal to the norm of real semantic representations (i.e., 1 in our experiments since we perform $\ell_2$ normalization). Note that in addition to learning $\{v_r\}_{r=1}^R$, we learn combination weights $\{\beta_{rc}\}_{r,c=1}^R$. Clearly, the constraint together with the third term in the objective encourages the sparse linear combination of the seen classes’ semantic representations. The last term in the objective demands that the norm of $b_r$ is not too far from the norm of $a_c$. 

29
We perform alternating optimization for minimizing the objective function with respect to \( \{v_r\}_{r=1}^R \) and \( \{\beta_{rc}\}_{r,c=1}^{R,S} \). While this process is nonconvex, there are useful heuristics to initialize the optimization routine. For example, if \( R = S \), then the simplest setting is to let \( b_r = a_r \) for \( r = 1, \ldots, R \). If \( R \leq S \), we can let them be (randomly) selected from the seen classes’ semantic representations \( \{b_1, b_2, \cdots, b_R\} \subseteq \{a_1, a_2, \cdots, a_S\} \), or first perform clustering on \( \{a_1, a_2, \cdots, a_S\} \) and then let each \( b_r \) be a combination of the seen classes’ semantic representations in cluster \( r \). If \( R > S \), we could use a combination of the above two strategies.

There are four hyper-parameters \( \lambda, \sigma, \eta, \) and \( \gamma \) to be tuned (Section 4.2.3). To reduce the search space during cross-validation, we first fix \( b_r = a_r \) for \( r = 1, \ldots, R \) and tune \( \lambda, \sigma \). Then we fix \( \lambda \) and \( \sigma \) and tune \( \eta \) and \( \gamma \). We describe in more detail how to cross-validate hyper-parameters in Section 4.4.

### Classification with synthesized classifiers

Given a data sample \( x \) from \( \mathcal{U} \) unseen classes and their corresponding semantic representations (or coordinates in other semantic spaces), we classify it in the label space \( \mathcal{U} \) by

\[
\hat{y} = \arg \max_{c \in \mathcal{U}} \mathbf{w}_c^\top \mathbf{x}
\]

with the classifiers being synthesized according to eq. (4.4). This is in sharp contrast to many existing two-stage methods (see Chapter 3). There, the image \( x \) needs to be first mapped with embedding-based functions (e.g., classifiers) and then the outputs of those functions are combined to predict a label in \( \mathcal{U} \).

### 4.3 Comparison to existing methods

We contrast our approach to some existing methods. [127] combines pre-trained classifiers of seen classes to construct new classifiers. To estimate the semantic representation (e.g., word vector) of a test image, [135] uses the decision values of pre-trained classifiers of seen objects to weighted average the corresponding semantic representations. Neither of them has the notion of base classifiers, which we introduce for constructing the classifiers and nothing else. We thus expect them to be more effective in transferring knowledge between seen and unseen classes than overloading the pretrained and fixed classifiers of the seen classes for dual duties. We note that [5] can be considered as a special case of our method. In [5], each attribute corresponds to a base and each “real” classifier corresponding to an actual object is represented as a linear combination of those bases, where the weights are the real objects’ “representations” in the form of attributes. This modeling is limiting as the number of bases is fundamentally limited by the number of attributes. Moreover, the model is strictly a subset of our model.\(^1\) [220, 221] propose similar ideas of aligning the visual and semantic spaces but take different approaches. Very recently, [191] extends our idea on predicting classifiers using the graph convolutional neural networks.

\(^1\)For interested readers, if we set the number of attributes as the number of phantom classes (each \( b_r \) is the one-hot representation of an attribute), and use Gaussian kernel with an isotropically diagonal covariance matrix in eq. (4.3) with properly set bandwidths (either very small or very large) for each attribute, we will recover the formulation in [5] when the bandwidths tend to zero or infinity.
4.4 Hyper-parameter tuning: cross-validation (CV) strategies

There are a few free hyper-parameters in our approach (Section 4.2.3). To choose the hyper-parameters in the conventional cross-validation (CV) for multi-way classification, one splits the training data into several folds such that they share the same set of class labels with one another. Clearly, this strategy is not sensible for zero-shot learning as it does not imitate what actually happens at the test stage. We thus introduce a new strategy for performing CV, inspired by the hyper-parameter tuning in [156]. The key difference of the new scheme to the conventional CV is that we split the data into several folds such that the class labels of these folds are disjoint. For clarity, we denote the conventional CV as sample-wise CV and our scheme as class-wise CV. Figure 4.2(b) and 4.2(c) illustrate the two scenarios, respectively. We empirically compare them in Section 4.5. Note that several existing models [6, 42, 156, 220] also follow similar hyper-parameter tuning procedures.

4.5 Empirical studies

We conduct extensive empirical studies of our approach SynC for the conventional zero-shot learning setting on four benchmark datasets—Animal with Attributes (AwA) [105], CUB-200-2011 Birds (CUB) [187], SUN Attribute (SUN) [142], and the full ImageNet Fall 2011 dataset [38] with more than 20,000 unseen classes.
Table 4.1: Key characteristics of studied datasets

<table>
<thead>
<tr>
<th>Dataset name</th>
<th># of seen classes</th>
<th># of unseen classes</th>
<th>Total # of images</th>
</tr>
</thead>
<tbody>
<tr>
<td>AwA†</td>
<td>40</td>
<td>10</td>
<td>30,475</td>
</tr>
<tr>
<td>CUB‡</td>
<td>150</td>
<td>50</td>
<td>11,788</td>
</tr>
<tr>
<td>SUN‡</td>
<td>645/646</td>
<td>72/71</td>
<td>14,340</td>
</tr>
<tr>
<td>ImageNet.§</td>
<td>1,000</td>
<td>20,842</td>
<td>14,197,122</td>
</tr>
</tbody>
</table>

†: Following the prescribed split in [106].
‡: 4 (or 10, respectively) random splits, reporting average.
§: Seen and unseen classes from ImageNet ILSVRC 2012 1K [157] and Fall 2011 release [38, 50, 135].

4.5.1 setup

Datasets We use four benchmark datasets in our experiments: the Animals with Attributes (AwA) [106], CUB-200-2011 Birds (CUB) [187], SUN Attribute (SUN) [142], and the ImageNet (with full 21,841 classes) [38]. Table 4.1 summarizes their key characteristics.

Semantic spaces For the classes in AwA, we use 85-dimensional binary or continuous attributes [106], as well as the 100 and 1,000 dimensional word vectors [130], derived from their class names and extracted by Fu et al. [51, 53]. For CUB and SUN, we use 312 and 102 dimensional continuous-valued attributes, respectively. We also thresh them at the global means to obtain binary-valued attributes, as suggested in [106]. Neither datasets have word vectors for their class names. For ImageNet, we train a skip-gram language model [130, 131] on the latest Wikipedia dump corpus\(^2\) (with more than 3 billion words) to extract a 500-dimensional word vector for each class. We ignore classes without word vectors in the experiments, resulting in 20,345 (out of 20,842) unseen classes. We also derive 21,632 dimensional semantic vectors of the class names using multidimensional scaling (MDS) on the WordNet hierarchy, as in [123]. For both the continuous attribute vectors and the word vector embeddings of the class names, we normalize them to have unit \(\ell_2\) norms unless stated otherwise.

Visual features Due to variations in features being used in literature, it is impractical to try all possible combinations of features and methods. Thus, we make a major distinction in using shallow features (such as color histograms, SIFT, PHOG, Fisher vectors) [5, 6, 83, 106, 155, 192] and deep learning features in several recent studies of zero-shot learning. Whenever possible, we use (shallow) features provided by those datasets or prior studies. For comparative studies, we also extract the following deep features: AlexNet [101] for AwA and CUB and GoogLeNet [175] for all datasets (all extracted with the Caffe package [85]). For AlexNet, we use the 4,096-dimensional activations of the penultimate layer (fc7) as features. For GoogLeNet, we take the 1,024-dimensional activations of the pooling units, as in [6].

\(^2\)http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2 on September 1, 2015
**Evaluation protocols**  For AwA, CUB, and SUN, we use the (normalized, by class-size) multi-way classification accuracy, as in previous work. Note that the accuracy is always computed on images from unseen classes.

Evaluating zero-shot learning on the large-scale ImageNet requires substantially different components from evaluating on the other three datasets. First, two evaluation metrics are used, as in [50]: Flat hit@K (F@K) and Hierarchical precision@K (HP@K).

F@K is defined as the percentage of test images for which the model returns the true label in its top K predictions. Note that, F@1 is the multi-way classification accuracy. HP@K takes into account the hierarchical organization of object categories. For each true label, we generate a ground-truth list of K closest categories in the hierarchy and compute the degree of overlapping (i.e., precision) between the ground-truth and the model’s top K predictions. For the detailed description of this metric, please see the Appendix of [50].

Secondly, following the procedure in [50, 135], we evaluate on three scenarios of increasing difficulty:

- **2-hop** contains 1,509 unseen classes that are within two tree hops of the seen 1K classes according to the ImageNet label hierarchy.
- **3-hop** contains 7,678 unseen classes that are within three tree hops of seen classes.
- **All** contains all 20,345 unseen classes in the ImageNet 2011 21K dataset that are not in the ILSVRC 2012 1K dataset.

The numbers of unseen classes are slightly different from what are used in [50, 135] due to the missing semantic representations (i.e., word vectors) for certain class names.

In addition to reporting published results, we have also reimplemented the state-of-the-art method ConSE [135] on this dataset.

### 4.5.2 Implementation details

We cross-validate all hyperparameters. For convenience, we set the number of phantom classes $R$ to be the same as the number of seen classes $S$, and set $b_r = a_c$ for $r = c$. We also experiment setting different $R$ and learning $b_r$. Our study (cf. Fig. 4.3) shows that when $R$ is about 60% of $S$, the performance saturates. We denote the three variants of our methods in constructing classifiers (Section 4.2.3) by Ours$^{o\text{-}vs\text{-}o}$ (one-versus-other), Ours$^{cs}$ (Crammer-Singer) and Ours$^{struct}$ (Crammer-Singer with structured loss).

### 4.5.3 Main results

Table 4.2 compares the proposed methods to the state-of-the-art from the previously published results on benchmark datasets. While there is a large degree of variations in implementation details, the main observation is that our methods attain the best performance in most scenarios. In what follows, we analyze those results in detail.

We also point out that the settings in some existing work are highly different from ours; we do not include their results for fair comparison [7, 51, 53, 55, 82, 97, 113, 212]. In some cases, even with additional data and attributes, those methods under-perform ours.

### Table 4.2: Comparison between our results and the previously published results in multi-way classification accuracies (in %) on the task of zero-shot learning. For each dataset, the best is in red and the 2nd best is in blue.

<table>
<thead>
<tr>
<th>Methods</th>
<th>AwA</th>
<th>CUB</th>
<th>SUN</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAP [106]</td>
<td>41.4</td>
<td>-</td>
<td>22.2</td>
<td>-</td>
</tr>
<tr>
<td>IAP [106]</td>
<td>42.2</td>
<td>-</td>
<td>18.0</td>
<td>-</td>
</tr>
<tr>
<td>BN [192]</td>
<td>43.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ALE [5]</td>
<td>37.4</td>
<td>18.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SJE [6]</td>
<td>66.7</td>
<td>50.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESZSL [156]</td>
<td>49.3</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ConSE[135]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.4</td>
</tr>
<tr>
<td>SSE-ReLU [220]*</td>
<td>76.3</td>
<td>30.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[221]*</td>
<td>80.5</td>
<td>42.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours^vs-o</td>
<td>69.7</td>
<td>53.4</td>
<td>62.8</td>
<td>1.4</td>
</tr>
<tr>
<td>Ours^s</td>
<td>68.4</td>
<td>51.6</td>
<td>52.9</td>
<td></td>
</tr>
<tr>
<td>Ours^struct</td>
<td>72.9</td>
<td>54.7</td>
<td>62.7</td>
<td>1.5</td>
</tr>
</tbody>
</table>

†: Results reported on a particular seen-unseen split.

*: Results were just brought to our attention. Note that VGG [170] instead of GoogLeNet features were used, improving on AwA but worsening on CUB.

### Table 4.3: Comparison between results by ConSE and our method on ImageNet. For both types of metrics, the higher the better.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Methods</th>
<th>Flat Hit@K</th>
<th>Hierarchical precision@K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1 2 5 10 20</td>
<td>2 5 10 20</td>
</tr>
<tr>
<td>2-hop</td>
<td>ConSE [135]</td>
<td>9.4 15.1 24.7 32.7 41.8</td>
<td>21.4 24.7 26.9 28.4</td>
</tr>
<tr>
<td></td>
<td>ConSE by us</td>
<td>8.3 12.9 21.8 30.9 41.7</td>
<td>21.5 23.8 27.5 31.3</td>
</tr>
<tr>
<td></td>
<td>Ours^vs-o</td>
<td>10.5 16.7 28.6 40.1 52.0</td>
<td>25.1 27.7 30.3 32.1</td>
</tr>
<tr>
<td></td>
<td>Ours^struct</td>
<td>9.8 15.3 25.8 35.8 46.5</td>
<td>23.8 25.8 28.2 29.6</td>
</tr>
<tr>
<td>3-hop</td>
<td>ConSE [135]</td>
<td>2.7 4.4 7.8 11.5 16.1</td>
<td>5.3 20.2 22.4 24.7</td>
</tr>
<tr>
<td></td>
<td>ConSE by us</td>
<td>2.6 4.1 7.3 11.1 16.4</td>
<td>6.7 21.4 23.8 26.3</td>
</tr>
<tr>
<td></td>
<td>Ours^vs-o</td>
<td>2.9 4.9 9.2 14.2 20.9</td>
<td>7.4 23.7 26.4 28.6</td>
</tr>
<tr>
<td></td>
<td>Ours^struct</td>
<td>2.9 4.7 8.7 13.0 18.6</td>
<td>8.0 22.8 25.0 26.7</td>
</tr>
<tr>
<td>All</td>
<td>ConSE [135]</td>
<td>1.4 2.2 3.9 5.8 8.3</td>
<td>2.5 7.8 9.2 10.4</td>
</tr>
<tr>
<td></td>
<td>ConSE by us</td>
<td>1.3 2.1 3.8 5.8 8.7</td>
<td>3.2 9.2 10.7 12.0</td>
</tr>
<tr>
<td></td>
<td>Ours^vs-o</td>
<td>1.4 2.4 4.5 7.1 10.9</td>
<td>3.1 9.0 10.9 12.5</td>
</tr>
<tr>
<td></td>
<td>Ours^struct</td>
<td>1.5 2.4 4.4 6.7 10.0</td>
<td>3.6 9.6 11.0 12.2</td>
</tr>
</tbody>
</table>
Table 4.4: Comparison between sample- and class-wise cross-validation for hyper-parameter tuning on CUB (learning with the one-versus-other loss).

<table>
<thead>
<tr>
<th>CV Scenarios</th>
<th>CUB (AlexNet)</th>
<th>CUB (GoogLeNet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample-wise</td>
<td>44.7</td>
<td>52.0</td>
</tr>
<tr>
<td>Class-wise</td>
<td>46.6</td>
<td>53.4</td>
</tr>
</tbody>
</table>

4.5.4 Large-scale zero-shot learning

One major limitation of most existing work on zero-shot learning is that the number of unseen classes is often small, dwarfed by the number of seen classes. However, real-world computer vision systems need to face a very large number of unseen objects. To this end, we evaluate our methods on the large-scale ImageNet dataset.

Table 4.3 summarizes our results and compares to the ConSE method [135], which is, to the best of our knowledge, the state-of-the-art method on this dataset.\footnote{We are aware of recent work by Lu [123] that introduces a novel form of semantic representations.} Note that in some cases, our own implementation ("ConSE by us" in the table) performs slightly worse than the reported results, possibly attributed to differences in visual features, word vector embeddings, and other implementation details. Nonetheless, the proposed methods (using the same setting as “ConSE by us”) always outperform both, especially in the very challenging scenario of All where the number of unseen classes is 20,345, significantly larger than the number of seen classes. Note also that, for both types of metrics, when $K$ is larger, the improvement over the existing approaches is more pronounced. It is also not surprising to notice that as the number of unseen classes increases from the setting 2-hop to All, the performance of both our methods and ConSE degrade.

4.5.5 Detailed analysis

We experiment extensively to understand the benefits of many factors in our and other algorithms. While trying all possible combinations is prohibitively expensive, we have provided a comprehensive set of results for comparison and drawing conclusions.

Cross-validation (CV) strategies Table 4.4 shows the results on CUB (averaged over four splits) using the hyper-parameters tuned by class-wise CV and sample-wise CV, respectively. The results based on class-wise CV are about 2% better than those of sample-wise CV, verifying the necessity of simulating the zero-shot learning scenario while we tune the hyper-parameters at the training stage.

Advantage of continuous attributes It is clear from Table 4.5 that, in general, continuous attributes as semantic representations for classes attain better performance than binary attributes. This is especially true when deep learning features are used to construct classifiers. It is somewhat
Table 4.5: Detailed analysis of various methods: the effect of feature and attribute types on multi-way classification accuracies (in %). Within each column, the best is in red and the 2nd best is in blue. We cite both previously published results (numbers in bold italics) and results from our implementations of those competing methods (numbers in normal font) to enhance comparability and to ease analysis (see texts for details). We use the shallow features provided by [106], [83], [142] for AwA, CUB, SUN, respectively.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Attribute type</th>
<th>Shallow features</th>
<th>Deep features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AwA</td>
<td>CUB</td>
</tr>
<tr>
<td>DAP [106]</td>
<td>binary</td>
<td><strong>41.4</strong></td>
<td>28.3</td>
</tr>
<tr>
<td>IAP [106]</td>
<td>binary</td>
<td><strong>42.2</strong></td>
<td>24.4</td>
</tr>
<tr>
<td>BN [192]</td>
<td>binary</td>
<td><strong>43.4</strong></td>
<td>-</td>
</tr>
<tr>
<td>ALE [5]†</td>
<td>binary</td>
<td><strong>37.4</strong></td>
<td><strong>18.0</strong></td>
</tr>
<tr>
<td>ALE</td>
<td>binary</td>
<td>34.8</td>
<td>27.8</td>
</tr>
<tr>
<td>SJE [6]</td>
<td>continuous</td>
<td><strong>42.3</strong>‡</td>
<td><strong>19.0</strong>†‡</td>
</tr>
<tr>
<td>SJE</td>
<td>continuous</td>
<td>36.2</td>
<td>34.6</td>
</tr>
<tr>
<td>ESZSL [156]§</td>
<td>continuous</td>
<td><strong>49.3</strong></td>
<td>37.0</td>
</tr>
<tr>
<td>ESZSL</td>
<td>continuous</td>
<td>44.1</td>
<td>38.3</td>
</tr>
<tr>
<td>ConSE [135]</td>
<td>continuous</td>
<td>36.5</td>
<td>23.7</td>
</tr>
<tr>
<td>COSTA [127]♯</td>
<td>continuous</td>
<td>38.9</td>
<td>28.3</td>
</tr>
<tr>
<td>Ours&lt;sup&gt;o-vs-o&lt;/sup&gt;</td>
<td>continuous</td>
<td>42.6</td>
<td>35.0</td>
</tr>
<tr>
<td>Ours&lt;sup&gt;cs&lt;/sup&gt;</td>
<td>continuous</td>
<td>42.1</td>
<td>34.7</td>
</tr>
<tr>
<td>Ours&lt;sup&gt;struct&lt;/sup&gt;</td>
<td>continuous</td>
<td>41.5</td>
<td>36.4</td>
</tr>
</tbody>
</table>

†: Results reported by the authors on a particular seen-unseen split.
‡: Based on Fisher vectors as shallow features, different from those provided in [83, 106, 142].
§: On the attribute vectors without $\ell_2$ normalization, while our own implementation shows that normalization helps in some cases.
♯: As co-occurrence statistics are not available, we combine pre-trained classifiers with the weights defined in eq. (4.2).
Table 4.6: Effect of types of semantic representations on AwA.

<table>
<thead>
<tr>
<th>Semantic representations</th>
<th>Dimensions</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>word vectors</td>
<td>100</td>
<td>42.2</td>
</tr>
<tr>
<td>word vectors</td>
<td>1000</td>
<td>57.5</td>
</tr>
<tr>
<td>attributes</td>
<td>85</td>
<td>69.7</td>
</tr>
<tr>
<td>attributes + word vectors</td>
<td>185</td>
<td>73.2</td>
</tr>
<tr>
<td>attributes + word vectors</td>
<td>1085</td>
<td><strong>76.3</strong></td>
</tr>
</tbody>
</table>

Table 4.7: Effect of learning semantic representations

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Types of embeddings</th>
<th>w/o learning</th>
<th>w/ learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AwA</td>
<td>attributes</td>
<td>69.7%</td>
<td>71.1%</td>
</tr>
<tr>
<td></td>
<td>100-d word vectors</td>
<td>42.2%</td>
<td>42.5%</td>
</tr>
<tr>
<td></td>
<td>1000-d word vectors</td>
<td>57.6%</td>
<td>56.6%</td>
</tr>
<tr>
<td>CUB</td>
<td>attributes</td>
<td>53.4%</td>
<td>54.2%</td>
</tr>
<tr>
<td>SUN</td>
<td>attributes</td>
<td>62.8%</td>
<td>63.3%</td>
</tr>
</tbody>
</table>

expected that continuous attributes provide a more accurate real-valued similarity measure among classes. This presumably is exploited further by more powerful classifiers.

**Advantage of deep features**  It is also clear from Table 4.5 that, across all methods, deep features significantly boost the performance based on shallow features. We use GoogLeNet and AlexNet (numbers in parentheses) and GoogLeNet generally outperforms AlexNet. It is worthwhile to point out that the reported results under deep features columns are obtained using linear classifiers, which outperform several nonlinear classifiers that use shallow features. This seems to suggest that deep features, often thought to be specifically adapted to seen training images, still work well when transferred to unseen images [50].

**Which types of semantic space?**  In Table 4.6, we show how effective our proposed method (Ours\textsuperscript{O-o}) exploits the two types of semantic spaces: (continuous) attributes and word-vector embeddings on AwA (the only dataset with both embedding types). We find that attributes yield better performance than word-vector embeddings. However, combining the two gives the best result, suggesting that these two semantic spaces could be complementary and further investigation is ensured.

Table 4.7 takes a different view on identifying the best semantic space. We study whether we can learn optimally the semantic representations (cf. Section 4.2.3) for the phantom classes that correspond to base classifiers. These preliminary studies seem to suggest that learning attributes could have a positive effect, though it is difficult to improve over word-vector embeddings. We plan to study this issue more thoroughly in the future.

**How many base classifiers are necessary?**  In Fig. 4.3, we investigate how many base classifiers are needed—so far, we have set that number to be the number of seen classes out of convenience. The plot shows that in fact, a smaller number (about 60% -70%) is enough for our
4.5.6 Qualitative results

In this subsection, we present qualitative results of our method. We first illustrate what visual information the models (classifiers) for unseen classes capture, when provided with only semantic embeddings (no example images). In Figure 4.4, we list (on top) the 10 unseen class labels of AwA, and show (in the middle) the top-5 images classified into each class $c$, according to the decision values $w_c^T x$. Misclassified images are marked with red boundaries. At the bottom, we show the first (highest score) misclassified image (according to the decision value) into each class and its ground-truth class label. According to the top images, our method reasonably captures discriminative visual properties of each unseen class based solely on its semantic embedding. We can also see that the misclassified images are with appearance so similar to that of predicted class that even humans cannot easily distinguish between the two. For example, the pig image at the bottom of the second column looks very similar to the image of hippos.

4.6 Summary

We have developed a novel classifier synthesis mechanism (SynC) for zero-shot learning by introducing the notion of “phantom” classes. The phantom classes connect the dots between the seen and unseen classes—the classifiers of the seen and unseen classes are constructed from the same base classifiers for the phantom classes and with the same coefficient functions. As a result, we can conveniently learn the classifier synthesis mechanism leveraging labeled data of the seen classes and then readily apply it to the unseen classes. SynC is conceptually clean, and flexible to incorporate various forms of similarity functions, classifiers, and semantic representations—by certain combinations, SynC can recover several existing methods, essentially a superset of them. Moreover, it is widely applicable to different visual recognition tasks, including fine-grained object, scene, and large-scale object recognition. Specifically, on the setting proposed by Google [50], where over 20,000 unseen categories are to be recognized, SynC so far holds the best performance and outperforms other methods by a margin, as reported in a recent survey [201].
Figure 4.4: Qualitative results of our method (Ours\textsuperscript{struct}) on AwA. **(Top)** We list the 10 unseen class labels. **(Middle)** We show the top-5 images classified into each class, according to the decision values. *Misclassified images are marked with red boundaries.* **(Bottom)** We show the first misclassified image (according to the decision value) into each class and its ground-truth class label.
Chapter 5

Generalized Zero-Shot Learning

In the previous chapter, we introduce our zero-shot learning algorithm SynC, which achieves superior performance on four benchmark datasets for the conventional setting—once models for unseen classes are constructed, they are judged based on their ability to discriminate among unseen classes, assuming the absence of seen objects during the test phase. Originally proposed in the seminal work of Lampert et al. [105], this setting has almost always been adopted for evaluating ZSL methods [138, 214, 154, 90, 5, 212, 50, 127, 135, 82, 7, 6, 53, 55, 113, 156, 97, 220, 221].

But, does this problem setting truly reflect what recognition in the wild entails? While the ability to learn novel concepts is by all means a trait that any zero-shot learning systems should possess, it is merely one side of the coin. The other important—yet so far under-studied—trait is the ability to remember past experiences, i.e., the seen classes.

Why is this trait desirable? Consider how data are distributed in the real world. The seen classes are often more common than the unseen ones; it is therefore unrealistic to assume that we will never encounter them during the test stage. For models generated by ZSL to be truly useful, they should not only accurately discriminate among either seen or unseen classes themselves but also accurately discriminate between the seen and unseen ones.

Thus, to understand better how existing ZSL approaches will perform in the real world, we advocate evaluating them in the setting of generalized zero-shot learning (GZSL), where test data are from both seen and unseen classes and we need to classify them into the joint labeling space of both types of classes. Previous work in this direction is scarce. See Chapter 3 for more details.

5.1 Overview

We conduct an extensive empirical study of several existing ZSL approaches in the new GZSL setting. We show that a straightforward application of classifiers constructed by those approaches performs poorly. In particular, test data from unseen classes are almost always classified as a class from the seen ones. We propose a surprisingly simple yet very effective method called calibrated stacking to address this problem. This method is mindful of the two conflicting forces: recognizing data from seen classes and recognizing data from unseen ones. We introduce a new performance metric called Area Under Seen-Unseen accuracy Curve (AUSUC) that can evaluate ZSL approaches on how well they can trade off between the two. We demonstrate the utility of this metric by evaluating several representative ZSL approaches under this metric on the benchmark datasets considered in the experiments of Chapter 4.
Figure 5.1: Comparisons of (a) conventional ZSL and (b) generalized ZSL in the testing phase—conventional ZSL assumes the absence of seen classes’ instances and only classifies test instances into one of the unseen classes. The notations follow those in Section 5.2.2.

5.2 Generalized zero-shot learning

In this section, we review the setting of generalized zero-shot learning that has been defined in Chapter 2. We then present empirical evidence to illustrate the difficulty of this problem.

5.2.1 Conventional and generalized zero-shot learning

Suppose we are given the training data $D = \{(x_n \in \mathbb{R}^D, y_n)\}_{n=1}^N$ with the labels $y_n$ from the label space of seen classes $S = \{1, 2, \cdots, S\}$. Denote by $U = \{S + 1, \cdots, S + U\}$ the label space of unseen classes. We use $T = S \cup U$ to represent the union of the two sets of classes.

In the (conventional) zero-shot learning (ZSL) setting, the main goal is to classify test data into the unseen classes, assuming the absence of the seen classes in the test phase. In other words, each test data point is assumed to come from and will be assigned to one of the labels in $U$.

Existing research on ZSL has been almost entirely focusing on this setting [105, 138, 214, 154, 90, 5, 212, 50, 127, 82, 7, 6, 53, 55, 113, 156, 97, 220, 221]. However, in real applications, the assumption of encountering data only from the unseen classes is hardly realistic. The seen classes are often the most common objects we see in the real world. Thus, the objective in the conventional ZSL does not truly reflect how the classifiers will perform recognition in the wild.

Motivated by this shortcomings of the conventional ZSL, we advocate studying the more general setting of generalized zero-shot learning (GZSL), where we no longer limit the possible class memberships of test data—each of them belongs to one of the classes in $T$. (See Fig. 5.1.)
5.2.2 Classifiers

Without the loss of generality, we assume that for each class $c \in T$, we have a discriminant scoring function $f_c(x)$ (or more generally $f(a_c, x)$), from which we would be able to derive the label for $x$. For instance, for an unseen class $u$, SynC defines $f_u(x) = w_u^T x$, where $w_u$ is the model parameter vector for the class $u$, constructed from its semantic representation $a_u$ (such as its attribute vector or the word vector associated with the name of the class). In ConSE [135], $f_u(x) = \cos(m(x), a_u)$, where $m(x)$ is the predicted embedding of the data sample $x$. In DAP/IAP [106], $f_u(x)$ is a probabilistic model of attribute vectors. We assume that similar discriminant functions for seen classes can be constructed in the same manner given their corresponding semantic representations.

How to assess an algorithm for GZSL? We define and differentiate the following performance metrics: $A_{U \rightarrow U}$ the accuracy of classifying test data from $U$ into $U$, $A_{S \rightarrow S}$ the accuracy of classifying test data from $S$ into $S$, and finally $A_{S \rightarrow T}$ and $A_{U \rightarrow T}$ the accuracies of classifying test data from either seen or unseen classes into the joint labeling space. Note that $A_{U \rightarrow U}$ is the standard performance metric used for conventional ZSL and $A_{S \rightarrow S}$ is the standard metric for multi-class classification. Furthermore, note that we do not report $A_{T \rightarrow T}$ as simply averaging $A_{S \rightarrow T}$ and $A_{U \rightarrow S}$ to compute $A_{T \rightarrow T}$ might be misleading when the two metrics are not balanced, as shown below.

5.2.3 Generalized ZSL is hard

To demonstrate the difficulty of GZSL, we report the empirical results of using a simple but intuitive algorithm for GZSL. Given the discriminant functions, we adopt the following classification rule

$$\hat{y} = \arg \max_{c \in T} f_c(x) = \arg \max_{c \in T} f(a_c, x) \quad (5.1)$$

which we refer to as direct stacking.

We use the rule on “stacking” classifiers from the following zero-shot learning approaches: DAP and IAP [106], ConSE [135], and Synthesized Classifiers (SynC). We tune the hyper-parameters for each approach based on class-wise cross validation. We test GZSL on two datasets AwA [106] and CUB [187]—details about those datasets can be found in Section 4.5.

Table 5.1 reports experimental results based on the 4 performance metrics we have described previously. Our goal here is not to compare between methods. Instead, we examine the impact of relaxing the assumption of the prior knowledge of whether data are from seen or unseen classes.

We observe that, in this setting of GZSL, the classification performance for unseen classes ($A_{U \rightarrow T}$) drops significantly from the performance in conventional ZSL ($A_{U \rightarrow U}$), while that of seen ones ($A_{S \rightarrow T}$) remains roughly the same as in the multi-class task ($A_{S \rightarrow S}$). That is, nearly all test data from unseen classes are misclassified into the seen classes. This unusual degradation in performance highlights the challenges of GZSL; as we only see labeled data from seen classes during training, the scoring functions of seen classes tend to dominate those of unseen classes, leading to biased predictions in GZSL and aggressively classifying a new data point into the label

\footnote{Note that in SynC, $w_u$ is a function of $a_u$ under fixed coordinates of phantom classes. Therefore, we can also view $w_u^T x$ as $f(a_u, x)$}
Table 5.1: Classification accuracies (%) on conventional ZSL ($A_{U\rightarrow U}$), multi-class classification for seen classes ($A_{S\rightarrow S}$), and GZSL ($A_{S\rightarrow T}$ and $A_{U\rightarrow T}$), on AwA and CUB. Significant drops are observed from $A_{U\rightarrow U}$ to $A_{U\rightarrow T}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>AwA $A_{U\rightarrow U}$</th>
<th>AwA $A_{S\rightarrow S}$</th>
<th>AwA $A_{U\rightarrow T}$</th>
<th>CUB $A_{U\rightarrow U}$</th>
<th>CUB $A_{S\rightarrow S}$</th>
<th>CUB $A_{U\rightarrow T}$</th>
<th>CUB $A_{S\rightarrow T}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAP [106]</td>
<td>51.1</td>
<td>178.5</td>
<td>177.9</td>
<td>38.8</td>
<td>56.0</td>
<td>4.0</td>
<td>55.1</td>
</tr>
<tr>
<td>IAP [106]</td>
<td>56.3</td>
<td>77.3</td>
<td>1.7</td>
<td>76.8</td>
<td>36.5</td>
<td>69.6</td>
<td>1.0</td>
</tr>
<tr>
<td>ConSE [135]</td>
<td>63.7</td>
<td>76.9</td>
<td>9.5</td>
<td>75.9</td>
<td>35.8</td>
<td>70.5</td>
<td>1.8</td>
</tr>
<tr>
<td>SynCo-vso</td>
<td>70.1</td>
<td>67.3</td>
<td>0.3</td>
<td>67.3</td>
<td>53.0</td>
<td>67.2</td>
<td>8.4</td>
</tr>
<tr>
<td>SynCstruct</td>
<td>73.4</td>
<td>81.0</td>
<td>0.4</td>
<td>81.0</td>
<td>54.4</td>
<td>73.0</td>
<td>13.2</td>
</tr>
</tbody>
</table>

space of $S$ because classifiers for the seen classes do not get trained on “negative” examples from the unseen classes.

5.3 Approach for GZSL

The previous example shows that the classifiers for unseen classes constructed by conventional ZSL methods should not be naively combined with models for seen classes to expand the labeling space required by GZSL.

In what follows, we propose a simple variant to the naive approach of direct stacking to curb such a problem. We also develop a metric that measures the performance of GZSL, by acknowledging that there is an inherent trade-off between recognizing seen classes and recognizing unseen classes. This metric, referred to as the Area Under Seen-Unseen accuracy Curve (AUSUC), balances the two conflicting forces. We conclude this section by describing two related approaches: despite their sophistication, they do not perform well empirically.

5.3.1 Calibrated stacking

Our approach stems from the observation that the scores of the discriminant functions for the seen classes are often greater than the scores for the unseen classes. Thus, intuitively, we would like to reduce the scores for the seen classes. This leads to the following classification rule:

$$\hat{y} = \arg\max_{c \in T} f_c(x) - \gamma [c \in S],$$

(5.2)

where the indicator $[c \in S] \in \{0, 1\}$ indicates whether or not $c$ is a seen class and $\gamma$ is a calibration factor. We term this adjustable rule as calibrated stacking. See Fig. 5.2 for an illustration.

Another way to interpret $\gamma$ is to regard it as the prior likelihood of a data point coming from unseen classes. When $\gamma = 0$, the calibrated stacking rule reverts back to the direct stacking rule, described previously.

It is also instructive to consider the two extreme cases of $\gamma$. When $\gamma \rightarrow +\infty$, the classification rule will ignore all seen classes and classify all data points into one of the unseen classes. When there is no new data point coming from seen classes, this classification rule essentially implements what one would do in the setting of conventional ZSL. On the other hand, when $\gamma \rightarrow -\infty$, the classification rule only considers the label space of seen classes as in standard multi-way
Figure 5.2: We observed that seen classes usually give higher scores than unseen classes, even to an unseen class instance (e.g., a zebra image). We thus introduce a calibration factor $\gamma$, either to reduce the scores of seen classes or to increase those of unseen classes (cf. eq. (5.2)).

Figure 5.3: The Seen-Unseen accuracy Curve (SUC) obtained by varying $\gamma$ in the calibrated stacking classification rule eq. (5.2). The AUSUC summarizes the curve by computing the area under it. We use the method SynC$^o$-vs-o on the AwA dataset, and tune hyper-parameters as in Table 5.1. The red cross denotes the accuracies by direct stacking.

classification. The calibrated stacking rule thus represents a middle ground between aggressively classifying every data point into seen classes and conservatively classifying every data point into unseen classes. Adjusting this hyperparameter thus gives a trade-off, which we exploit to define a new performance metric.

5.3.2 Area Under Seen-Unseen Accuracy Curve (AUSUC)

Varying the calibration factor $\gamma$, we can compute a series of classification accuracies ($A_{U \rightarrow T}$, $A_{S \rightarrow T}$). Fig. 5.3 plots those points for the dataset AwA using the classifiers generated by SynC based on class-wise cross validation. We call such a plot the Seen-Unseen accuracy Curve (SUC).

On the curve, $\gamma = 0$ corresponds to direct stacking, denoted by a cross. The curve is similar to many familiar curves for representing conflicting goals, such as the Precision-Recall (PR) curve and the Receiving Operator Characteristic (ROC) curve, with two ends for the extreme cases ($\gamma \rightarrow -\infty$ and $\gamma \rightarrow +\infty$).
A convenient way to summarize the plot with one number is to use the Area Under SUC (AUSUC). The higher the area is, the better an algorithm is able to balance $A_{\mathcal{U} \rightarrow \mathcal{T}}$ and $A_{\mathcal{S} \rightarrow \mathcal{T}}$. We evaluate the performance of existing zero-shot learning methods under this metric, as well as provide further insights and analyses in Section 5.4.

An immediate and important use of the metric AUSUC is for model selection. Many ZSL learning methods require tuning hyperparameters—previous work tune them based on the accuracy $A_{\mathcal{U} \rightarrow \mathcal{U}}$. The selected model, however, does not necessarily balance optimally between $A_{\mathcal{U} \rightarrow \mathcal{T}}$ and $A_{\mathcal{S} \rightarrow \mathcal{T}}$. Instead, we advocate using AUSUC for model selection and hyperparameter tuning. Models with higher values of AUSUC are likely to perform in balance for the task of GZSL. We provide detailed discussions in Section 5.4.2.

5.3.3 Comparisons to alternative approaches

As introduced in Chapter 3, Socher et al. [172] propose a two-stage zero-shot learning approach that first predicts whether an image is of seen or unseen classes according to certain novelty scores, and then accordingly applies the corresponding classifiers. If we define a new form of novelty score $N(x) = \max_{u \in \mathcal{U}} f_u(x) - \max_{s \in \mathcal{S}} f_s(x)$ in eq. (3.15), we recover the prediction rule in eq. (5.2). However, this relation holds only if we are interested in predicting one label $\hat{y}$. When we are interested in predicting a set of labels (for example, hoping that the correct labels are in the top $K$ predicted labels, (i.e., the Flat hit@K metric, cf. Section 5.4), the two prediction rules will give different results.

5.4 Empirical studies

5.4.1 Setup

Datasets, features, and semantic representations  We mainly use three benchmark datasets: the Animals with Attributes (AwA) [106], CUB-200-2011 Birds (CUB) [187], and ImageNet [157]. Please be refer to Section 4.5 for details. We use the GoogLeNet deep features.

Compared methods  We examine SynC and several representative conventional zero-shot learning approaches, described briefly below. Direct Attribute Prediction (DAP) and Indirect Attribute Prediction (IAP) [106] are probabilistic models that perform attribute predictions as an intermediate step and then use them to compute MAP predictions of unseen class labels. ConSE [135] makes use of pre-trained classifiers for seen classes and their probabilistic outputs to infer the semantic representations of each test example, and then classifies it into the unseen class with the most similar semantic representations. We use binary attributes for DAP and IAP, and continuous attributes and word2vec for ConSE and SynC, following [106, 135].

Generalized zero-shot learning tasks  There are no previously established benchmark tasks for GZSL. We thus define a set of tasks that reflects more closely how data are distributed in real-world applications.

\[2\] If a single $\gamma$ is desired, the “F-score” that balances $A_{\mathcal{U} \rightarrow \mathcal{T}}$ and $A_{\mathcal{S} \rightarrow \mathcal{T}}$ can be used.
We construct the GZSL tasks by composing test data as a combination of images from both seen and unseen classes. We follow existing splits of the datasets for the conventional ZSL to separate seen and unseen classes. Moreover, for the datasets AwA and CUB, we hold out 20% of the data points from the seen classes (previously, all of them are used for training in the conventional zero-shot setting) and merge them with the data from the unseen classes to form the test set; for ImageNet, we combine its validation set (having the same classes as its training set) and the 21K classes that are not in the ILSVRC 2012 1K dataset.

**Evaluation metrics**  While we will primarily report the performance of ZSL approaches under the metric Area Under Seen-Unseen accuracy Curve (AUSUC) developed in Section 5.3.1, we explain how its two accuracy components $A_{S \rightarrow T}$ and $A_{U \rightarrow T}$ are computed below.

For AwA and CUB, seen and unseen accuracies correspond to (normalized-by-class-size) multi-way classification accuracy, where the seen accuracy is computed on the 20% images from the seen classes and the unseen accuracy is computed on images from unseen classes.

For ImageNet, seen and unseen accuracies correspond to Flat hit@K (F@K), defined as the percentage of test images for which the model returns the true label in its top K predictions. Note that, F@1 is the unnormalized multi-way classification accuracy. Moreover, following the procedure in [50, 135], we evaluate on three scenarios of increasing difficulty: (1) 2-hop contains 1,509 unseen classes that are within two tree hops of the 1K seen classes according to the ImageNet label hierarchy\(^3\). (2) 3-hop contains 7,678 unseen classes that are within three tree hops of the seen classes. (3) All contains all 20,345 unseen classes.

### 5.4.2 Hyper-parameter tuning strategies

**Cross-validation with AUSUC**  In Section 5.3.2, we introduce the Area Under Seen-Unseen accuracy Curve (AUSUC), which is analogous to many metrics in computer vision and machine learning that balance two conflicting (sub)metrics, such as area under ROC. To tune the hyper-parameters based on this metric\(^4\), we simulate the generalized zero-shot learning setting during cross-validation.

Concretely, we split the training data into 5 folds A1, A2, A3, A4 and A5 so that the class labels of these folds are disjoint. We further split 80% and 20% of data from each fold (A1-A5, respectively) into pseudo-train and pseudo-test sets, respectively. We then combine the pseudo-train sets of four folds (for example, A1-A4) for training, and validate on (i) the pseudo-test sets of such four folds (i.e., A1-A4) and (ii) the pseudo-train set of the remaining fold (i.e., A5). That is, the remaining fold serves as the pseudo-unseen classes in cross-validation. We repeat this process for 5 rounds—each round selects a fold as the “remaining” fold, and computes AUSUC on the corresponding validation set. Finally, the average of AUSUCs over all rounds is used to select hyper-parameters.

**Comparison to an alternative strategy**  Another strategy for hyper-parameter tuning is to find two sets of hyper-parameters: one optimized for seen classes and the other for unseen classes. The

\(^3\)http://www.image-net.org/api/xml/structure_released.xml

\(^4\)AUSUC is computed by varying the $\gamma$ factor within a range. If a single $\gamma$ is desired, another measure such as “F-score” balancing $A_{U \rightarrow T}$ and $A_{S \rightarrow T}$ can be used. One can also assume a prior probability of whether any instance is seen or unseen to select the factor.
Table 5.2: Comparison of performance measured in AUSUC between two cross-validation strategies on AwA and CUB. One strategy is based on accuracies ($A_{S\rightarrow S}$ and $A_{U\rightarrow U}$) and the other is based on AUSUC. See text for details.

<table>
<thead>
<tr>
<th>Method</th>
<th>AwA</th>
<th>CUB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CV strategies</td>
<td>CV strategies</td>
</tr>
<tr>
<td></td>
<td>Accuracies AUSUC</td>
<td>Accuracies AUSUC</td>
</tr>
<tr>
<td>DAP [106]</td>
<td>0.341 0.366</td>
<td>0.202 0.194</td>
</tr>
<tr>
<td>IAP [106]</td>
<td>0.366 0.394</td>
<td>0.194 0.199</td>
</tr>
<tr>
<td>ConSE [135]</td>
<td><strong>0.443</strong> 0.428</td>
<td>0.190 0.212</td>
</tr>
<tr>
<td>SynC\textsuperscript{o-vs-o}</td>
<td>0.539 <strong>0.568</strong></td>
<td>0.324 <strong>0.336</strong></td>
</tr>
<tr>
<td>SynC\textsuperscript{struct}</td>
<td>0.551 <strong>0.583</strong></td>
<td>0.356 0.356</td>
</tr>
</tbody>
</table>

standard cross-validation technique, where $A_{S\rightarrow S}$ is optimized, can be used for the former. For the latter, it has been shown that the class-wise cross-validation technique, where the conventional zero-shot learning task is simulated, outperforms the standard technique. In this case, $A_{U\rightarrow U}$ is optimized. We thus use the first set of hyper-parameters to construct the scoring functions for the seen classes, and use the second set for the unseen classes (cf. Section 5.2).

In this subsection, we show that the strategy that jointly optimizes hyper-parameters based on AUSUC in most cases leads to better models for GZSL than the strategy that optimizes seen and unseen classifiers’ performances separately. On AwA and CUB, we perform 5-fold cross-validation based on the two strategies and compare the performance of those selected models in Table 5.2. In general, cross-validation based on AUSUC leads to better models for GZSL. In the following we thus stick with cross-validation with AUSUC.

5.4.3 Which method to use to perform GZSL?

Table 5.3 provides an experimental comparison between several methods utilizing seen and unseen classifiers for generalized ZSL, with hyperparameters cross-validated to maximize AUSUC.

The results show that, irrespective of which ZSL methods are used to generate models for seen and unseen classes, our method of calibrated stacking for generalized ZSL outperforms other methods. In particular, despite their probabilistic justification, the two novelty detection methods do not perform well. We believe that this is because most existing zero-shot learning methods are discriminative and optimized to take full advantage of class labels and semantic information. In contrast, either Gaussian or LoOP approach models all the seen classes as a whole, possibly at the cost of modeling inter-class differences.

5.4.4 Which zero-shot learning approach is more robust to GZSL?

Fig. 5.4 contrasts in detail several ZSL approaches when tested on the task of GZSL, using the method of calibrated stacking. Clearly, the SynC method dominates all other methods in the whole ranges. The crosses on the plots mark the results of direct stacking (Section 5.2).

\textsuperscript{5}The exceptions are ConSE on AwA and DAP on CUB.
Table 5.3: Performances measured in AUSUC of several methods for Generalized Zero-Shot Learning on AwA and CUB. The higher the better (the upper bound is 1).

<table>
<thead>
<tr>
<th>Method</th>
<th>AwA</th>
<th>CUB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Novelty detection [172]</td>
<td>Novelty detection [172]</td>
</tr>
<tr>
<td></td>
<td>Gaussian</td>
<td>Stacking</td>
</tr>
<tr>
<td>DAP</td>
<td>0.302</td>
<td>0.366</td>
</tr>
<tr>
<td>IAP</td>
<td>0.307</td>
<td>0.394</td>
</tr>
<tr>
<td>ConSE</td>
<td>0.342</td>
<td>0.428</td>
</tr>
<tr>
<td>SynC&lt;sub&gt;o-vs-o&lt;/sub&gt;</td>
<td>0.420</td>
<td>0.568</td>
</tr>
<tr>
<td>SynC&lt;sub&gt;struct&lt;/sub&gt;</td>
<td>0.424</td>
<td>0.583</td>
</tr>
</tbody>
</table>

Figure 5.4: Comparison between several ZSL approaches on the task of GZSL for AwA and CUB.

Table 5.4: Performances measured in AUSUC by different zero-shot learning approaches on GZSL on ImageNet, using our method of calibrated stacking.

<table>
<thead>
<tr>
<th>Unseen classes</th>
<th>Method</th>
<th>Flat hit@K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2-hop</td>
<td>ConSE</td>
<td>0.042</td>
</tr>
<tr>
<td></td>
<td>SynC&lt;sub&gt;o-vs-o&lt;/sub&gt;</td>
<td>0.044</td>
</tr>
<tr>
<td></td>
<td>SynC&lt;sub&gt;struct&lt;/sub&gt;</td>
<td>0.043</td>
</tr>
<tr>
<td>3-hop</td>
<td>ConSE</td>
<td>0.013</td>
</tr>
<tr>
<td></td>
<td>SynC&lt;sub&gt;o-vs-o&lt;/sub&gt;</td>
<td>0.012</td>
</tr>
<tr>
<td></td>
<td>SynC&lt;sub&gt;struct&lt;/sub&gt;</td>
<td>0.013</td>
</tr>
<tr>
<td>All</td>
<td>ConSE</td>
<td>0.007</td>
</tr>
<tr>
<td></td>
<td>SynC&lt;sub&gt;o-vs-o&lt;/sub&gt;</td>
<td>0.006</td>
</tr>
<tr>
<td></td>
<td>SynC&lt;sub&gt;struct&lt;/sub&gt;</td>
<td>0.007</td>
</tr>
</tbody>
</table>

Fig. 5.5 contrasts in detail ConSE to SynC, the two known methods for large-scale ZSL. When the accuracies measured in Flat hit@1 (i.e., multi-class classification accuracy), neither method dominates the other, suggesting the different trade-offs by the two methods. However, when we measure hit rates in the top $K > 1$, SynC dominates ConSE. Table 5.4 gives summarized
Figure 5.5: Comparison between ConSE and SynC of their performances on the task of GZSL for ImageNet where the unseen classes are within 2 tree-hops from seen classes.

comparison in AUSUC between the two methods on the ImageNet dataset. We observe that SynC in general outperforms ConSE except when Flat hit@1 is used, in which case the two methods’ performances are nearly indistinguishable.

5.5 Summary

Zero-shot learning (ZSL) methods have been studied in the unrealistic setting where test data are assumed to come from unseen classes only. In this chapter, we advocate studying the problem of generalized zero-shot learning (GZSL) where the test data’s class memberships are unconstrained. We show empirically that naively using the classifiers constructed by ZSL approaches does not perform well in the generalized setting. Motivated by this, we propose a simple but effective calibration method that can be used to balance two conflicting forces: recognizing data from seen classes versus those from unseen ones. We develop a performance metric to characterize such a trade-off and examine the utility of this metric in evaluating various ZSL approaches. SynC outperforms the compared ones. Starting from the work being published [28], much new work has been dedicated to the generalized setting, ranging from visual object recognition to video-based action recognition [98, 43, 204, 114, 33, 13, 215, 191, 25, 200, 15, 229, 224].
Chapter 6

From Zero-Shot Learning to Conventional Supervised Learning

The generalized zero-shot learning (GZSL) setting, approach, and evaluation metric introduced in the previous chapter allow us to realistically and fairly compare zero-shot learning with conventional supervised learning, in which for any class \( c \in \mathcal{T} \) we have labeled training data. This comparison is extremely important in understanding how far the current development of zero-shot learning techniques is from the performance that can be achieved by conventional supervised learning if we put more effort on collecting and labeling data.

To this end, we conduct a large-scale study including 1,000 seen and 1,000 or over 20,000 unseen classes. Our analysis shows a large gap between the GZSL approaches (using the existing semantic representations) and multi-class classification. We then demonstrate that by improving the representations to incorporate domain cues—e.g., peeking few instances of each category and treating the average features as the representations—such a gap can be largely reduced even using the same ZSL approaches, suggesting the next step to advance ZSL.

In the following, we start with the comparison among zero-shot, few-shot, and the conventional supervised learning paradigms. We then describe our experimental setup, present the results, and provide the key insights.

6.1 Comparisons among different learning paradigms

Conventional supervised learning for classification assumes that for all the categories of interest (i.e., \( \mathcal{T} \)), sufficient training examples are accessible. Zero-shot learning (ZSL), on the other hand, separates \( \mathcal{T} \) into two disjoint subsets \( \mathcal{S} \) and \( \mathcal{U} \), where for classes in \( \mathcal{U} \) no training examples are accessible. Such a separation can be applied to other learning paradigms like one-shot or few-shot learning [186, 171] as well, where for classes in \( \mathcal{U} \) only one or few training examples are accessible. See Fig 6.1 for an illustration. In this case, conventional supervised learning can also be viewed as many-shot learning.

To construct classifiers for \( \mathcal{T} \), in supervised learning we can directly train a multi-class classifier using the one-versus-other or Crammer-Singer loss mentioned in Section 4.2. For zero-shot learning, we leverage the class semantic representations to transfer the classifiers or discriminative knowledge from \( \mathcal{S} \) to \( \mathcal{U} \). For example, our SynC algorithm learns a mechanism (from training data of \( \mathcal{S} \)) to synthesize the classifier of any class given its semantic representation.

We note that SynC (and many other ZSL algorithms) is not designed for a specific type of semantic representations. That is, it can be applied to few-shot or many-shot learning as long as we have class semantic representations. While semantic representations are mostly not provided
Figure 6.1: The comparison of zero-shot, few-shot, and conventional supervised learning (i.e., many shot-learning). For all the paradigms, categories of interest can be separated into two portions: one with many training examples per class; one with zero, few, or again many examples. For ZSL, the first (second) portion is called seen (unseen) classes, and extra class semantic representations $a_c$ are provided. In our SynC algorithm, we learn a mechanism $h$ to synthesize the classifier $w_c$ given the corresponding $a_c$. We can actually learn and apply the same mechanism to the other mechanisms if we have $a_c$: for example, constructing $a_c$ by average visual features.

in these learning paradigms, we can indeed construct them using visual features—for example, by taking the average visual features of images of each class. In the following we then experiment with this idea to connect multiple learning paradigms and analyze the performance gap.

### 6.2 Empirical studies

#### 6.2.1 Overview

Zero-shot learning, either in conventional setting or generalized setting, is a challenging problem as there is no labeled data for the unseen classes. The performance of ZSL methods depends on at least two factors: (1) how seen and unseen classes are related; (2) how effectively the relation can be exploited by learning algorithms to generate models for the unseen classes. For generalized zero-shot learning, the performance further depends on how classifiers for seen and unseen classes are combined to classify new data into the joint label space.

Despite extensive study in ZSL, several questions remain understudied. For example, given a dataset and a split of seen and unseen classes, what is the best possible performance of any ZSL method? How far are we from there? What is the most crucial component we can improve in order to reduce the gap between the state-of-the-art and the ideal performances? In this section, we empirically analyze ZSL methods in detail and shed light on some of those questions.
6.2.2 Setup

As ZSL methods do not use labeled data from unseen classes for training classifiers, one reasonable estimate of their best possible performance is to measure the performance on a multi-class classification task where annotated data on the unseen classes are provided.

Concretely, to construct the multi-class classification task, on AwA and CUB, we randomly select 80% of the data along with their labels from all classes (seen and unseen) to train classifiers. The remaining 20% will be used to assess both the multi-class classifiers and the classifiers from ZSL. Note that, for ZSL, only the seen classes from the 80% are used for training—the portion belonging to the unseen classes are not used.

On ImageNet, to reduce the computational cost (of constructing multi-class classifiers which would involve 20,345-way classification), we subsample another 1,000 unseen classes from its original 20,345 unseen classes. We call this new dataset ImageNet-2K (including the 1K seen classes from ImageNet). Out of those 1,000 unseen classes, we randomly select 50 samples per class and reserve them for testing and use the remaining examples (along with their labels) to train 2000-way classifiers.

For ZSL methods, we use either attribute vectors or word vectors (WORD2VECTOR) as semantic representations. Since SynC<sup>o-vs-o</sup> performs well on a range of datasets and settings, we focus on this method. For multi-class classification, we train one-versus-others SVMs. Once we obtain the classifiers for both seen and unseen classes, we use the calibrated stacking decision rule to combine (as in generalized ZSL) and vary the calibration factor γ to obtain the Seen-Unseen accuracy Curve, exemplified in Fig. 5.3.

6.2.3 Results

How far are we from the ideal performance? Fig. 6.2 displays the accuracy Curves for ImageNet-2K<sup>1</sup>. Clearly, there is a large gap between the performances of GZSL using the default

---

<sup>1</sup>Similar trends are observed for AwA and CUB
Table 6.1: Comparison of performances measured in AUSUC between GZSL (using \textsc{Word2Vec} and \textsc{G-attr}) and multi-class classification on \textsc{ImageNet-2K}. Few-shot results are averaged over 100 rounds. GZSL with \textsc{G-attr} improves upon GZSL with \textsc{Word2Vec} significantly and quickly approaches multi-class classification performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>Flat hit@K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>\textsc{Word2Vec}</td>
<td>0.04</td>
</tr>
<tr>
<td>G-attr from 1 image</td>
<td>0.08±0.003</td>
</tr>
<tr>
<td>G-attr from 10 images</td>
<td>0.20±0.002</td>
</tr>
<tr>
<td>G-attr from 100 images</td>
<td>0.25±0.001</td>
</tr>
<tr>
<td>G-attr from all images</td>
<td>0.25</td>
</tr>
<tr>
<td>Multi-class classification</td>
<td>0.35</td>
</tr>
</tbody>
</table>

\textsc{Word2Vec} semantic representations and the ideal performance indicated by the multi-class classifiers. Note that the cross marks indicate the results of direct stacking. The multi-class classifiers not only dominate GZSL in the whole ranges (thus, with high AUSUCs) but also are capable of learning classifiers that are well-balanced (such that direct stacking works well).

**How much can idealized semantic representations help?** We hypothesize that a large portion of the gap between GZSL and multi-class classification can be attributed to the weak semantic representations used by the GZSL approach.

We investigate this by using a form of idealized semantic representations. As the success of zero-shot learning relies heavily on how accurate semantic representations represent visual similarity among classes, we examine the idea of \textit{visual features as semantic representations}. Concretely, for each class, semantic representations can be obtained by averaging visual features of images belonging to that class. We call them \textsc{G-attr} as we derive the visual features from GoogLeNet. Note that, for unseen classes, we only use the reserved training examples to derive the semantic representations; we do not use their labels to train classifiers.

Fig. 6.2 shows the performance of GZSL using \textsc{G-attr}—the gaps to the multi-class classification performances are significantly reduced from those made by GZSL using \textsc{Word2Vec}. In some cases, GZSL can almost match the performance of multi-class classifiers without using any labels from the unseen classes!

**How much labeled data to improve GZSL’s performance?** Imagine we are given a budget to label data from unseen classes, how much those labels can improve GZSL’s performance?

Table 6.1 contrasts the AUSUCs obtained by GZSL to those from multi-class classification on \textsc{ImageNet-2K}, where GZSL is allowed to use visual features as embeddings—those features can be computed from a few labeled images from the unseen classes, a scenario we can refer to as “few-shot” learning. Using about (randomly sampled) 100 labeled images per class, GZSL can quickly approach the performance of multi-class classifiers, which use about 1,000 labeled images per class. Moreover, those \textsc{G-attr} visual features as semantic representations improve upon \textsc{Word2Vec} more significantly under Flat hit@K = 1 than when K > 1.
Table 6.2: Comparison of performances measured in AUSUC between GZSL with \textsc{word2vec} and GZSL with \textsc{G-attr} on the full \textbf{ImageNet} with over 20,000 unseen classes. Few-shot results are averaged over 20 rounds.

<table>
<thead>
<tr>
<th>Method</th>
<th>Flat hit@K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>\textsc{word2vec}</td>
<td>0.006</td>
</tr>
<tr>
<td>G-attr from 1 image</td>
<td>0.018±0.0002</td>
</tr>
<tr>
<td>G-attr from 10 images</td>
<td>0.050±0.0002</td>
</tr>
<tr>
<td>G-attr from 100 images</td>
<td>0.065±0.0001</td>
</tr>
<tr>
<td>G-attr from all images</td>
<td>0.067</td>
</tr>
</tbody>
</table>

We further examine on the whole \textbf{ImageNet} with 20,345 unseen classes in Table 6.2, where we keep 80\% of the unseen classes’ examples to derive \textsc{G-attr} and test on the rest, and observe similar trends. Specifically on Flat hit@1, the performance of G-attr from merely 1 image is boosted \textbf{threefold} of that by \textsc{word2vec}, while G-attr from 100 images achieves over tenfold.

### 6.3 Summary

The studies show a large gap between the GZSL approaches and multi-class classification (by conventional supervised learning)—the latter achieves three times better AUSUC than the former. Under the hit@5 metric, the multi-class classifier achieves 0.66 AUSUC while the GZSL approach by SynC achieves only 0.17 (the maximum is 1.0).

We hypothesize that the gap is largely attributed to the weak class semantic representations. As the success of ZSL relies heavily on how accurate semantic representations represent visual similarity among classes, we examine the idea of visual features as semantic representations.

The performance is encouraging—by treating the average visual features (of both seen and unseen classes) over only a few examples as the semantic representations, the gap can already be significantly reduced even by using the same zero-shot learning algorithm.

Such a way to build up semantic representations, however, is not realistic for zero-shot learning—we should not observe any labeled example of unseen classes. In the next chapter, we develop an algorithm to improve semantic representations without seeing examples of unseen classes, according to the insights of this chapter.
Chapter 7

Improving Semantic Representations by Predicting Visual Exemplars (EXEM)

The insights from the previous chapter suggest that designing high quality semantic representations or improving the existing ones should be the focus of zero-shot learning researches. To this end, we propose to learn a mapping from the original semantic representations to the average visual features (called visual exemplars), using the seen classes’ data—visual exemplars are exactly the ones used in the previous chapter to replace the original representations. The resulting mapping is then used to obtain improved representations, which can be either plugged into any ZSL approaches or treated as the (single) training instances for unseen classes so that supervised algorithms like nearest neighbors can be applied. Fig. 7.1 shows the conceptual diagram.

Figure 7.1: Given the semantic information and visual features of the seen classes, our method learns a kernel-based regressor $\psi(\cdot)$ such that the semantic representation $a_c$ of class $c$ can predict well its class exemplar (center) $v_c$ that characterizes the clustering structure. The learned $\psi(\cdot)$ can be used to predict the visual feature vectors of the unseen classes for nearest-neighbor (NN) classification, or to improve the semantic representations for existing ZSL approaches.

7.1 Approach

Our approach is based on the structural constraint that takes advantage of the clustering structure assumption in the semantic embedding space. The constraint forces the semantic representations to be predictive of their visual exemplars (i.e., cluster centers). In this section, we describe how
we achieve this goal. First, we describe how we learn a function to predict the visual exemplars from the semantic representations. Second, given a novel semantic representation, we describe how we apply this function to perform zero-shot learning.

### 7.1.1 Learning to predict the visual exemplars from the semantic representations

For each class \( c \), we would like to find a transformation function \( \psi(\cdot) \) such that \( \psi(a_c) \approx v_c \), where \( v_c \in \mathbb{R}^d \) is the visual exemplar for the class. In this chapter, we create the visual exemplar of a class by averaging the PCA projections of data belonging to that class. That is, we consider \( v_c = \frac{1}{|I_c|} \sum_{n \in I_c} M x_n \), where \( I_c = \{i : y_i = c\} \) and \( M \in \mathbb{R}^{d \times D} \) is the PCA projection matrix computed over training data of the seen classes. We note that \( M \) is fixed for all data points (i.e., not class-specific) and is used in Eq. (7.1).

Given training visual exemplars and semantic representations, we learn \( d \) support vector regressors (SVR) with the RBF kernel—each of them predicts each dimension of visual exemplars from their corresponding semantic representations. Specifically, for each dimension \( d = 1, \ldots, d \), we use the \( \nu \)-SVR formulation [164]. Details are in Section 7.2.

Note that the PCA step is introduced for both the computational and statistical benefits. In addition to reducing dimensionality for faster computation, PCA decorrelates the dimensions of visual features such that we can predict these dimensions independently rather than jointly.

See Section 7.3 for analysis on applying SVR and PCA.

### 7.1.2 Zero-shot learning based on the predicted visual exemplars

Now that we learn the transformation function \( \psi(\cdot) \), how do we use it to perform zero-shot classification? We first apply \( \psi(\cdot) \) to all semantic representations \( a_u \) of the unseen classes. We consider two main approaches that depend on how we interpret these predicted exemplars \( \psi(a_u) \).

#### 7.1.2.1 Predicted exemplars as training data

An obvious approach is to use \( \psi(a_u) \) as data directly. Since there is only one data point per class, a natural choice is to use a nearest neighbor classifier. Then, the classifier outputs the label of the closest exemplar for each novel data point \( x \) that we would like to classify:

\[
\hat{y} = \arg\min_u \text{dis}_{NN}(M x, \psi(a_u)),
\]

where we adopt the (standardized) Euclidean distance as \( \text{dis}_{NN} \) in the experiments.

#### 7.1.2.2 Predicted exemplars as the ideal semantic representations

The other approach is to use \( \psi(a_u) \) as the ideal semantic representations (“ideal” in the sense that they have knowledge about visual features) and plug them into any existing zero-shot learning framework. We provide two examples.

In the method of convex combination of semantic embeddings (ConSE) [135], their original semantic embeddings are replaced with the corresponding predicted exemplars, while the combining coefficients remain the same. In the method of synthesized classifiers (SynC), the predicted exemplars are used to define the similarity values between the unseen classes and the
bases, which in turn are used to compute the combination weights for constructing classifiers. In particular, their similarity measure is of the form \( \exp\left\{-\text{dis}(\mathbf{a}_c, \mathbf{b}_r)\right\} \sum_{r=1}^{R} \exp\left\{-\text{dis}(\mathbf{a}_c, \mathbf{b}_r)\right\} \), where \( \text{dis} \) is the (scaled) Euclidean distance and \( \mathbf{b}_r \)'s are the semantic representations of the base classes. In this case, we simply need to change this similarity measure to \( \exp\left\{-\text{dis}(\psi(\mathbf{a}_c), \psi(\mathbf{b}_r))\right\} \sum_{r=1}^{R} \exp\left\{-\text{dis}(\psi(\mathbf{a}_c), \psi(\mathbf{b}_r))\right\} \).

### 7.1.3 Comparison to related approaches

One appealing property of our approach is its scalability: we learn and predict at the exemplar (class) level so the runtime and memory footprint of our approach depend only on the number of seen classes rather the number of training data points. This is much more efficient than other ZSL algorithms that learn at the level of each individual training instance [44, 105, 138, 5, 212, 50, 172, 135, 82, 127, 6, 156, 220, 221, 123].

Several methods propose to learn visual exemplars\(^1\) by preserving structures obtained in the semantic space [189, 119]. However, our approach \( \text{predicts} \) them with a regressor such that they may or may not strictly follow the structure in the semantic space, and thus they are more flexible and could even better reflect similarities between classes in the visual feature space.

Similar in spirit to our work, [129] proposes using nearest class mean classifiers for ZSL. The Mahalanobis metric learning in this work could be thought of as learning a linear transformation of semantic representations (their “zero-shot prior” means, which are in the visual feature space). Our approach learns a highly non-linear transformation. Moreover, our \text{EXEM (1NNs)}\(^{\text{cf. Section 7.3}}\) learns a (simpler, i.e., diagonal) metric over the learned exemplars. Finally, the main focus of [129] is on \textit{incremental}, not zero-shot, learning settings (see also [152, 147]).

[218] proposes to use a deep feature space as the semantic embedding space for ZSL. Though similar to ours, they do not compute average of visual features (exemplars) but train neural networks to predict \textit{all} visual features from their semantic representations. Their model learning takes significantly longer time than ours. Neural networks are more prone to overfitting and give inferior results (cf. Section 7.3). Additionally, we provide empirical studies on much larger-scale datasets for zero-shot learning, and analyze the effect of PCA.

### 7.2 Other details

**SVR formulation for predicting visual exemplars** Given semantic representation-visual exemplar pairs of the seen classes, we learn \(d\) support vector regressors (SVR) with RBF kernel.

\(^1\text{Exemplars are used loosely here and do not necessarily mean class-specific feature averages.}\)
Specifically, for each dimension \( d = 1, \ldots, d \) of \( v_c \), SVR is learned based on the \( \nu \)-SVR formulation [164]:

\[
\min_{w, \xi, \xi', \epsilon} \frac{1}{2} w^\top w + \lambda (\nu \epsilon + \frac{1}{S} \sum_{c=1}^{S} (\xi_c + \xi'_c)) \\
\text{s.t.} w^\top \theta^{rbf}(a_c) - v_c \leq \epsilon + \xi_c \\
v_c - w^\top \theta^{rbf}(a_c) \leq \epsilon + \xi'_c \\
\xi_c \geq 0, \xi'_c \geq 0,
\]

where \( \theta^{rbf} \) is an implicit nonlinear mapping based on our kernel. We have dropped the subscript \( d \) for aesthetic reasons but readers are reminded that each regressor is trained independently with its own target values (i.e., \( v_{cd} \)) and parameters (i.e., \( w_d \)). We found that the regression error is not sensitive to \( \lambda \) and set it to 1 in all experiments. We jointly tune \( \nu \in (0,1] \) and the kernel bandwidth and finally apply the same set of hyper-parameters for all the \( d \) regressors. Details on hyper-parameter tuning can be found in Section 7.3. The resulting \( \psi(\cdot) = [w_1^\top \theta^{rbf}(\cdot), \cdots, w_d^\top \theta^{rbf}(\cdot)]^\top \), where \( w_d \) is from the \( d \)-th regressor.

## 7.3 Empirical studies

We conduct extensive empirical studies of our approach EXEM for both the conventional and generalized settings on four benchmark datasets—Animal with Attributes (AwA) [105], CUB-200-2011 Birds (CUB) [187], SUN Attribute (SUN) [142], and the full ImageNet Fall 2011 dataset [38] with more than 20,000 unseen classes. Despite its simplicity, our approach outperforms other existing ZSL approaches in most cases, demonstrating the potential of improving semantic representations towards visual exemplars.

### 7.3.1 Setup

**Datasets, features, and semantic representations** Please be refer to Section 4.5 for details. We use the GoogLeNet deep features. For ImageNet, we further derive 21,632 dimensional semantic vectors of the class names using multidimensional scaling (MDS) on the WordNet hierarchy, as in [123]. We normalize the class semantic representations to have unit \( \ell_2 \) norms.

### 7.3.2 Implementation details

**Variants of our ZSL models given predicted exemplars** The main step of our method is to predict visual exemplars that are well-informed about visual features. How we proceed to perform zero-shot classification (i.e., classifying test data into the label space of unseen classes) based on such exemplars is entirely up to us. In this chapter, we consider the following zero-shot classification procedures that take advantage of the predicted exemplars:

- **EXEM (ZSL method):** ZSL method with predicted exemplars as semantic representations, where \( \text{ZSL method} = \text{ConSE} [135], \text{LatEm} [199], \text{and SynC}. \)

- **EXEM (1NN):** 1-nearest neighbor classifier with the Euclidean distance to the exemplars.
• **EXEM (1NNs):** 1-nearest neighbor classifier with the *standardized* Euclidean distance to the exemplars, where the standard deviation is obtained by averaging the intra-class standard deviations of all seen classes.

**EXEM (ZSL method)** regards the predicted exemplars as the ideal semantic representations (Section 7.1.2.2). On the other hand, **EXEM (1NN)** treats predicted exemplars as data prototypes (Section 7.1.2.1). The standardized Euclidean distance in **EXEM (1NNs)** is introduced as a way to scale the variance of different dimensions of visual features. In other words, it helps reduce the effect of *collapsing* data that is caused by our usage of the average of each class’ data as cluster centers.

**Hyper-parameter tuning** There are several hyper-parameters to be tuned in our experiments: (a) projected dimensionality $d$ for PCA and (b) $\lambda$, $\nu$, and the RBF-kernel bandwidth in SVR. For (a), we found that the ZSL performance is not sensitive to $d$ and thus set $d = 500$ for all experiments. For (b), we perform *class-wise* cross-validation (CV), with one exception—We found $\lambda = 1$ works robustly on all datasets for zero-shot learning.

The *class-wise* CV can be done as follows. We hold out data from a subset of seen classes as pseudo-unseen classes, train our models on the remaining folds (which belong to the remaining classes), and tune hyper-parameters based on a certain performance metric on the held-out fold. This scenario simulates the ZSL setting and has been shown to outperform the conventional CV in which each fold contains a portion of training examples from all classes.

We consider the following two performance metrics. The first one minimizes the distance between the predicted exemplars and the ground-truth (average of PCA-projected validation data of each class) in $\mathbb{R}^d$. We use the Euclidean distance in this case. We term this measure CV-*distance*. This approach does not assume the downstream task at training and aims to measure the quality of predicted exemplars by its *faithfulness*.

The other approach maximizes the zero-shot classification accuracy on the validation set. This measure can easily be obtained for **EXEM (1NN)** and **EXEM (1NNs)**, which use simple decision rules that have no further hyper-parameters to tune. Empirically, we found that CV-*accuracy* generally leads to slightly better performance. The results reported in the following for these two approaches are thus based on this measure.

On the other hand, **EXEM (SYNC$^{\text{O-VS-O}}$)**, **EXEM (SYNC$^{\text{STRUCT}}$)**, **EXEM (CONSE)**, and **EXEM (LATEM)** require further hyper-parameter tuning. For computational purposes, we use CV-*distance* for tuning hyper-parameters of the regressors, followed by the hyper-parameter tuning for SYNC and CONSE using the predicted exemplars. Since SYNC and CONSE construct their classifiers based on the distance values between class semantic representations, we do not expect a significant performance drop in this case. (We remind the reader that, in **EXEM (SYNC$^{\text{O-VS-O}}$)**, **EXEM (SYNC$^{\text{STRUCT}}$)**, **EXEM (CONSE)**, and **EXEM (LATEM)**, the predicted exemplars are used as semantic representations.)

### 7.3.3 Predicted visual exemplars

We first show that predicted visual exemplars better reflect visual similarities between classes than semantic representations. Let $D_{a_u}$ be the pairwise Euclidean distance matrix between *unseen* classes computed from semantic representations (i.e., $U$ by $U$), $D_{\psi(a_u)}$ the distance matrix
Table 7.1: We compute the Euclidean distance matrix between the unseen classes based on semantic representations ($D_{a_u}$), predicted exemplars ($D_{\psi(a_u)}$), and real exemplars ($D_{v_u}$). Our method leads to $D_{\psi(a_u)}$ that is better correlated with $D_{v_u}$ than $D_{a_u}$ is. See text for more details.

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>Correlation to $D_{v_u}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Semantic distances</td>
</tr>
<tr>
<td>AwA</td>
<td>0.862</td>
</tr>
<tr>
<td>CUB</td>
<td>0.777 ± 0.021</td>
</tr>
<tr>
<td>SUN</td>
<td>0.784 ± 0.022</td>
</tr>
</tbody>
</table>

Figure 7.2: t-SNE [184] visualization of randomly selected real images (crosses) and predicted visual exemplars (circles) for the unseen classes on (from left to right) AwA, CUB, SUN, and ImageNet. Different colors of symbols denote different unseen classes. Perfect predictions of visual features would result in well-aligned crosses and circles of the same color. Plots for CUB and SUN are based on their first splits. Plots for ImageNet are based on randomly selected 48 unseen classes from 2-hop and word vectors as semantic representations. Best viewed in color.
computed from predicted exemplars, and \( D_{u} \) the distance matrix computed from real exemplars (which we do not have access to). Table 7.1 shows that the correlation between \( D_{\psi(u)} \) and \( D_{u} \) is much higher than that between \( D_{a} \) and \( D_{u} \). Importantly, we improve this correlation without access to any data of the unseen classes.

We then show some t-SNE [184] visualization of predicted visual exemplars of the unseen classes. Ideally, we would like them to be as close to their corresponding real images as possible. In Fig. 7.2, we demonstrate that this is indeed the case for many of the unseen classes; for those unseen classes (each of which denoted by a color), their real images (crosses) and our predicted visual exemplars (circles) are well-aligned.

The quality of predicted exemplars (in this case based on the distance to the real images) depends on two main factors: the predictive capability of semantic representations and the number of semantic representation-visual exemplar pairs available for training, which in this case is equal to the number of seen classes \( S \). On AwA where we have only 40 training pairs, the predicted exemplars are surprisingly accurate, mostly either placed in their corresponding clusters or at least closer to their clusters than predicted exemplars of the other unseen classes. Thus, we expect them to be useful for discriminating among the unseen classes. On ImageNet, the predicted exemplars are not as accurate as we would have hoped, but this is expected since the word vectors are purely learned from text.

We also observe relatively well-separated clusters in the semantic representation space (in our case, also the visual feature space since we only apply PCA projections to the visual features), confirming our assumption about the existence of clustering structures. On CUB, we observe that these clusters are more mixed than on other datasets. This is not surprising given that it is a fine-grained classification dataset of bird species.

### 7.3.4 Results on the conventional setting

#### 7.3.4.1 Main results

Table 7.2 summarizes our results in the form of multi-way classification accuracies on all datasets. We significantly outperform recent state-of-the-art baselines when using GoogLeNet features.

We note that, on AwA, several recent methods obtain higher accuracies due to using a more optimistic evaluation metric (per-sample accuracy) and new types of deep features [221, 220]. This has been shown to be unsuccessfully replicated (cf. Table 2 in [198]).

Our alternative approach of treating predicted visual exemplars as the ideal semantic representations significantly outperforms taking semantic representations as given. EXEM (SYNC), EXEM (CONSE), EXEM (LATEM) outperform their corresponding base ZSL methods relatively by 5.9-6.8%, 11.4-27.6%, and 1.1-17.1%, respectively. This again suggests improved quality of semantic representations (on the predicted exemplar space).

Furthermore, we find that there is no clear winner between using predicted exemplars as ideal semantic representations or as data prototypes. The former seems to perform better on datasets with fewer seen classes. Nonetheless, we remind that using 1-nearest-neighbor classifiers clearly scales much better than zero-shot learning methods; EXEM (1NN) and EXEM (1NNS) are more efficient than EXEM (SYNC), EXEM (CONSE), and EXEM (LATEM).

Finally, we find that in general using the standardized Euclidean distance instead of the Euclidean distance for nearest neighbor classifiers helps improve the accuracy, especially on CUB, suggesting there is a certain effect of collapsing actual data during training. The only exception
Table 7.2: Comparison between existing ZSL approaches in multi-way classification accuracies (in %) on four benchmark datasets. For each dataset, we mark the best in red and the second best in blue. *Italic* numbers denote per-sample accuracy instead of per-class accuracy. On **ImageNet**, we report results for both types of semantic representations: Word vectors (wv) and MDS embeddings derived from WordNet (hie). All the results are based on GoogLeNet features [175].

<table>
<thead>
<tr>
<th>Approach</th>
<th>AwA</th>
<th>CUB</th>
<th>SUN</th>
<th>ImageNet</th>
<th>wv</th>
<th>hie</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CONSE</strong> † [135]</td>
<td>63.3</td>
<td>36.2</td>
<td>51.9</td>
<td>1.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>BIDILEL</strong> [189]</td>
<td>72.4</td>
<td>49.7 §</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>LATEM</strong> ‡ [199]</td>
<td>72.1</td>
<td>48.0</td>
<td>64.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CCA [123]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.8</td>
</tr>
<tr>
<td>SYNC\textsuperscript{o-v}s-o</td>
<td>69.7</td>
<td>53.4</td>
<td>62.8</td>
<td>1.4</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>SYNC\textsuperscript{struct}</td>
<td>72.9</td>
<td>54.5</td>
<td>62.7</td>
<td>1.5</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td><strong>EXEM</strong> (CONSE)</td>
<td>70.5</td>
<td>46.2</td>
<td>60.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>EXEM</strong> (LATEM) ‡</td>
<td>72.9</td>
<td>56.2</td>
<td>67.4</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td><strong>EXEM</strong> (SYNC\textsuperscript{o-v}s-o)</td>
<td>73.8</td>
<td>56.2</td>
<td>66.5</td>
<td>1.6</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td><strong>EXEM</strong> (SYNC\textsuperscript{struct})</td>
<td><strong>77.2</strong></td>
<td><strong>59.8</strong></td>
<td>66.1</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td><strong>EXEM</strong> (1NN)</td>
<td>76.2</td>
<td>56.3</td>
<td><strong>69.6</strong></td>
<td>1.7</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td><strong>EXEM</strong> (1NNs)</td>
<td><strong>76.5</strong></td>
<td><strong>58.5</strong></td>
<td><strong>67.3</strong></td>
<td>1.8</td>
<td>2.0</td>
<td></td>
</tr>
</tbody>
</table>

§: on a particular split of seen/unseen classes. †: our implementation. ‡: based on the code of [199], averaged over 5 different initializations.

is on **SUN**. We suspect that the standard deviation values computed on the seen classes on this dataset may not be robust enough as each class has only 20 images.

### 7.3.4.2 Large-scale zero-shot classification results

We then provide expanded results for **ImageNet**, following evaluation protocols in the literature. In Table 7.3 and 7.4, we provide results based on the exemplars predicted by word vectors and MDS features derived from WordNet, respectively. We consider SYNC\textsuperscript{o-v}s-o, rather than SYNC\textsuperscript{struct}, as the former shows better performance on **ImageNet**. Regardless of the types of metrics used, our approach outperforms the baselines significantly when using word vectors as semantic representations. For example, on 2-hop, we are able to improve the F@1 accuracy by 2% over the state-of-the-art. However, we note that this improvement is not as significant when using MDS-WordNet features as semantic representations.

We observe that the 1-nearest-neighbor classifiers perform better than using predicted exemplars as more powerful semantic representations. We suspect that, when the number of classes is very high, zero-shot learning methods (CONSE or SYNC) do not fully take advantage of the meaning provided by each dimension of the exemplars.
Table 7.3: Comparison between existing ZSL approaches on **ImageNet** using **word vectors** of the class names as semantic representations. For both metrics (in %), the higher the better. The best is in red. The numbers of unseen classes are listed in parentheses. †: our implementation.

<table>
<thead>
<tr>
<th>Test data</th>
<th>Approach</th>
<th>Flat Hit@K</th>
<th>Hierarchical precision@K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>K= 1 2 5 10 20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSE† [135]</td>
<td>11.8 18.9 31.8 43.2 54.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC0-RO □ [135]</td>
<td>11.7 18.3 30.9 42.7 54.8</td>
<td></td>
</tr>
<tr>
<td>2-hop (1,509)</td>
<td>EXEM (SYNC0-RO □)</td>
<td>12.5 19.5 32.3 43.7 55.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXEM (1NN)</td>
<td>21.5 23.8 27.5 31.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXEM (1NNs)</td>
<td>25.1 27.7 30.3 32.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSE† [135]</td>
<td>2.9 4.9 9.2 14.2 20.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC0-RO □ [135]</td>
<td>2.6 4.1 7.3 11.1 16.4</td>
<td></td>
</tr>
<tr>
<td>3-hop (7,678)</td>
<td>EXEM (SYNC0-RO □)</td>
<td>3.4 5.6 10.3 15.7 22.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXEM (1NN)</td>
<td>7.4 23.7 26.4 28.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXEM (1NNs)</td>
<td>26.9 29.1 31.1 32.0</td>
<td></td>
</tr>
<tr>
<td>All (20,345)</td>
<td>CONSE† [135]</td>
<td>1.3 2.1 3.8 5.8 8.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC0-RO □ [135]</td>
<td>1.4 2.4 4.5 7.1 10.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXEM (SYNC0-RO □)</td>
<td>1.6 2.7 5.0 7.8 11.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXEM (1NN)</td>
<td>3.2 9.2 10.7 12.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXEM (1NNs)</td>
<td>3.1 9.0 10.9 12.5</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.4: Comparison between existing ZSL approaches on **ImageNet** (with 20,842 unseen classes) using **MDS embeddings derived from WordNet** [123] as semantic representations. The higher, the better (in %). The best is in red.

<table>
<thead>
<tr>
<th>Test data</th>
<th>Approach</th>
<th>Flat Hit@K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>K= 1 2 5 10 20</td>
</tr>
<tr>
<td>All (20,842)</td>
<td>CCA [123]</td>
<td>1.8 3.0 5.2 7.3 9.7</td>
</tr>
<tr>
<td></td>
<td>SYNC0-RO □</td>
<td>2.0 3.4 6.0 8.8 12.5</td>
</tr>
<tr>
<td></td>
<td>EXEM (SYNC0-RO □)</td>
<td>2.0 3.3 6.1 9.0 12.9</td>
</tr>
<tr>
<td></td>
<td>EXEM (1NN)</td>
<td>2.0 3.4 6.3 9.2 13.1</td>
</tr>
<tr>
<td></td>
<td>EXEM (1NNs)</td>
<td>2.0 3.4 6.2 9.2 13.2</td>
</tr>
</tbody>
</table>
Table 7.5: Accuracy of EXEM (1NN) on AwA, CUB, and SUN when predicted exemplars are from original visual features (No PCA) and PCA-projected features (PCA with $d = 1024$ and $d = 500$).

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>No PCA $d = 1024$</th>
<th>PCA $d = 1024$</th>
<th>PCA $d = 500$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AwA</td>
<td>77.8</td>
<td>76.2</td>
<td>76.2</td>
</tr>
<tr>
<td>CUB</td>
<td>55.1</td>
<td>56.3</td>
<td>56.3</td>
</tr>
<tr>
<td>SUN</td>
<td>69.2</td>
<td>69.6</td>
<td>69.6</td>
</tr>
</tbody>
</table>

Table 7.6: Comparison between EXEM (1NN) with support vector regressors (SVR) and with 2-layer multi-layer perceptron (MLP) for predicting visual exemplars. Results on CUB are for the first split. Each number for MLP is an average over 3 random initialization.

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>How to predict exemplars</th>
<th>No PCA $d = 1024$</th>
<th>PCA $d = 1024$</th>
<th>PCA $d = 500$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AwA</td>
<td>SVR</td>
<td>77.8</td>
<td>76.2</td>
<td>76.2</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>76.1 ± 0.5</td>
<td>76.4 ± 0.1</td>
<td>75.5 ± 1.7</td>
</tr>
<tr>
<td>CUB</td>
<td>SVR</td>
<td>57.1</td>
<td>59.4</td>
<td>59.4</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>53.8 ± 0.3</td>
<td>54.2 ± 0.3</td>
<td>53.8 ± 0.5</td>
</tr>
</tbody>
</table>

### 7.3.4.3 Analysis

**PCA or not?** Table 7.5 investigates the effect of PCA. In general, EXEM (1NN) performs comparably with and without PCA. Moreover, decreasing PCA projected dimension $d$ from 1024 to 500 does not hurt the performance. Clearly, a smaller PCA dimension leads to faster computation due to fewer regressors to be trained.

**Kernel regression vs. Multi-layer perceptron** We compare two approaches for predicting visual exemplars: kernel-based support vector regressors (SVR) and 2-layer multi-layer perceptron (MLP) with ReLU nonlinearity. MLP weights are $\ell_2$ regularized, and we cross-validate the regularization constant.

Table 7.6 shows that SVR performs more robustly than MLP. One explanation is that MLP is prone to overfitting due to the small training set size (the number of seen classes) as well as the model selection challenge imposed by ZSL scenarios. SVR also comes with other benefits; it is more efficient and less susceptible to initialization.

### 7.3.5 Results on the generalized setting

We evaluate our methods and baselines using the Area Under Seen-Unseen accuracy Curve (AUSUC) and report the results in Table 7.7. Following the same evaluation procedure as before, our approach again outperforms the baselines on all datasets.

Recently, Xian et al. [198] proposes to unify the evaluation protocol in terms of image features, class semantic representations, data splits, and evaluation criteria for conventional and generalized zero-shot learning. In their protocol, GZSL is evaluated by the harmonic mean of
Table 7.7: Generalized ZSL results in Area Under Seen-Unseen accuracy Curve (AUSUC) on AwA, CUB, and SUN. For each dataset, we mark the best in red and the second best in blue. All approaches use GoogLeNet as the visual features and calibrated stacking to combine the scores for seen and unseen classes.

<table>
<thead>
<tr>
<th>Approach</th>
<th>AwA</th>
<th>CUB</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAP† [106]</td>
<td>0.366</td>
<td>0.194</td>
<td>0.096</td>
</tr>
<tr>
<td>IAP† [106]</td>
<td>0.394</td>
<td>0.199</td>
<td>0.145</td>
</tr>
<tr>
<td>CONSE† [135]</td>
<td>0.428</td>
<td>0.212</td>
<td>0.200</td>
</tr>
<tr>
<td>ESZSL† [156]</td>
<td>0.449</td>
<td>0.243</td>
<td>0.026</td>
</tr>
<tr>
<td>SYNCO-vs-o†</td>
<td>0.568</td>
<td>0.336</td>
<td>0.242</td>
</tr>
<tr>
<td>SYNCESTRUCT†</td>
<td>0.583</td>
<td>0.356</td>
<td>0.260</td>
</tr>
<tr>
<td>EXEM (SYNCO-vs-o)</td>
<td>0.553</td>
<td>0.365</td>
<td>0.265</td>
</tr>
<tr>
<td>EXEM (SYNCESTRUCT)</td>
<td>0.587</td>
<td>0.397</td>
<td>0.288</td>
</tr>
<tr>
<td>EXEM (1NN)</td>
<td>0.570</td>
<td>0.318</td>
<td>0.284</td>
</tr>
<tr>
<td>EXEM (1NNs)</td>
<td><strong>0.584</strong></td>
<td><strong>0.373</strong></td>
<td><strong>0.287</strong></td>
</tr>
</tbody>
</table>

†: our implementation.

seen and unseen classes’ accuracies. Technically, AUSUC provides a more complete picture of zero-shot learning method’s performance, but it is less simpler than the harmonic mean.

### 7.4 Summary

We developed a novel approach by learning a mapping from the original semantic representations to the average visual features, using the seen classes’ data. The resulting mapping is then used to obtain improved representations, which can be either plugged into any ZSL approaches or treated as the (single) training instances for unseen classes so that supervised algorithms like nearest neighbors can be applied. While extremely simple, the latter way leads to promising results, even outperforming SynC on the large-scale zero-shot learning task.
Part III

Domain Generalization for Visual Question Answering
Chapter 8

Introduction to Visual Question Answering and Its Challenges

So far we have talked about how to recognize unseen objects and differentiate them from seen ones with the help of external class semantic representations, under the learning paradigm called zero-shot learning (cf. Part II). To focus on object recognition there, we made an assumption—the question to an intelligent system is always “What is the animal (or object, scene, etc.)?”—so that we can ignore the information from questions. In this part, we will reconsider questions and focus on visual question answering (Visual QA), a much difficult task than object recognition.

Specifically, given an image, the system needs to understand the questions and then comes up with the answers, rather than just outputting all the object names within that image. (See Fig. 8.1 for an illustration.) The questions can begin with words other than “what”. The corresponding answers thus may go beyond object names to further include counts (e.g., three), time (e.g., at night), or even relationships among objects (e.g., to the left of the truck). In other words, Visual QA requires comprehending and reasoning with both visual and language information, which is an essential functionality for general artificial intelligence (AI).

To master Visual QA, we need not only novel learning algorithms (including model architectures) and faithful evaluation metrics, but also new data collections to provide learning signal and test environment. Moreover, as humans are known to have remarkable ability in generalizing and adapting their intelligence to new environments (i.e., in the wild), it is thus crucial to investigate whether the learned models have acquired such an ability. See Fig. 8.2 for an illustration.

Figure 8.1: The visual question answering (Visual QA) task [14]: given an image an intelligent system needs to answer questions related to the image.
In this chapter, we first review existing work on Visual QA\(^1\), and then discuss the remaining challenges. We conclude with the outline of our work to be presented in the following chapters.

### 8.1 Review of existing work on Visual QA

#### 8.1.1 Datasets

In merely the last four years, more than a dozen datasets have been released for Visual QA [89, 195, 72, 67, 88, 2]. In all the datasets, there are a collection of images (I). Most of them use use natural images from large-scale common image databases (e.g. MSCOCO [117]), while some are based on synthetic ones. Usually for each image, *multiple questions (Q)* and their corresponding “correct” answers (T), referred as *targets*, are generated. This can be achieved either by human annotators, or with an automatic procedure that uses captions or question templates and detailed image annotations. In our work, we will focus on VQA [14], Visual7W [230], Visual Genome (VG) [100], COCOQA [150], and VQA2 [67], which are among the most widely-used datasets in the literature.

---

\(^1\)Please also be referred to [195, 89] for overviews of the status quo of the Visual QA task.
Besides the pairs of questions and correct answers, VQA [14], Visual7W [230], and visual Madlibs [213] provide “negative” candidate answers (D), referred as decoys, for each pair so that the task can be evaluated in multiple-choice selection accuracy.

8.1.2 Evaluation

While ideally a Visual QA system can generate free-form answers [59], evaluating the answers is challenging and not amenable to automatic evaluation. Thus, so far a convenient paradigm is to evaluate machine systems using multiple-choice (MC) based Visual QA [14, 230, 80]. The machine is presented the correct answer, along with several decoys and the aim is to select the right one. The evaluation is then automatic: one just needs to record the accuracy of selecting the right answer. Alternatively, the open-ended (OE) setting is to select one from the top frequent answers and compare it to multiple human-annotated ones [10, 14, 18, 56, 67, 88, 122, 203, 209, 211, 227], avoiding constructing decoys that are too easy such that the performance is artificially boosted [14, 67].

8.1.3 Algorithms

As summarized in [89, 195, 72], in open-end Visual QA one popular framework of algorithms is to learn a joint image-question embedding and perform multi-way classification (for predicting top-frequency answers) on top [227, 10, 18, 56, 209, 122]. Though lacking the ability to generate novel answers beyond the training set, this framework has been shown to outperform other methods that dedicate for free-form answer generation [195, 89].

Different from this line of research, in the multiple-choice setting, algorithms are usually designed to learn a scoring function with the image, question, and a candidate answer as the input [80, 56, 168]. Even a simple multi-layer perceptron (MLP) model achieves the state of the art [80, 56, 168]. Such methods can take the advantage of answer semantics but fail to scale up inferencing along the increasing number of answer candidates.

8.1.4 Analysis

In [48], Ferraro et al. surveyed several exiting image captioning and Visual QA datasets in terms of their linguistic patterns. They proposed several metrics including perplexity, part of speech distribution, and syntactic complexity to characterize those datasets, demonstrating the existence of the reporting bias—the frequency that annotators write about actions, events, or states does not reflect the real-world frequencies. However, they do not explicitly show how such a bias affects the downstream tasks (i.e., Visual QA and captioning).

Specifically for Visual QA, there have been several work discussing the bias within a single dataset [67, 219, 80, 87]. For example, [67, 219] argue the existence of priors on answers given the question types and the correlation between the questions and answers (without images) in VQA [14]. They propose to augment the original datasets with additional IQT (i.e., image-question-target) triplets to resolve such issues. [80, 2] studies biases across datasets, and show the difficulties in transferring learned knowledge across datasets.
8.2 Challenges

While Visual QA has attracted significant attention, together with seemingly remarkable progress, there are still many challenges to resolve to ensure that we are on the right track towards AI.

- What kind of knowledge a Visual QA system actually learns—does it truly understand the multi-modal information? or it simply relies on and over-fits to the incidental statistics or correlations.

- The current experimental setup mainly focuses on training and testing within the same dataset. It is unclear how the learned system can be generalized to real environment where both the visual and language data might have distribution mismatch.

- State-of-the-art systems for different evaluation metrics are designed differently. It would be desirable to have a unified system or algorithm to simultaneously master both metrics.

8.3 Contributions and outline of Part III

In my thesis, I strive to conduct comprehensive studies to answer the above questions. Then according to the issues disclosed, we develop corresponding solutions to advance Visual QA.

Chapter 9 We started with multiple-choice Visual QA, which can be evaluated by the selection accuracy without considering the semantic ambiguity. Through careful analysis, we showed the design of negative candidate answers (i.e., decoys) has a significant impact on how and what the models learn from the existing datasets. In particular, the resulting model can ignore the visual information, the question, or both while still doing well on the task. We developed automatic procedures to remedy such design deficiencies by re-constructing decoy answers. Empirical studies show that the deficiencies have been alleviated in the remedied datasets and the performance on them is likely a more faithful indicator of the difference among learning models.

Chapter 10 We then studied cross-dataset generalization as a proxy to evaluate how the learned models can be applied to real-world environment, reminiscent of the seminal work by Torralba and Efros [181] on object recognition. We showed that the language components contain strong dataset characteristics (e.g., phrasing styles)—by looking at them alone, a machine can detect the origin of a Visual QA instance (i.e., an image-question-answer triplet). We performed so far the most comprehensive analysis to show that such characteristics significantly prevent cross-dataset generalization, evaluated among five popular datasets (see Fig. 8.3). In other words, current Visual QA models cannot effectively handle unfamiliar language in new environment.

To this end, we developed a novel domain adaptation algorithm for Visual QA so that we can properly transfer the learned knowledge (see Fig. 8.4). We introduced a framework by adapting the unfamiliar language usage (target domain) to what the learned Visual QA model has been trained on (source domain) so that we can re-use the model without re-training. Our algorithm minimizes the domain mismatch while ensuring the consistency among different modalities (i.e., images, questions, and answers), given only limited amount of data from the target domain.
Figure 8.3: We experiment knowledge transfer across five popular datasets: VQA [14], Visual7W [230], Visual Genome (VG) [100], COCOQA [150], and VQA2 [67]. We train a model on one dataset and investigate how well it can perform on the others.

Figure 8.4: We introduced a framework by adapting the unfamiliar language usage (target domain) to what the learned Visual QA model has been trained on (source domain) so that we can re-use the model without re-training.
Figure 8.5: Denote $i$ as an image, $q$ as a question, and $c$ as a candidate answer, we aim to learn a scoring function $f(i, q, c)$ so that it gives a high score if $c$ is the target answer of the $(i, q)$ pair. We factorize $f(i, q, c)$ into $h(i, q)$ and $g(c)$, in which we can take advantage of existing joint embedding of vision and language for $h(i, q)$. Moreover, $g(c)$ can effectively captures the answer semantic ignored in many state-of-the-art models. The scoring function is learned to maximize the likelihood of outputting the target answer from a set of stochastically sampled candidates.

Chapter 11  We further developed a probabilistic and factorization framework of Visual QA algorithms that can be applied to both the multiple-choice and open-ended settings. Our framework effectively leverages the answer semantics and can directly account for out-of-vocabulary instances (see Fig. 8.5), drastically increasing the transferability. More importantly, both work in Chapter 10 and 11 can be applied to existing models so that we can stand on the shoulder of their insightful architecture design in learning joint vision and language embedding.
Chapter 9

Creating Better Visual Question Answering Datasets

9.1 Overview

In this chapter, we study how to design high-quality multiple choices for the Visual QA task. In this task, the machine (or the human annotator) is presented with an image, a question and a list of candidate answers. The goal is to select the correct answer through a consistent understanding of the image, the question and each of the candidate answers. As in any multiple-choice based tests (such as GRE), designing what should be presented as negative answers—we refer them as decoys—is as important as deciding the questions to ask. We all have had the experience of exploiting the elimination strategy: This question is easy—none of the three answers could be right so the remaining one must be correct!

While a clever strategy for taking exams, such “shortcuts” prevent us from studying faithfully how different learning algorithms comprehend the meanings in images and languages (e.g., the quality of the embeddings of both images and languages in a semantic space). It has been noted that machines can achieve very high accuracies of selecting the correct answer without the visual input (i.e., the image), the question, or both [80, 14]. Clearly, the learning algorithms have over-fit on incidental statistics in the datasets. For instance, if the decoy answers have rarely been used as the correct answers (to any questions), then the machine can rule out a decoy answer with a binary classifier that determines whether the answers are in the set of the correct answers—note that this classifier does not need to examine the image and it just needs to memorize the list of the correct answers in the training dataset. See Fig. 9.1 for an example, and Section 9.3 for more and detailed analysis.

We focus on minimizing the impacts of exploiting such shortcuts. We suggest a set of principles for creating decoy answers. In light of the amount of human efforts in curating existing datasets for the Visual QA task, we propose two procedures that revise those datasets such that the decoy answers are better designed. In contrast to some earlier works, the procedures are fully automatic and do not incur additional human annotator efforts. We apply the procedures to revise both Visual7W [230] and VQA [14]. Additionally, we create new multiple-choice based datasets from COCOQA [150] and the recently released VQA2 [67] and Visual Genome datasets [100]. The one based on Visual Genome becomes the largest multiple-choice dataset for the Visual QA task, with more than one million image-question-candidate answers triplets.

We conduct extensive empirical and human studies to demonstrate the effectiveness of our procedures in creating high-quality datasets for the Visual QA task. In particular, we show that machines need to use all three information (image, questions and answers) to perform well—any
Figure 9.1: An illustration of how the shortcuts in the Visual7W dataset [230] should be remedied. In the original dataset, the correct answer “A train” is easily selected by a machine as it is far often used as the correct answer than the other decoy (negative) answers. (The numbers in the brackets are probability scores computed using eq. (9.2)). Our two procedures—QoU and IoU (cf. Section 9.4)—create alternative decoys such that both the correct answer and the decoys are highly likely by examining either the image or the question alone. In these cases, machines make mistakes unless they consider all information together. Thus, the alternative decoys suggested our procedures are better designed to gauge how well a learning algorithm can understand all information equally well.

missing information induces a large drop in performance. Furthermore, we show that humans dominate machines in the task. However, given the revised datasets are likely reflecting the true gap between the human and the machine understanding of multimodal information, we expect that advances in learning algorithms likely focus more on the task itself instead of overfitting to the idiosyncrasies in the datasets.

The rest of the chapter is organized as follows. In Section 9.2, we describe related work. In Section 9.3, we analyze and discuss the design deficiencies in existing datasets. In Section 9.4, we describe our automatic procedures for remedying those deficiencies. In Section 9.5 we conduct experiments and analysis. We conclude the chapter in Section 9.6.

9.2 Related work

In VQA [14], the decoys consist of human-generated plausible answers as well as high-frequency and random answers from the datasets. In Visual7W [230], the decoys are all human-generated plausible ones. Note that, humans generate those decoys by only looking at the questions and the correct answers but not the images. Thus, the decoys might be unrelated to the corresponding images. A learning algorithm can potentially examine the image alone and be able to identify the correct answer. In visual Madlibs [213], the questions are generated with a limited set of question
templates and the detailed annotations (e.g., objects) of the images. Thus, similarly, a learning model can examine the image alone and deduce the correct answer.

Our work is inspired by the experiments in [80] where they observe that machines without looking at images or questions can still perform well on the Visual QA task. Others have also reported similar issues [67, 219, 87, 1, 88, 2], though not in the multiple-choice setting. Our work extends theirs by providing more detailed analysis as well as automatic procedures to remedy those design deficiencies.

Besides Visual QA, VisDial [36] and Ding et al. [39] also propose automatic ways to generate decoys for the tasks of multiple-choice visual captioning and dialog, respectively.

9.3 Analysis of decoy answers’ effects

In this section, we examine in detail the dataset Visual7W [230], a popular choice for the Visual QA task. We demonstrate how the deficiencies in designing decoy questions impact the performance of learning algorithms.

In multiple-choice Visual QA datasets, a training or test example is a triplet that consists of an image $I$, a question $Q$, and a candidate answer set $A$. The set $A$ contains a target $T$ (the correct answer) and $K$ decoys (incorrect answers) denoted by $D$. An IQA triplet is thus $\{I, Q, A = \{T, D_1, \cdots, D_K\}\}$. We use $C$ to denote either the target or a decoy.

9.3.1 Visual QA models

We investigate how well a learning algorithm can perform when supplied with different modalities of information. We concentrate on the one hidden-layer MLP model proposed in [80], which has achieved state-of-the-art results on the dataset Visual7W. The model computes a scoring function $f(c, i)$

$$f(c, i) = \sigma(U \max(0, W g(c, i)) + b)$$

(9.1)

over a candidate answer $c$ and the multimodal information $i$, where $g$ is the joint feature of $(c, i)$ and $\sigma(x) = 1/(1 + \exp(-x))$. The information $i$ can be null, the image ($I$) alone, the question ($Q$) alone, or the combination of both ($I+Q$).

Given an IQA triplet, we use the penultimate layer of ResNet-200 [73] as visual features to represent $I$ and the average word2vec embeddings [131] as text features to represent $Q$ and $C$. To form the joint feature $g(c, i)$, we just concatenate the features together. The candidate $c \in A$ that has the highest $f(c, i)$ score in prediction is selected as the model output.

We use the standard training, validation, and test splits of Visual7W, where each contains 69,817, 28,020, and 42,031 examples respectively. Each question has 4 candidate answers. The parameters of $f(c, i)$ are learned by minimizing the binary logistic loss of predicting whether or not a candidate $c$ is the target of an IQA triplet. Details are in Section 9.5.
### Table 9.1: Accuracy of selecting the right answers out of 4 choices (%) on the Visual QA task on Visual7W.

<table>
<thead>
<tr>
<th>Information used</th>
<th>Machine</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td>25.0</td>
<td>25.0</td>
</tr>
<tr>
<td>A</td>
<td>52.9</td>
<td>-</td>
</tr>
<tr>
<td>I + A</td>
<td>62.4</td>
<td>75.3</td>
</tr>
<tr>
<td>Q + A</td>
<td>58.2</td>
<td>36.4</td>
</tr>
<tr>
<td>I + Q + A</td>
<td>65.7</td>
<td>88.4</td>
</tr>
</tbody>
</table>

#### 9.3.2 Analysis results

**Machines find shortcuts** Table 9.1 summarizes the performance of the learning models, together with the human studies we performed on a subset of 1,000 triplets (cf. Section 9.5 for details). There are a few interesting observations.

First, in the row of “A” where only the candidate answers (and whether they are right or wrong) are used to train a learning model, the model performs significantly better than random guessing and humans (52.9% vs. 25%)—humans will deem each of the answers equally likely without looking at both the image and the question! Note that in this case, the information $i$ in eq. (9.1) contains nothing. The model learns the specific statistics of the candidate answers in the dataset and exploits those. Adding the information about the image (i.e., the row of “I+A”), the machine improves significantly and gets close to the performance when all information is used (62.4% vs. 65.7%). There is a weaker correlation between the question and the answers as “Q+A” improves over “A” only modestly. This is expected. In the Visual7W dataset, the decoys are generated by human annotators as plausible answers to the questions without being shown the images—thus, many IQA triplets can be solved by object, attribute or concept detection on the image, without understanding the questions. This is indeed the case also for humans—humans can achieve 75.3% by considering “I+A” and not “Q”. Note that the difference between machine and human on “I+A” are likely due to their difference in understanding visual information.

Note that human improves significantly from “I+A” to “I+Q+A” with “Q” added, while the machine does so only marginally. The difference can be attributed to the difference in understanding the question and correlating with the answers between the two. Since each image corresponds to multiple questions or have multiple objects, solely relying on the image itself will not work well in principle. Such difference clearly indicates that in the Visual QA model, the language component is weak as the model cannot fully exploit the information in “Q”, making a smaller relative improvement 5.3% (from 62.4% to 65.7%) where humans improved relatively 17.4%.

**Shortcuts are due to design deficiencies** We probe deeper on how the decoy answers have impacted the performance of learning models.
As explained above, the decoys are drawn from all plausible answers to a question, irrespective of whether they are visually grounded or not. We have also discovered that the targets (i.e., correct answers) are infrequently used as decoys. Specifically, among the 69,817 training samples, there are 19,503 unique correct answers and each one of them is used about 3.6 times as correct answers to a question. However, among all the $69,817 \times 3 \approx 210K$ decoys, each correct answer appears 7.2 times on average, far below a chance level of 10.7 times ($210K \div 19,503 \approx 10.7$). This disparity exists in the test samples too. Consequently, the following rule, computing each answer’s likelihood of being correct,

$$P(\text{correct}|C) = \begin{cases} 
0.5, & \text{if C is never seen in training,} \\
\frac{\# \text{times C as target}}{\# \text{times C as target} + (\# \text{times C as decoys})/K}, & \text{otherwise},
\end{cases}$$

(9.2)

should perform well. Essentially, it measures how unbiased C is used as the target and the decoys. Indeed, it attains an accuracy of 48.73% on the test data, far better than the random guess and is close to the learning model using the answers’ information only (the “A” row in Table 9.1).

**Good rules for designing decoys** Based on our analysis, we summarize the following guidance rules to design decoys: (1) **Question only Unresolvable (QoU)**. The decoys need to be equally plausible to the question. Otherwise, machines can rely on the correlation between the question and candidate answers to tell the target from decoys, even without the images. Note that this is a principle that is being followed by most datasets. (2) **Neutrality**. The decoys answers should be equally likely used as the correct answers. (3) **Image only Unresolvable (IoU)**. The decoys need to be plausible to the image. That is, they should appear in the image, or there exist questions so that the decoys can be treated as targets to the image. Otherwise, Visual QA can be resolved by objects, attributes, or concepts detection in images, even without the questions.

Ideally, each decoy in an IQA triplet should meet the three principles. **Neutrality** is comparably easier to achieve by reusing terms in the whole set of targets as decoys. On the contrary, a decoy may hardly meet QoU and IoU simultaneously\(^1\). However, as long as all decoys of an IQA triplet meet Neutrality and some meet QoU and others meet IoU, the triplet as a whole achieves the three principles—a machine ignoring either images or questions will likely perform poorly.

### 9.4 Creating better Visual QA datasets

In this section, we describe our approaches of remedying deficiencies in the existing datasets for the Visual QA task. We introduce two automatic and widely-applicable procedures to create new decoys that can prevent learning models from exploiting incident statistics in the datasets.

#### 9.4.1 Methods

**Main ideas** Our procedures operate on a dataset that already contains image-question-target (IQT) triplets, i.e., we do not assume it has decoys already. For instance, we have used our

\(^1\)E.g., in Fig 9.1, for the question “What vehicle is pictured?”, the only answer that meets both principles is “train”, which is the correct answer instead of being a decoy.
procedures to create a multiple-choice dataset from the Visual Genome dataset which has no decoy. We assume that each image in the dataset is coupled with “multiple” QT pairs, which is the case in nearly all the existing datasets. Given an IQT triplet (I, Q, T), we create two sets of decoy answers.

- **QoU-decoys.** We search among all other triplets that have similar questions to Q. The targets of those triplets are then collected as the decoys for T. As the targets to similar questions are likely plausible for the question Q, QoU-decoys likely follow the rules of **Neutrali** and **Question only Unresolvable (QoU)**. We compute the average **WORD2VEC** [131] to represent a question, and use the cosine similarity to measure the similarity between questions.

- **IoU-decoys.** We collect the targets from other triplets of the **same** image to be the decoys for T. The resulting decoys thus definitely follow the rules of **Neutrali** and **Image only Unresolvable (IoU)**.

We then combine the triplet (I, Q, T) with QoU-decoys and IoU-decoys to form an IQA triplet as a training or test sample.

**Resolving ambiguous decoys** One potential drawback of automatically selected decoys is that they may be semantically similar, ambiguous, or rephrased terms to the target [230]. We utilize two filtering steps to alleviate it. First, we perform string matching between a decoy and the target, deleting those decoys that contain or are covered by the target (e.g., “daytime” vs. “during the daytime” and “ponytail” vs. “pony tail”).

Secondly, we utilize the WordNet hierarchy and the Wu-Palmer (WUP) score [196] to eliminate semantically similar decoys. The WUP score measures how similar two word senses are (in the range of [0, 1]), based on the depth of them in the taxonomy and that of their least common subsumer. We compute the similarity of two strings according to the WUP scores in a similar manner to [124], in which the WUP score is used to evaluate Visual QA performance. We eliminate decoys that have higher WUP-based similarity to the target. We use the NLTK toolkit [21] to compute the similarity.

**Other details** For QoU-decoys, we sort and keep for each triplet the top \( N \) (e.g., 10,000) similar triplets from the entire dataset according to the question similarity. Then for each triplet, we compute the WUP-based similarity of each potential decoy to the target successively, and accept those with similarity below 0.9 until we have \( K \) decoys. We choose 0.9 according to [124]. We also perform such a check among selected decoys to ensure they are not very similar to each other. For IoU-decoys, the potential decoys are sorted randomly. The WUP-based similarity with a threshold of 0.9 is then applied to remove ambiguous decoys.

**9.4.2 Comparison to other datasets**

Several authors have noticed the design deficiencies in the existing databases and have proposed “fixes” [14, 213, 230, 36]. No dataset has used a procedure to generate IoU-decoys. We empirically show that how the IoU-decoys significantly remedy the design deficiencies in the datasets.
Several previous efforts have generated decoys that are similar in spirit to our QoU-decoys. Yu et al. [213], Das et al. [36], and Ding et al. [39] automatically find decoys from similar questions or captions based on question templates and annotated objects, tri-grams and GLOVE embeddings [143], and paragraph vectors [108] and linguistic surface similarity, respectively. The later two are for different tasks from Visual QA, and only Ding et al. [39] consider removing semantically ambiguous decoys like ours. Antol et al. [14] and Zhu et al. [230] ask humans to create decoys, given the questions and targets. As shown earlier, such decoys may disobey the rule of Neutrality.

Goyal et al. [67] augment the VQA dataset [14] (by human efforts) with additional IQT triplets to eliminate the shortcuts (language prior) in the open-ended setting. Their effort is complementary to ours on the multiple-choice setting. Note that an extended task of Visual QA, visual dialog [36], also adopts the latter setting.

9.5 Empirical studies

9.5.1 Dataset

We examine our automatic procedures for creating decoys on five datasets. Table 9.2 summarizes the characteristics of the three datasets—VQA, Visual7W, and Visual Genome—we focus on.

**VQA Real [14]** The dataset uses images from MSCOCO [117] under the same splits for training/validation/testing to construct IQA triplets. Totally 614,163 IQA triplets are generated for 204,721 images. Each question has 18 candidate answers: in general 3 decoys are human-generated, 4 are randomly sampled, and 10 are randomly sampled frequent-occurring targets. As the test set does not indicate the targets, our studies focus on the training and validation sets.

**Visual7W Telling (Visual7W) [230]** The dataset uses 47,300 images from MSCOCO [117] and contains 139,868 IQA triplets. Each has 3 decoys generated by humans.

**Visual Genome (VG) [100]** The dataset uses 101,174 images from MSCOCO [117] and contains 1,445,322 IQT triplets. No decoys are provided. Human annotators are asked to write diverse pairs of questions and answers freely about an image or with respect to some regions of it. On average an image is coupled with 14 question-answer pairs. We divide the dataset into non-overlapping 50%/20%/30% for training/validation/testing. Additionally, we partition such that each portion is a “superset” of the corresponding one in Visual7W, respectively.

**COCOQA [150]** This dataset contains in total 117,684 auto-generated IQT triplets with no decoy answers. Therefore, we create decoys using our proposed approach and follow the original data split, leading to a training set and a testing set with 78,736 IQA triplets and 38,948 IQA triplets, respectfully.

**VQA2 [67]** VQA2 is a successive dataset of VQA, which pairs each IQT triplet with a complementary one to reduce the correlation between questions and answers. There are 443,757 training IQT triplets and 214,354 validation IQT triplets, with no decoys. We generate decoys using our
<table>
<thead>
<tr>
<th>Dataset Name</th>
<th># of Images</th>
<th># of triplets</th>
<th># of decoys per triplet</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQA</td>
<td>train 83k</td>
<td>val 41k</td>
<td>test 81k</td>
</tr>
<tr>
<td>Visual7W</td>
<td>train 14k</td>
<td>val 5k</td>
<td>test 8k</td>
</tr>
<tr>
<td>VG</td>
<td>train 49k</td>
<td>val 19k</td>
<td>test 29k</td>
</tr>
</tbody>
</table>

Table 9.2: Summary of Visual QA datasets.

We do not consider the test split as it does not indicate the targets (correct answers).

Creating decoys We create 3 QoU-decoys and 3 IoU-decoys for every IQT triplet in each dataset, following the steps in Section 9.4.1. In the cases that we cannot find 3 decoys, we include random ones from the original set of decoys for VQA and Visual7W; for other datasets, we randomly include those from the top 10 frequently-occurring targets.

9.5.2 Setup

Visual QA models We utilize the MLP models mentioned in Section 9.3 for all the experiments. We denote MLP-A, MLP-QA, MLP-IA, MLP-IQA as the models using A (Answers only), Q+A (Question plus Answers), I+A (Image plus Answers), and I+Q+A (Image, Question and Answers) for multimodal information, respectively. The hidden-layer has 8,192 neurons. We use a 200-layer ResNet [73] to compute visual features which are 2,048-dimensional. The ResNet is pre-trained on ImageNet [157]. The \textsc{Word2Vec} feature [131] for questions and answers are 300-dimensional, pre-trained on Google News. The parameters of the MLP models are learned by minimizing the binary logistic loss of predicting whether or not a candidate answer is the target of the corresponding IQA triplet.

We further experiment with a variant of the spatial memory network (denoted as Attention) [203] and the \textsc{HieCoAtt} model [122] adjusted for the multiple-choice setting. Both models utilize the attention mechanism.

Optimization We train all our models using stochastic gradient based optimization method with mini-batch size of 100, momentum of 0.9, and the stepped learning rate policy: the learning rate is divided by 10 after every $M$ mini-batches. We set the initial learning rate to be 0.01 (we further consider 0.001 for the case of fine-tuning). For each model, we train with at most 600,000 iterations. We treat $M$ and the number of iterations as hyper-parameters of training. We tune the hyper-parameters on the validation set.

Within each mini-batch, we sample 100 IQA triplets. For each triplet, we randomly choose to use QoU-decoys or IoU-decoys when training on IoU+QoU, or QoU-decoys or IoU-decoys or Orig when training on All. We then take the target and 3 decoys for each triplet to train the binary classifier (i.e., minimize the logistic loss). Specifically on VQA, which has 17 Orig decoys for a triplet, we randomly choose 3 decoys out of them. That is, 100 triplets in the mini-batch corresponds to 400 examples with binary labels. This procedure is to prevent unbalanced

\footnote{We experiment on using different features in Section 9.5.4.}
Table 9.3: Test accuracy (%) on Visual7W.

<table>
<thead>
<tr>
<th>Method</th>
<th>Orig</th>
<th>IoU</th>
<th>QoU</th>
<th>IoU +QoU</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP-A</td>
<td>52.9</td>
<td>27.0</td>
<td>34.1</td>
<td>17.7</td>
<td>15.6</td>
</tr>
<tr>
<td>MLP-IA</td>
<td>62.4</td>
<td>27.3</td>
<td>55.0</td>
<td>23.6</td>
<td>22.2</td>
</tr>
<tr>
<td>MLP-QA</td>
<td>58.2</td>
<td>84.1</td>
<td>40.7</td>
<td>37.8</td>
<td>31.9</td>
</tr>
<tr>
<td>MLP-IQA</td>
<td>65.7</td>
<td>84.1</td>
<td>57.6</td>
<td>52.0</td>
<td>45.1</td>
</tr>
<tr>
<td>HieCoAtt*</td>
<td>63.9</td>
<td>-</td>
<td>-</td>
<td>51.5</td>
<td>-</td>
</tr>
<tr>
<td>Attn*</td>
<td>65.9</td>
<td>-</td>
<td>-</td>
<td>52.8</td>
<td>-</td>
</tr>
<tr>
<td>Human</td>
<td>88.4</td>
<td>-</td>
<td>-</td>
<td>84.1</td>
<td>-</td>
</tr>
<tr>
<td>Random</td>
<td>25.0</td>
<td>25.0</td>
<td>25.0</td>
<td>14.3</td>
<td>10.0</td>
</tr>
</tbody>
</table>

*: based on our implementation or modification

9.5.3 Main results

We present the main results on VQA, Visual7W, and Visual Genome. The performances of learning models and humans on the 3 datasets are reported in Table 9.3, 9.4, and 9.5\(^3\).

\(^3\)We note that in Table 9.3, the 4.3% drop of the human performance on IoU +QoU, compared to Orig, is likely due to that IoU +QoU has more candidates (7 per question). Besides, the human performance on qaVG cannot be directly compared to that on the other datasets, since the questions on qaVG tend to focus on local image regions and are considered harder.
### Table 9.4: Accuracy (%) on the validation set in VQA.

<table>
<thead>
<tr>
<th>Method</th>
<th>Orig</th>
<th>IoU</th>
<th>QoU</th>
<th>IoU +QoU</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP-A</td>
<td>31.2</td>
<td>39.9</td>
<td>45.7</td>
<td>31.2</td>
<td>27.4</td>
</tr>
<tr>
<td>MLP-IA</td>
<td>42.0</td>
<td>39.8</td>
<td>55.1</td>
<td>34.1</td>
<td>28.7</td>
</tr>
<tr>
<td>MLP-QA</td>
<td>58.0</td>
<td>84.7</td>
<td>55.1</td>
<td>54.4</td>
<td>50.0</td>
</tr>
<tr>
<td>MLP-IQA</td>
<td>64.6</td>
<td>85.2</td>
<td>65.4</td>
<td>63.7</td>
<td>58.9</td>
</tr>
<tr>
<td>HieCoAtt*</td>
<td>63.0</td>
<td>-</td>
<td>63.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Attntion*</td>
<td>66.0</td>
<td>-</td>
<td>66.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Human</td>
<td>88.5</td>
<td>-</td>
<td>89.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Random</td>
<td>5.6</td>
<td>25.0</td>
<td>25.0</td>
<td>14.3</td>
<td>4.2</td>
</tr>
</tbody>
</table>

*: based on our implementation or modification
†: taken from [14]

**Effectiveness of new decoys** A better set of decoys will force learning models to integrate all 3 pieces of information—images, questions and answers—to make the correct selection from multiple-choices. In particular, they should prevent learning algorithms from exploiting shortcuts such that partial information is sufficient for performing well on the Visual QA task.

Table 9.3 clearly indicates that those goals have been achieved. With the Orig decoys, the relatively small gain from MLP-IA to MLP-IQA suggests that the question information can be ignored to attain good performance. However, with the IoU-decoys which require questions to help to resolve (as image itself is inadequate to resolve), the gain is substantial (from 27.3% to 84.1%). Likewise, with the QoU-decoys (question itself is not adequate to resolve), including images information improves from 40.7% (MLP-QA) substantially to 57.6% (MLP-IQA). Note that with the Orig decoys, this gain is smaller (58.2% vs. 65.7%).

It is expected that MLP-IA matches better QoU-decoys but not IoU-decoys, and MLP-QA is the other way around. Thus it is natural to combine these two decoys. What is particularly appealing is that MLP-IQA improves noticeably over models learned with partial information on the combined IoU +QoU-decoys (and “All” decoys⁴). Furthermore, using answer information only (MLP-A) attains about the chance-level accuracy.

On the VQA dataset (Table 9.4), the same observations hold, though to a lesser degree. On any of the IoU or QoU columns, we observe substantial gains when the complementary information is added to the model (such as MLP-IA to MLP-IQA). All these improvements are much more visible than those observed on the original decoy sets.

Combining both Table 9.3 and 9.4, we notice that the improvements from MLP-QA to MLP-IQA tend to be lower when facing IoU-decoys. This is also expected as it is difficult to have decoys that are simultaneously both IoU and QoU—such answers tend to be the target answers. Nonetheless, we deem this as a future direction to explore.

---

⁴We note that the decoys in Orig are not trivial, which can be seen from the gap between All and IoU +QoU. Our main concern on Orig is that for those questions that machines can accurately answer, they mostly rely on only partial information. This will thus hinder designing machines to fully comprehend and reason from multimodal information. We further experiment on random decoys, which can achieve Neutrality but not the other two principles, to demonstrate the effectiveness of our methods in Section 9.5.4.
<table>
<thead>
<tr>
<th>Method</th>
<th>IoU</th>
<th>QoU</th>
<th>IoU +QoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP-A</td>
<td>29.1</td>
<td>36.2</td>
<td>19.5</td>
</tr>
<tr>
<td>MLP-IA</td>
<td>29.5</td>
<td>60.2</td>
<td>25.2</td>
</tr>
<tr>
<td>MLP-QA</td>
<td>89.3</td>
<td>45.6</td>
<td>43.9</td>
</tr>
<tr>
<td>MLP-IQA</td>
<td>89.2</td>
<td>64.3</td>
<td>58.5</td>
</tr>
<tr>
<td>HieCoAtt∗</td>
<td>-</td>
<td>-</td>
<td>57.5</td>
</tr>
<tr>
<td>Attention∗</td>
<td>-</td>
<td>-</td>
<td>60.1</td>
</tr>
<tr>
<td>Human</td>
<td>-</td>
<td>-</td>
<td>82.5</td>
</tr>
<tr>
<td>Random</td>
<td>25.0</td>
<td>25.0</td>
<td>14.3</td>
</tr>
</tbody>
</table>

*: based on our implementation or modification

Table 9.5: Test accuracy (%) on qaVG.

**Differences across datasets**  Contrasting Visual7W to VQA (on the column IoU +QoU), we notice that Visual7W tends to have bigger improvements in general. This is due to the fact that VQA has many questions with “Yes” or “No” as the targets—the only valid decoy to the target “Yes” is “No”, and vice versa. As such decoys are already captured by Orig of VQA (“Yes” and “No” are both top frequently-occurring targets), adding other decoy answers will not make any noticeable improvement. In Section 9.5.4, however, we show that once we remove such questions/answers pairs, the degree of improvements increases substantially.

**Comparison on Visual QA models**  As presented in Table 9.3 and 9.4, MLP-IQA is on par with or even outperforms Attention and HieCoAtt on the Orig decoys, showing how the shortcuts make it difficult to compare different models. By eliminating the shortcuts (i.e., on the combined IoU +QoU-decoys), the advantage of using sophisticated models becomes obvious (Attention outperforms MLP-IQA by 3% in Table 9.4), indicating the importance to design advanced models for achieving human-level performance on Visual QA.

For completeness, we include the results on the Visual Genome dataset in Table 9.5. This dataset has no “Orig” decoys, and we have created a multiple-choice based dataset qaVG from it for the task—it has over 1 million triplets, the largest dataset on this task to our knowledge. On the combined IoU +QoU-decoys, we again clearly see that machines need to use all the information to succeed.

With qaVG, we also investigate whether it can help improve the multiple-choice performances on the other two datasets. We use the MLP-IQA trained on qaVG with both IoU and QoU decoys to initialize the models for the Visual7W and VQA datasets. We report the accuracies before and after fine-tuning, together with the best results learned solely on those two datasets. As shown in Table 9.6, fine-tuning largely improves the performance, justifying the finding by Fukui et al. [56].

**9.5.4 Additional results and analysis**

**Results on VQA w/o QA pairs that have Yes/No as the targets**  The validation set of VQA contains 45,478 QA pairs (out of totally 12,1512 pairs) that have Yes or No as the correct answers. The only reasonable decoy to Yes is No, and vice versa—any other decoy could be easily recognized in principle. Since both of them are among top 10 frequently-occurring answers, they are
Datasets | Decoys | Best w/o using qaVG | qaVG model initial | fine-tuned |
--- | --- | --- | --- | --- |
Visual7W | Orig | 65.7 | 60.5 | 69.1 |
| IoU +QoU | 52.0 | 58.1 | 58.7 |
| All | 45.1 | 48.9 | 51.0 |
VQA | Orig | 64.6 | 42.2 | 65.6 |
| IoU +QoU | 63.7 | 47.9 | 64.1 |
| All | 58.9 | 37.5 | 59.4 |

Table 9.6: Using models trained on qaVG to improve Visual7W and VQA (Accuracy in %).

<table>
<thead>
<tr>
<th>Method</th>
<th>Orig</th>
<th>IoU</th>
<th>QoU</th>
<th>IoU +QoU</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP-A</td>
<td>28.8</td>
<td>42.9</td>
<td>34.5</td>
<td>23.6</td>
<td>15.8</td>
</tr>
<tr>
<td>MLP-IA</td>
<td>43.0</td>
<td>44.8</td>
<td>53.2</td>
<td>35.5</td>
<td>28.5</td>
</tr>
<tr>
<td>MLP-QA</td>
<td>45.8</td>
<td>80.7</td>
<td>39.3</td>
<td>38.2</td>
<td>31.9</td>
</tr>
<tr>
<td>MLP-IQA</td>
<td>55.6</td>
<td>81.8</td>
<td>56.6</td>
<td>53.7</td>
<td>46.5</td>
</tr>
<tr>
<td>HieCoAtt*</td>
<td>54.8</td>
<td>-</td>
<td>-</td>
<td>55.6</td>
<td>-</td>
</tr>
<tr>
<td>Attention*</td>
<td>58.5</td>
<td>-</td>
<td>-</td>
<td>58.6</td>
<td>-</td>
</tr>
<tr>
<td>Human-IQA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>85.5</td>
<td>-</td>
</tr>
<tr>
<td>Random</td>
<td>5.6</td>
<td>25.0</td>
<td>25.0</td>
<td>14.3</td>
<td>4.2</td>
</tr>
</tbody>
</table>

*: based on our implementation or modification

Table 9.7: Accuracy (%) on VQA−2014val, which contains 76,034 triplets.

already included in the Orig decoys—our IoU-decoys and QoU-decoys can hardly make noticeable improvement. We thus remove all those pairs (denoted as Yes/No QA pairs) to investigate the improvement on the remaining pairs, for which having multiple choices makes sense. We denote the subset of VQA as VQA− (we remove Yes/No pairs in training and validation sets).

We conduct experiments on VQA−, and Table 9.7 summarizes the machines’ as well as humans’ results. Compared to Table 9.4, most of the results drop, which is expected as those removed Yes/No pairs are considered simpler and easier ones—their effective random chance is 50%. The exception is for the MLP-IA models, which performs roughly the same or even better on VQA−, suggesting that Yes/No pairs are somehow difficult to MLP-IA. This, however, makes sense since without the questions (e.g., those start with “Is there a ...” or “Does the person ...”), a machine cannot directly tell if the correct answer falls into Yes or No, or others.

We see that on VQA−, the improvement by our IoU-decoys and QoU-decoys becomes significant. The gain brought by images on QoU (from 39.3% to 56.6%) is much larger than that on Orig (from 45.8% to 55.6%). Similarly, the gain brought by questions on IoU (from 44.8% to 81.8%) is much larger than that on Orig (from 43.0% to 55.6%). After combining IoU-decoys and QoU-decoys as in IoU +QoU and All, the improvement by either including images to MLP-QA or including questions to MLP-IA is noticeable higher than that on Orig. Moreover, even with only 6 decoys, the performance by MLP-A on IoU +QoU is already lower than that on Orig, which has 17 decoys, demonstrating the effectiveness of our decoys in preventing machines from overfitting to the incidental statistics. These observations together demonstrate how our proposed ways for creating decoys improve the quality of multiple-choice Visual QA datasets.
Results on COCOQA and VQA2 For both datasets, we conduct experiments using the MLP-based models. As shown in Table 9.8, we clearly see that with only answers being visible to the model (MLP-A), the performance is close to random (on the column of IoU +QoU-decoys), and far from observing all three sources of information (MLP-IQA). Meanwhile, models that can observe either images and answers (MLP-IA) or questions and answers (MLP-QA) fail to predict as good as the model that observe all three sources of information. Results in Table 9.9 also shows a similar trend. These empirical observations meet our expectation and again verify the effectiveness of our proposed methods for creating decoys.

We also perform a more in-depth experiment on VQA2, removing triplets with Yes/No as the target. We name this subset as VQA2−. Table 9.10 shows the experimental results on VQA2−. Comparing to Table 9.9, we see that the overall performance for each model decreases as the dataset becomes more challenging on average. Specifically, the model that observes question and answer on VQA2− performs much worse than that on VQA2 (37.2% vs. 48.1%).

Analysis on different question and answer embeddings We consider GLOVE [143] and the embedding learned from translation [125] on both question and answer embeddings. The results on Visual7W (IoU + QoU, compared to Table 9.3 that uses WORD2VEC) are in Table 9.11. We do not observe significant difference among different embeddings, which is likely due to that both the questions and answers are short (averagely 7 words for questions and 2 for answers).

Analysis on random decoys We conduct the analysis on sampling random decoys, instead of our IoU-decoys and QoU-decoys, on Visual7W. We collect 6 additional random decoys for each Orig IQA triplet so the answer set will contain 10 candidates, the same as All in Table 9.3. We consider two strategies: (A) uniformly random decoys from unique correct answers, and (B) weighted random decoys w.r.t. their frequencies. The results are in Table 9.12. We see that different random strategies lead to drastically different results. Moreover, compared to the All column in Table 9.3, we see that our methods lead to a larger relative gap between MLP-IQA
<table>
<thead>
<tr>
<th>Method</th>
<th>IoU</th>
<th>QoU</th>
<th>IoU +QoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP-A</td>
<td>39.8</td>
<td>33.7</td>
<td>21.3</td>
</tr>
<tr>
<td>MLP-IA</td>
<td>40.3</td>
<td>53.0</td>
<td>31.0</td>
</tr>
<tr>
<td>MLP-QA</td>
<td>84.8</td>
<td>37.6</td>
<td>37.2</td>
</tr>
<tr>
<td>MLP-IQA</td>
<td>85.9</td>
<td>56.1</td>
<td>53.8</td>
</tr>
<tr>
<td>Random</td>
<td>25.0</td>
<td>25.0</td>
<td>14.3</td>
</tr>
</tbody>
</table>

Table 9.10: Test accuracy (%) on VQA2^-2017val, which contains 134,813 triplets.

<table>
<thead>
<tr>
<th>Method</th>
<th>GLOVE</th>
<th>Translation</th>
<th>WORD2VEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP-A</td>
<td>18.0</td>
<td>18.0</td>
<td>17.7</td>
</tr>
<tr>
<td>MLP-IA</td>
<td>23.6</td>
<td>23.2</td>
<td>23.6</td>
</tr>
<tr>
<td>MLP-QA</td>
<td>38.1</td>
<td>38.3</td>
<td>37.8</td>
</tr>
<tr>
<td>MLP-IQA</td>
<td>52.5</td>
<td>51.4</td>
<td>52.0</td>
</tr>
<tr>
<td>Random</td>
<td>14.3</td>
<td>14.3</td>
<td>14.3</td>
</tr>
</tbody>
</table>

Table 9.11: Test accuracy (%) on Visual7W, comparing different embeddings for questions and answers. The results are reported for the IoU +QoU-decoys.

Random strategies are worse than both random strategies, demonstrating the effectiveness of our methods in creating decoys.

9.5.5 Qualitative results

In Fig. 9.2, we present examples of image-question-target triplets from V7W, VQA, and VG, together with our IoU-decoys (A, B, C) and QoU-decoys (D, E, F). G is the target. The predictions by the corresponding MLP-IQA are also included. Ignoring information from images or questions makes it extremely challenging to answer the triplet correctly, even for humans.

Our automatic procedures do fail at some triplets, resulting in ambiguous decoys to the targets. See Fig. 9.3 for examples. We categorized those failure cases into two situations.

- Our filtering steps in Section 9.4 fail, as observed in the top example. The WUP-based similarity relies on the WordNet hierarchy. For some semantically similar words like “lady” and “woman”, the similarity is only 0.632, much lower than that of 0.857 between “cat”

<table>
<thead>
<tr>
<th>Method</th>
<th>(A)</th>
<th>(B)</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP-A</td>
<td>39.6</td>
<td>11.6</td>
<td>15.6</td>
</tr>
<tr>
<td>MLP-IA</td>
<td>53.4</td>
<td>40.3</td>
<td>22.2</td>
</tr>
<tr>
<td>MLP-QA</td>
<td>52.3</td>
<td>50.3</td>
<td>31.9</td>
</tr>
<tr>
<td>MLP-IQA</td>
<td>61.5</td>
<td>60.2</td>
<td>45.1</td>
</tr>
<tr>
<td>Random</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
</tr>
</tbody>
</table>

Table 9.12: Test accuracy (%) on Visual7W, comparing different random decoy strategies to our methods: (A) Orig + uniformly random decoys from unique correct answers, (B) Orig + weighted random decoys w.r.t. their frequencies, and All (Orig+IoU +QoU).
What is the train traveling over?

What is the color of his wetsuit?
A. When waves are bigger. B. It is not soft and fine. C. It is a picture of nature. D. Green. E. Blue. F. Red. G. It is black.

Where do the stairs lead?
A. A parking lot. B. The building. C. The windows. D. From the canal to the bridge. E. Up. F. To the building. G. To the plane.

What is the right man on the right holding?

What is the man wearing?

What are these people about to do?


Who is wearing glasses?

Where are several trees?

Figure 9.2: Example image-question-target triplets from Visual7W, VQA, and VG, together with our IoU-decoys (A, B, C) and QoU-decoys (D, E, F). G is the target. Machine’s selections are denoted by green ticks (correct) or red crosses (wrong).

Who is wearing glasses?

Where are several trees?

Figure 9.3: Ambiguous examples by our IoU-decoys (A, B, C) and QoU-decoys (D, E, F). G is the target. Ambiguous decoys F are marked.

and “dog”. This issue can be alleviated by considering alternative semantic measures by word2vec or by those used in [36, 39] for searching similar questions.

• The question is ambiguous to answer. In the bottom example in Fig. 9.3, both candidates D and F seem valid as a target. Another representative case is when asked about the background of an image. In images that contain sky and mountains in the distance, both terms can be valid.

9.6 Summary

We perform detailed analysis on existing datasets for multiple-choice Visual QA. We found that the design of decoys can inadvertently provide “shortcuts” for machines to exploit to perform well on the task. We describe several principles of constructing good decoys and propose automatic procedures to remedy existing datasets and create new ones. We conduct extensive empirical studies to demonstrate the effectiveness of our methods in creating better Visual QA datasets. The remedied datasets and the newly created ones are released and available at http://www.teds.usc.edu/website_vqa/.
Chapter 10

Cross-dataset Adaptation

10.1 Overview

In this chapter, we study the cross-dataset performance gap. Specifically, can the machine learn knowledge well enough on one dataset so as to answer adeptly questions from another dataset? Such study will highlight the similarity and difference among different datasets and guides the development of future ones. It also sheds lights on how well learning machines can understand visual and textual information in their generality, instead of learning and reasoning with dataset-specific knowledge.

Studying the performance gap across datasets is reminiscent of the seminal work by Torralba and Efros [181]. There, the authors study the bias in image datasets for object recognition. They have showed that the idiosyncrasies in the data collection process cause domain mismatch such that classifiers learnt on one dataset degrade significantly on another dataset [64, 62, 63, 95, 126, 180, 74, 179].

The language data in the Visual QA datasets introduces an addition layer of difficulty to bias in the visual data (see Fig. 10.1). For instance, [48] analyzes several datasets and illustrates their difference in syntactic complexity as well as within- and cross-dataset perplexity. As such, data in Visual QA datasets are likely more taletelling the origins from which datasets they come.

To validate this hypothesis, we had designed a Name That Dataset! experiment, similar to the one in [181] for comparing visual object images. We show that the two popular Visual QA datasets VQA [14] and Visual7W [230] are almost complete distinguishable using either the question or answer data. See Section 10.2 for the details of this experiment.

Thus, Visual QA systems that are optimized on one of those datasets can focus on dataset-specific knowledge such as the type of questions and how the questions and answers are phrased. This type of bias exploitation hinders cross-dataset generalization and does not result in AI systems that can reason well over vision and text information in different or new characteristics.

In this chapter, we investigate the issue of cross-dataset generalization in Visual QA. We assume that there is a source domain with a sufficiently large amount of annotated data such that a strong Visual QA model can be built, albeit adapted to the characteristics of the source domain well. However, we are interested in using the learned system to answer questions from another (target) domain. The target domain does not provide enough data to train a Visual QA system from scratch. We show that in this domain-mismatch setting, applying directly the learned system from the source to the target domain results in poor performance.
Figure 10.1: An illustration of the dataset bias in visual question answering. Given the same image, Visual QA datasets like VQA [14] (right) and Visual7W [230] (left) provide different styles of questions, correct answers (red), and candidate answer sets, each can contributes to the bias to prevent cross-dataset generalization.

Table 10.1: Results of Name That Dataset!

<table>
<thead>
<tr>
<th>Information</th>
<th>I</th>
<th>Q</th>
<th>T</th>
<th>D</th>
<th>Q + T</th>
<th>Q + D</th>
<th>T + D</th>
<th>Q + T + D</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>52.3%</td>
<td>76.3%</td>
<td>74.7%</td>
<td>95.8%</td>
<td>79.8%</td>
<td>97.5%</td>
<td>97.4%</td>
<td>97.5%</td>
<td>50.00%</td>
</tr>
</tbody>
</table>

We thus propose a novel adaptation algorithm for Visual QA. Our method has two components. The first is to reduce the difference in statistical distributions by transforming the feature representation of the data in the target dataset. We use an adversarial type of loss to measure the degree of differences—the transformation is optimized such that it is difficult to detect the origins of the transformed features. The second component is to maximize the likelihood of answering questions (in the target dataset) correctly using the Visual QA model trained on the source dataset. This ensures the learned transformation from optimizing domain matches retaining the semantic understanding encoded in the Visual QA model learned on the source domain.

The rest of this chapter is organized as follows. In Section 10.2, we analyze the dataset bias via the game Name That Dataset! In Section 10.3, we define tasks of domain adaptation for Visual QA. In Section 10.3.2, we describe the proposed domain adaptation algorithm. We leave the details of our algorithm in Section 10.5. In Section 10.4, we conduct extensive experimental studies and further analysis. Section 10.6 gives the conclusion.

10.2 Visual QA and bias in the datasets

In what follows, we describe a simple experiment Name That Dataset! to illustrate the biases in Visual QA datasets—questions and answers are idiosyncratically constructed such that a classifier can easily tell one apart from the other by using them as inputs. We then discuss how those biases give rise to poor cross-dataset generalization errors.
Given an IQA triplet, where \( A = \{C_1, \ldots, C_K\} \)

\[
\begin{align*}
I & \\
Q & \\
C_k & \\
\text{MLP} & \\
\text{M}(I, Q, C_1) & \rightarrow \\
\vdots & \\
\text{M}(I, Q, C_k) & \rightarrow \\
\text{M}(I, Q, C_K) & \rightarrow \hat{T} \\
\text{argmax} & \\
\end{align*}
\]

Figure 10.2: An illustration of the MLP-based model for multiple-choice Visual QA. Given an IQA triplet, we compute the \( M(I, Q, C_k) \) score for each candidate answer \( C_k \). The candidate answer that has the highest score is selected as the model’s answer.

### 10.2.1 Visual QA

In Visual QA datasets, a training or test example is a IQT triplet that consists of an image \( I \), a question \( Q \), and a (ground-truth) correct answer \( T \). During evaluation or testing, given a pair of \( I \) and \( Q \), a machine needs to generate an answer that matches exactly or is semantically similar to \( T \).

In this chapter, we focus on multiple-choice based Visual QA, since the two most-widely studied datasets—VQA [14] and Visual7W [230]—both consider such a setting. In this setting, the correct answer \( T \) is accompanied by a set of \( K \) “negative” candidate answers, resulting in a candidate answer set \( A \) consist of a single \( T \) and \( K \) decoys denoted by \( D \). An IQA triplet is thus \( \{I, Q, A = \{T, D_1, \ldots, D_K\}\} \). We use \( C \) to denote an element in \( A \). During testing, given \( I \), \( Q \), and \( A \), a machine needs to select \( T \) from \( A \). Multiple-choice based Visual QA has the benefit of simplified evaluation procedure and has been popularly studied [80, 211, 56, 168, 94]. Note that in the recent datasets like VQA2 [67], the candidate set \( A \) is expanded to include the most frequent answers from the whole training set, instead of a smaller subset typically used in earlier datasets. Despite this subtle difference, we do not lose in generality by studying cross-dataset generalization with multiple-choice based Visual QA datasets.

We follow Chapter 9 to train one-hidden-layer MLP models for multiple-choice based Visual QA. The MLP \( M \) takes the concatenated features of an IQC triplet as input and outputs a compatible score \( M(I, Q, C) \in [0, 1] \), measuring how likely \( C \) is the correct answer to the IQ pair. During training, \( M \) is learned to maximize the binary cross-entropy, where each IQC triplet is labeled with 1 if \( C \) is the correct answer; 0, otherwise. During testing, given an IQA triplet, the \( C \in A \) that leads to the highest score is selected as the model’s answer. We use the penultimate layer of ResNet-200 [73] as visual features to represent \( I \) and the average \textsc{word2vec} embeddings [131] as text features to represent \( Q \) and \( C \), as in [80]. See Fig. 10.2 for an illustration.

### 10.2.2 Bias in the datasets

We refer the term “bias” to any idiosyncrasies in the datasets that learning algorithms can overfit to and cause poor cross-dataset generalization.

---

1Some datasets provide multiple correct answers to accommodate the ambiguity in the answers.
**Name That Dataset!** To investigate the degree and the cause of the bias, we construct a game *Name That Dataset!*, similar to the one described in [181] for object recognition datasets. In this game, the machine has access to the examples (i.e., either IQT or IQA triplet) and needs to decide which dataset those examples belong to. We experiment on two popular datasets Visual7W [23] and VQA [14]. We use the same visual and text features described in Section 10.2.1 to represent I, Q, T, and D. We then concatenate these features to form the joint feature. We examine different combination of I, Q, T, D as the input to a one-hidden-layer MLP for predicting the dataset from which the sample comes. We sample 40,000, 5,000 and 20,000 triplets from each dataset and merge them to be the training, validation and test sets.

As shown in Table 10.1, all components but images lead to strong detection of the data origin, with the decoys contributing the most (i.e., 95.8% alone). Combining multiple components further improve the detection accuracy, suggesting that datasets contain different correlations or relationships among components. Concatenating all the components together results in nearly 100% classification accuracy. In other words, the image, question, and answers in each dataset are constructed characteristically. Their distributions (in the joint space) are sufficiently distant from each other. Thus, one would not expect a Visual QA system trained on one dataset to work well on the other datasets. See below for results validating this observation.

**Question Type is just one biasing factor** Question type is an obvious culprit of the bias. In Visual7W, questions are mostly in the 6W categories (i.e., what, where, how, when, why, who). On the other hand, the VQA dataset contains additional questions whose correct answers are either Yes or No. Those questions barely start with the 6W words. We create a new dataset called VQA− by removing the Yes or No questions from the original VQA dataset.

We reran the *Name That Dataset!* (after retraining on the new dataset). The accuracies of using Q or Q+T have dropped from 76.3% and 79.8% to 69.7% and 73.8%, respectively, which are still noticeably higher than 50% by chance. This indicates that the questions or correct answers may phrased differently between the two datasets (e.g., the length or the use of vocabularies). Combining them with the decoys (i.e., Q+T+D) raises the accuracy to 96.9%, again nearly distinguishing the two datasets completely. This reflects that the incorrect answers must be created very differently across the two datasets (In most cases, decoys are freely selected by the data collectors—being incorrect answers to the questions affords the data collectors to sample from unconstrained spaces of possible words and phrases.)

**Poor cross-dataset generalization** Using the model described in Section 10.2.1, we obtain the Visual QA accuracies of 65.7% and 55.6% on Visual7W and VQA− when training and testing using the same dataset. However, when the learned models are applied to the other dataset, the performance drops significantly to 53.4% (trained on VQA− but applied to Visual7W) and 28.1% (trained on Visual7W but applied to VQA−). See Table 10.3 for the details.

We further evaluate a variant of the spatial memory network [20], a more sophisticated Visual QA model. A similar performance drop is observed. See Table 10.7 for details.

---

2 Visual7W [23] has 3 decoys per triplet and VQA [14] has 17 decoys. For fair comparison, we subsample 3 decoys for VQA. We then average the word2vec embedding of each decoy to be the feature of decoys.
Table 10.2: Various Settings for cross-dataset Adaptation. Source domain always provide I, Q and A (T+D) while the target domain provides the same only during testing.

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Data from Target at Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting[Q]</td>
<td>Q</td>
</tr>
<tr>
<td>Setting[Q+T] (or [Q+T+D])</td>
<td>Q, T (or Q, T+D)</td>
</tr>
<tr>
<td>Setting[T] (or [T+D])</td>
<td>T (or T+D)</td>
</tr>
</tbody>
</table>

10.3 Cross-dataset adaptation

We propose to overcome the cross-dataset bias (and the poor cross-dataset generalization) with the idea of domain adaptation. Similar ideas have been developed in the past to overcome the dataset bias for object recognition [159, 64].

10.3.1 Main idea

We assume that we have a source domain (or dataset) with plenty of annotated data in the form of Image-Question-Candidate Answers (IQA) triplets such that we can build a strong Visual QA system. We are then interested in applying this system to the target domain. However, we do not assume there is any annotated data (i.e., IQA/IQT triplets) from the target domain such that re-training (either using the target domain alone or jointly with the source domain) or fine-tuning [137, 194] the system is feasible. Instead, the target domain provides unsupervised data. The target domain could provide images, images and questions (without either correct or incorrect answers), questions, questions with either correct or incorrect answers or both, or simply a set of candidate answers (either correct or incorrect or both). This last two scenarios are particularly interesting. From the results in Table 10.1, the discrepancy in textual information is a major contributor to domain mismatch, cf. the columns starting Q.

Given the target domain data, it is not feasible to train an “in-domain” model with the data (as it is incomplete and unsupervised). We thus need to model jointly the source domain supervised data and the target domain data that reflect distribution mismatch. Table 10.2 lists the settings we work on.

10.3.2 Approach

Our approach has two components. In the first part, we match features encoding questions and/or answers across two domains. In the second part, we ensure the correct answers from the target domain have higher likelihood in the Visual QA model trained on the source domain. Note that we do not re-train the Visual QA model as we do not have access to complete data on the target domain.

---

3 Annotated data from the target data, if any, can be easily incorporated into our method as a supervised learning discriminative loss.

4 Most existing datasets are derived from MSCOCO. Thus there are limited discrepancies between images, as shown in the column I in Table 10.1. Our method can also be extended to handle large discrepancy in images. Alternatively, existing methods of domain adaptation for visual recognition could be applied to images first to reduce the discrepancy.
Matching domain  The main idea is to transform features computed on the target domain \((TD)\) to match those features computed on the source domain \((SD)\). To this end, let \(g_q(\cdot)\) and \(g_a(\cdot)\) denote the transformation for the features on the questions and on the answers respectively. We also use \(f_q, f_t, f_d,\) and \(f_c\) to denote feature representations of a question, a correct answer, an incorrect decoy, or a candidate answer. In the Visual QA model, all these features are computed by the average \textsc{word2vec} embeddings of words.

The matching is computed as the Jensen-Shannon Divergence (JSD) between the two empirical distributions across the datasets. For the Setting\([Q]\), the matching is

\[
m(TD \rightarrow SD) = JSD(\hat{p}_{SD}(f_q), \hat{p}_{TD}(g_q(f_q)))
\]  (10.1)

where \(\hat{p}_{SD}(f_q)\) is the empirical distribution of the questions in the source domain and \(\hat{p}_{TD}(g_q(f_q))\) is the empirical distribution of the questions in the target domain, after being transformed with \(g_q(\cdot)\).

The JSD divergence between two distributions \(P\) and \(P'\) is computed as

\[
JSD(P,P') = \frac{1}{2} \left\{ KL\left( P; \frac{P+P'}{2} \right) + KL\left( P'; \frac{P+P'}{2} \right) \right\},
\]  (10.2)

while \(KL\) is the KL divergence between two distributions. The JSD divergence is closely related to discriminating two distributions with a binary classifier [66] but difficult to compute. We thus use an adversarial lose to approximate it. See Section 10.5 for details.

For both the Setting\([Q+T]\) and the Setting\([Q+T+D]\), the matching is

\[
m(TD \rightarrow SD) = JSD(\hat{p}_{SD}(f_q, f_t), \hat{p}_{TD}(g_q(f_q), g_a(f_t)))
\]  (10.3)

with the empirical distributions computed over both the questions and the correct answers. Note that even when the decoy information is available, we deliberately ignore them in computing domain mismatch. This is because the decoys can be designed very differently even for the same \(IQT\) triplet. Matching the distributions of \(D\) thus can cause undesired mismatch of \(T\) since they share the same transform during testing\(^5\).

For the Setting\([T]\) and Setting\([T+D]\), the matching is

\[
m(TD \rightarrow SD) = JSD(\hat{p}_{SD}(f_t), \hat{p}_{TD}(g_a(f_t)))
\]  (10.4)

while the empirical distributions are computed over the correct answers only.

Leverage Source Domain for Discriminative Learning  In the Setting\([Q+T]\), Setting\([Q+T+D]\), Setting\([T]\) and Setting\([T+D]\), the learner has access to the correct answers \(T\) (and the incorrect answers \(D\)) from the target domain. As we intend to use the transformed feature \(g_q(f_q)\) and

\(^5\)Consider the following highly contrived example. To answer the question “what is in the cup?”, the annotators in the source domain could answer with “water” as the correct answer, and “coffee”, “juice” as decoys, while the annotators in the target domain could answer with “sparkling water” (as that is the correct answer), then “cat” (as in cupcats), and “cake” (as in cupcakes) as decoys. While it is intuitive to match the distribution of correct answers, it makes less sense to match the distributions of the decoys as they are much more dispersed.
with the Visual QA model trained on the source domain, we would like those transformed features to have high likelihood of being correct (or incorrect).

To this end, we can leverage the source domain’s data which always contain both T and D. The main idea is to construct a Visual QA model on the source domain using the same partial information as in the target domain, then to assess how likely the transformed features remain to be correct (or incorrect).

In the following, we use the Setting\([Q+T+D]\) as an example (other settings can be formulated similarly). Let \(h_{SD}(q, c)\) be a model trained on the source domain such that it tells us the likelihood an answer \(c\) can be correct with respect to question \(q\). Without loss of generality, we assume \(h_{SD}(q, c)\) is the output of a binary logistic regression.

To use this model on the target data, we compute the following loss for every pair of question and candidate answer:

\[
\ell(q, c) = \begin{cases} 
- \log h_{SD}(g_q(f_q), q_a(f_c)) & \text{if } c \text{ is correct}, \\
- \log(1 - h_{SD}(g_q(f_q), q_a(f_c))) & \text{otherwise}.
\end{cases}
\]

The intuition is to raise the likelihood of any correct answers and lowering the likelihood of any incorrect ones. Thus, even we do not have a complete data for training models on the target domain discriminatively, we have found a surrogate to minimize,

\[
\hat{\ell}_{TD} = \sum_{(q, c) \in TD} \ell(q, c),
\]

measuring all the data provided in the target data and how they are likely to be correct or incorrect.

### 10.3.3 Joint optimization

We learn the feature transformation by jointly balancing the domain matching and the discriminative loss surrogate

\[
\text{arg min}_{g_q, g_a} m(TD \to SD) + \lambda \hat{\ell}_{TD}.
\]

We select \(\lambda\) to be large while still allowing \(m(TD \to SD)\) to decrease in optimization: \(\lambda\) is 0.5 for Setting\([Q+T+D]\) and Setting\([T+D]\), and 0.1 for the other experiments. The learning objective can be similarly constructed when the target domain provides Q and T, T, or T+D, as explained above. If the target domain only provides Q, we omit the term \(\hat{\ell}_{TD}\).

Once the feature transformations are learnt, we use the Visual QA model on the source domain \(M_{SD}\), trained using image, question, and answers all together to make an inference on an IQA triplet \((i, q, A)\) from the target

\[
\hat{t} = \text{arg max}_{c \in A} M_{SD}(f_i, g_q(f_q), g_a(f_c)),
\]

where we identify the best candidate answer from the pool of the correct answers and their decoys \(A\) using the source domain’s model. See Section 10.4.2 and 10.5 for the parameterization of \(g_q(\cdot)\) and \(g_a(\cdot)\), and details of the algorithm.
10.3.4 Related work on domain adaptation

Extensive prior work has been done to adapt the domain mismatch between datasets [182, 58, 183, 34, 65, 63], mostly for visual recognition while we study a new task of Visual QA. One popular method is to learn a transformation that aligns source and target domains according to a certain criterion. Inspired by the recent flourish of Generative Adversarial Network [66], many algorithms [58, 183, 34, 208] train a domain discriminator as a new criterion for learning such a transformation. Our method applies a similar approach, but aims to perform adaptation simultaneously on data with multiple modalities (i.e., images, questions, and answers). To this end, we leverage the Visual QA knowledge learned from the source domain to ensure that the transformed features are semantically aligned. Moreover, in contrast to most existing methods, we learn the transformation from the target domain to the source one, similar to [174, 183]6, enabling applying the learned Visual QA model from the source domain without re-training.

10.4 Empirical studies

10.4.1 Dataset

We first evaluate our algorithms on the domain adaptation settings defined in Section 10.3 between Visual7W [230] and VQA [14]. Experiments are conducted on both the original datasets and the revised version presented in Chapter 9. We then include Visual Genome [100], COCOQA [150], and VQA2 [67] with the decoys created in Chapter 9, leading to a comprehensive study of cross-dataset generalization. See Chapter 9 for more details.

Evaluation metric For Visual7W, VG, and COCOQA, we compute the accuracy of picking the correct answer from multiple choices. For VQA and VQA2, we follow its protocol to compute accuracy, comparing the picked answer to the 10 human-annotated correct answers. The accuracy is computed based on the number of exact matches among the 10 answers (divided by 3 and clipped at 1).

10.4.2 Experimental setup

Visual QA model In all our experiments, we use a one hidden-layer MLP model (with 8,192 hidden nodes and ReLU) to perform binary classification on each input IQC (image, question, candidate answer) triplet, following the setup as in [80] and Chapter 9. Please see Fig. 10.2 and Section 10.2.1 for explanation. The candidate $C \in A$ that has the largest score is then selected as the answer of the model. Such a simple model has achieved the state-of-the-art results on Visual7W and comparable results on VQA.

For images, we extract convolutional activation from the last layer of a 200-layer Residual Network [73]; for questions and answers, we extract the 300-dimensional WORD2VEC [131] embedding for each words in a question/answer and compute their average as the feature. We then concatenate these features to be the input to the MLP model. Besides the Visual QA model

---

6 Most DA algorithms, when given a target domain, adjust the features for both domains and retrain the source model on the adjusted features—they need to retrain the model when facing a new target domain. Note that [174, 183] do not incorporate the learned source-domain knowledge as ours.
that takes I, Q, and C as input, we also train two models that use only Q + C and C alone as the input. These two models can serve as $h_{SD}$ described in Sect 10.3.2.

Using simple models like MLP and average word2vec embeddings adds credibility to our studies—if models with limited capacity can latch on to the bias, models with higher capacity can only do better in memorizing the bias.

**Domain adaptation model** We parameterize the transformation $g_q(\cdot), g_a(\cdot)$ as a one hidden-layer MLP model (with 128 hidden nodes and ReLU) with residual connections directly from input to output. Such a design choice is due to the fact that the target embedding can already serve as a good starting point of the transforms. We approximate the $m(TD \rightarrow SD)$ measure by adversarially learning a one hidden-layer MLP model (with 8,192 hidden nodes and ReLU) for binary classification between the source and the transformed target domain data, following the same architecture as the classifier in Name That Dataset! game.

For all our experiments on training $g_q(\cdot), g_a(\cdot)$ and approximating $m(TD \rightarrow SD)$, we use Adam [96] for stochastic gradient-based optimization.

**Domain adaptation settings** As mentioned in Section 10.2, VQA (as well as VQA2) has around 30% of the IQA triplets with the correct answers to be either “Yes” or “NO”. On the other hand, Visual7W, COCOQA, and VG barely have triplets with such correct answers. Therefore, we remove those triplets from VQA and VQA2, leading to a reduced dataset VQA$^-$ and VQA2$^-$ that has 153,047/76,034 and 276,875/133,813 training/validation triplets, respectively.

We learn the Visual QA model using the training split of the source dataset and learn the domain adaptation transform using the training split of both datasets.

**Other implementation details** Questions in Visual7W, COCOQA, VG, VQA$^-$, and VQA2$^-$ are mostly started with the 6W words. The frequencies, however, vary among datasets. To encourage $g_q$ to focus on matching the phrasing style rather than transforming one question type to the others, when training the binary classifier for $m(TD \rightarrow SD)$ with Adams, we perform weighted sampling instead of uniform sampling from the source domain—the weights are determined by the ratio of frequency of each of the 6W question types between the target and source domain. This trick makes our algorithm more stable.

### 10.4.3 Experimental results on Visual7W and VQA−

We experiment on the five domain adaptation (DA) settings introduced in Section 10.3 using the proposed algorithm. We also compare with ADDA [183] and CORAL [174], two DA algorithms that can also learn transformations from the target to the source domain and achieves comparable results on many benchmark datasets. Specifically, we learn two transformations to match the (joint) distribution of the questions and target answers. We only report the best performance among the five settings for ADDA and CORAL. Table 10.3 and Table 10.4 summarize the results on the original and revised datasets, together with Direct transfer without any domain adaptation and Within domain performance where the Visual QA model is learned using the supervised data (i.e., IQA triplets) of the target domain. Such supervised data is inaccessible in the adaptation settings we considered.
Table 10.3: Domain adaptation (DA) results (in %) on original VQA [3] and Visual7W [230].

<table>
<thead>
<tr>
<th></th>
<th>VQA → Visual7W</th>
<th>Visual7W → VQA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direct</strong></td>
<td>[174] [183]</td>
<td>[174] [183]</td>
</tr>
<tr>
<td>[Q]</td>
<td>53.4</td>
<td>28.1</td>
</tr>
<tr>
<td>[T]</td>
<td>53.4</td>
<td>26.9</td>
</tr>
<tr>
<td>[T+D]</td>
<td>54.1</td>
<td>29.2</td>
</tr>
<tr>
<td>[Q+T]</td>
<td>53.6</td>
<td>28.1</td>
</tr>
<tr>
<td>[Q+T+D]</td>
<td>54.5</td>
<td>29.7</td>
</tr>
<tr>
<td><strong>Within</strong></td>
<td>55.7</td>
<td>33.6</td>
</tr>
<tr>
<td></td>
<td>55.2</td>
<td>29.4</td>
</tr>
<tr>
<td></td>
<td><strong>58.5</strong></td>
<td><strong>35.2</strong></td>
</tr>
<tr>
<td></td>
<td>65.7</td>
<td>55.6</td>
</tr>
</tbody>
</table>

Table 10.4: Domain adaptation (DA) results (in %) on revised VQA and Visual7W from Chapter 9. (best DA result in bold)

<table>
<thead>
<tr>
<th></th>
<th>VQA → Visual7W</th>
<th>Visual7W → VQA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direct</strong></td>
<td>[174] [183]</td>
<td>[174] [183]</td>
</tr>
<tr>
<td>[Q]</td>
<td>46.1</td>
<td>45.6</td>
</tr>
<tr>
<td>[T]</td>
<td>47.2</td>
<td>45.3</td>
</tr>
<tr>
<td>[T+D]</td>
<td>47.8</td>
<td>45.9</td>
</tr>
<tr>
<td>[Q+T]</td>
<td>46.2</td>
<td>45.9</td>
</tr>
<tr>
<td>[Q+T+D]</td>
<td>47.6</td>
<td>45.9</td>
</tr>
<tr>
<td><strong>Within</strong></td>
<td>48.4</td>
<td>47.8</td>
</tr>
<tr>
<td></td>
<td><strong>49.3</strong></td>
<td><strong>48.1</strong></td>
</tr>
<tr>
<td></td>
<td>52.0</td>
<td>53.7</td>
</tr>
</tbody>
</table>

**Domain mismatch hurts cross-dataset generalization**  The significant performance drop in comparing **Within** domain and **Direct** transfer performance suggests that the learned Visual QA models indeed exploit certain domain-specific bias that may not exist in the other datasets. Such a drop is much severe between the original datasets than the revised datasets. Note that the two versions of datasets are different only in the decoys, and the revised datasets create decoys for both datasets by the same automatic procedure. Such an observation, together with the finding from *Name That Dataset!* game, indicate that decoys contribute the most to the domain mismatch in Visual QA.

**Comparison on domain adaptation algorithms**  Our domain adaptation algorithm outperforms **Direct** transfer in all the cases. On contrary, CORAL [174], which aims to match the first and second order statistics between domains, fails in several cases, indicating that for domain adaptation in Visual QA, it is crucial to consider higher order statistics.

We also examine setting $\lambda$ in Eq. (10.6) to 0 for the [T] and [Q+T] settings\(^7\) (essentially ADDA [183] extended to multiple modalities), which leads to a drop of $\sim 1\%$, demonstrating the effectiveness of leveraging the source domain for discriminative learning. See the Section 10.4.5 for more details.

**Different domain adaptation settings**  Among the five settings, we see that [T] generally gives larger improvement over **Direct** than [Q], suggesting that the domain mismatch in answers hinder more in cross-dataset generalization.

\(^7\)When $\lambda = 0$, D has no effect (i.e., [Q+T+D] is equivalent to [Q+T]).
Table 10.5: DA results (in %) on original datasets, with target data sub-sampling by 1/16. FT: fine-tuning. (best DA result in bold)

<table>
<thead>
<tr>
<th>VQA → Visual7W</th>
<th>Direct</th>
<th>[174]</th>
<th>[183]</th>
<th>[Q]</th>
<th>[T]</th>
<th>[T+D]</th>
<th>[Q+T]</th>
<th>[Q+T+D]</th>
<th>Within</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>53.4</td>
<td>52.6</td>
<td>54.0</td>
<td>53.6</td>
<td>54.4</td>
<td>56.3</td>
<td>55.1</td>
<td>58.2</td>
<td>53.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visual7W → VQA</th>
<th>Direct</th>
<th>[174]</th>
<th>[183]</th>
<th>[Q]</th>
<th>[T]</th>
<th>[T+D]</th>
<th>[Q+T]</th>
<th>[Q+T+D]</th>
<th>Within</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>28.1</td>
<td>26.5</td>
<td>28.8</td>
<td>28.1</td>
<td>29.3</td>
<td>33.4</td>
<td>29.2</td>
<td>35.2</td>
<td>44.1</td>
</tr>
</tbody>
</table>

Table 10.6: DA results (in %) on revised datasets, with target data sub-sampling by 1/16. FT: fine-tuning. (best DA result in bold)

<table>
<thead>
<tr>
<th>VQA → Visual7W</th>
<th>Direct</th>
<th>[174]</th>
<th>[183]</th>
<th>[Q]</th>
<th>[T]</th>
<th>[T+D]</th>
<th>[Q+T]</th>
<th>[Q+T+D]</th>
<th>Within</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>46.1</td>
<td>45.6</td>
<td>47.8</td>
<td>46.1</td>
<td>47.5</td>
<td>47.6</td>
<td>48.3</td>
<td>49.1</td>
<td>39.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visual7W → VQA</th>
<th>Direct</th>
<th>[174]</th>
<th>[183]</th>
<th>[Q]</th>
<th>[T]</th>
<th>[T+D]</th>
<th>[Q+T]</th>
<th>[Q+T+D]</th>
<th>Within</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>45.6</td>
<td>44.8</td>
<td>45.6</td>
<td>46.0</td>
<td>45.9</td>
<td>47.8</td>
<td>45.8</td>
<td>48.0</td>
<td>43.1</td>
</tr>
</tbody>
</table>

Extra information on top of [T] or [Q] generally benefits the domain adaptation performance, with [Q+T+D] giving the best performance. Note that different setting corresponds to different objectives in Eq. (10.6) for learning the transformations $g_q$ and $g_a$. Comparing [T] to [T+D], we see that adding D helps take more advantage of exploiting the source domain’s Visual QA knowledge, leading to a $g_a$ that better differentiates the correct answers from the decoys. On the other hand, adding T to [Q], or vice versa, helps constructing a better measure to match the feature distribution between domains.

**Domain adaptation using a subset of data** The domain adaptation results presented in Table 10.3 and 10.4 are based on learning the transformations using all the training examples of the source and target domain. We further investigate the robustness of the proposed algorithm under a limited number of target examples. We present the results using only 1/16 of the them in Table 10.5 and 10.6. The proposed algorithm can still learn the transformations well under such a scenario, with a slight drop in performance (i.e., < 0.5%). In contrast, learning Visual QA models with the same amount of limited target data (assuming the IQA triplets are accessible) from scratch leads to significant performance drop. We also include the results by fine-tuning, which is infeasible in any setting of Table 10.2 but can serve as an upper bound.

We further consider domain adaptation (under Setting[Q+T+D] with $\lambda = 0.1$) between Visual7W [230] and VQA− [14] for both the original and revised decoys using $\frac{1}{2a}$ of training data of the target domain, where $a \in [0, 1, \cdots, 6]$. The results are shown in Fig. 10.3. Note that the Within results are from models trained on the same sub-sampled size using the supervised IQA triplets from the target domain.

98
As shown, our domain adaptation (DA) algorithm is highly robust to the accessible data size from the target domain. On the other hand, the Within results from models training from scratch significantly degrade when the data size decreases. Except the case Visual7W → VQA− (original), domain adaptation (DA) using our algorithm outperforms the Within results after a certain sub-sampling rate. For example, on the case VQA− → Visual7W (revised), DA already outperforms Within under \( \frac{1}{4} \) of the target data.

**Results on sophisticated Visual QA model** We further investigate a variant of the spatial memory network (SMem) [203] and the HieCoAtt model [122] for Visual QA, which utilizes the question to guide the visual attention on certain parts of the image for extracting better visual features. The results are shown in Table 10.7 and 10.8, where a similar trend of improvement is observed.

**Qualitative analysis** We shown in Fig 10.4 the results on each question type (out of the 6\( W \) words) when transferring from VQA− to Visual7W in Table 10.3 (on the original datasets). DA ([Q+T+D]) outperforms Direct at all the question types. The question type that improves the most from Direct to DA is “When” (from 41.8 to 63.4, while Within is 80.3). Other types improve 1.0 ∼ 5.0. This is because that the “When”-type question is scarcely seen in VQA−, and our
Table 10.7: DA results (in %) on VQA and Visual7W (both original and revised) using a variant of the SMem model [203].

<table>
<thead>
<tr>
<th></th>
<th>original</th>
<th>revised</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQA $\rightarrow$ Visual7W</td>
<td>56.3</td>
<td>48.6</td>
</tr>
<tr>
<td>Visual7W $\rightarrow$ VQA $\rightarrow$ Visual7W $\rightarrow$ VQA $\rightarrow$ Visual7W $\rightarrow$ VQA $\rightarrow$ Visual7W $\rightarrow$ VQA $\rightarrow$ Visual7W</td>
<td>61.0</td>
<td>51.2</td>
</tr>
<tr>
<td>Direct $\rightarrow$ [Q+T+D]</td>
<td>65.9</td>
<td>52.8</td>
</tr>
<tr>
<td>Within Direct $\rightarrow$ [Q+T+D]</td>
<td>27.5</td>
<td>46.6</td>
</tr>
<tr>
<td>[Q+T+D] Within [Q+T+D]</td>
<td>34.1</td>
<td>48.4</td>
</tr>
<tr>
<td>[Q+T+D] Within [Q+T+D]</td>
<td>58.5</td>
<td>58.6</td>
</tr>
</tbody>
</table>

Table 10.8: DA results (in %) on VQA and Visual7W (both original and revised) using a variant of the HieCoAtt model [122].

<table>
<thead>
<tr>
<th></th>
<th>original</th>
<th>revised</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQA $\rightarrow$ Visual7W</td>
<td>51.5</td>
<td>46.4</td>
</tr>
<tr>
<td>Visual7W $\rightarrow$ VQA $\rightarrow$ Visual7W $\rightarrow$ VQA $\rightarrow$ Visual7W $\rightarrow$ VQA $\rightarrow$ Visual7W $\rightarrow$ VQA $\rightarrow$ Visual7W</td>
<td>56.2</td>
<td>48.2</td>
</tr>
<tr>
<td>Direct $\rightarrow$ [Q+T+D]</td>
<td>63.9</td>
<td>51.5</td>
</tr>
<tr>
<td>Within Direct $\rightarrow$ [Q+T+D]</td>
<td>27.2</td>
<td>44.5</td>
</tr>
<tr>
<td>[Q+T+D] Within [Q+T+D]</td>
<td>33.1</td>
<td>46.3</td>
</tr>
<tr>
<td>[Q+T+D] Within [Q+T+D]</td>
<td>54.8</td>
<td>55.6</td>
</tr>
</tbody>
</table>

DA algorithm, together with the weighted sampling trick, significantly reduces the mismatch of question/answer phrasing of such a type.

10.4.4 Experimental results across five datasets

We perform a more comprehensive study on transferring the learned Visual QA models across five different datasets. We use the revised candidate answers for all of them to reduce the mismatch on how the decoys are constructed. We consider the [Q+T+D] setting, and limit the disclosed target data to 1/16 of its training split size. The models for Within are also trained on such a size, using the supervised IQA triplets. Table 10.9 summarizes the results, where rows/columns correspond to the source/target domains.

On almost all (source, target) pairs, domain adaptation (DA) outperforms Direct, demonstrating the wide applicability and robustness of our algorithm. The exception is on (VQA $\rightarrow$, VQA2 $\rightarrow$), where DA degrades by 0.1%. This is likely due to the fact that these two datasets are constructed similarly and thus no performance gain can be achieved. Such a case can also be seen between Visual7W and VG. Specifically, domain adaptation is only capable in transferring the knowledge learned in the source domain, but cannot acquire new knowledge in the target domain.

The reduced training size significantly limits the performance of training from scratch (Within). In many cases Within is downplayed by DA, or even by Direct, showing the essential demand to leverage source domain knowledge. Among the five datasets, Visual QA models trained on VG
seems to generalize the best—the DA results to any target domain outperforms the corresponding Within—indicating the good quality of VG.

In contrast, Visual QA models trained on COCOQA can hardly transfer to other datasets—none of its DA results to other datasets is higher than Within. It is also interesting to see that none of the DA results from other source domain (except VG) to COCOQA outperforms COCOQA’s Within. This is, however, not surprising given how differently in the way COCOQA is constructed; i.e., the questions and answers are automatically generated from the captions in MSCOCO. Such a significant domain mismatch can also be witnessed from the gap between Direct and DA on any pair that involves COCOQA. The performance gain by DA over Direct is on average over 4.5%, larger than the gain of any other pair, further demonstrating the effectiveness of our algorithms in reducing the mismatch between domains.

### 10.4.5 Additional experimental results

**The effect of the discriminative loss surrogate** We provide in Table 10.10 the domain adaptation results on the [T] and [Q+T] settings when \( \lambda \) is set to 0 (cf. Eq. (10.6)), which corresponds to omitting the discriminative loss surrogate \( \hat{\ell}_{TD} \). In most of the cases, the results with \( \lambda = 0.1 \) outperforms \( \lambda = 0 \), showing the effectiveness of leveraging the source domain for discriminative learning. Also note that when D is provided for the target domain (i.e., [T+D] or [Q+T+D]), it is the \( \hat{\ell}_{TD} \) term that utilizes the information of D, leading to better results than [T] or [Q+T], respectively.
Table 10.10: Domain adaptation (DA) results (in %) with or without the discriminative loss surrogate term

<table>
<thead>
<tr>
<th></th>
<th>(\text{original} )</th>
<th>(\text{revised} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\text{VQA}^- \to \text{Visual7W} )</td>
<td>(\text{Visual7W} \to \text{VQA}^- )</td>
</tr>
<tr>
<td>(\lambda = 0)</td>
<td>54.1</td>
<td>29.2</td>
</tr>
<tr>
<td>(\lambda = 0.1)</td>
<td>54.5</td>
<td>29.2</td>
</tr>
</tbody>
</table>

Table 10.10: Domain adaptation (DA) results (in %) with or without the discriminative loss surrogate term

We further experiment on different values of \(\lambda\), as shown in Fig. 10.5. For [Q+T], we achieve consistent improvement for \(\lambda \leq 0.1\). For [Q+T+D], we can get even better results by choosing a larger \(\lambda\) (e.g. \(\lambda = 0.5\)).

**Open-ended (OE) results** We apply Visual QA models learned with the multiple-choice setting to evaluate on the open-ended one (i.e., select an answer from the top frequent ones, or from the set of all possible answers in the training data). The result on transferring from VQA\(^-\) to COCOQA is in Table 10.11. Our adaptation algorithm still helps transferring.

**Experimental results across five datasets: using the whole target domain data** Table 10.12 summarizes the results of the same study, except that now all the training examples of the target domain are used. The models for **Within** are also trained on such a size, using the supervised IQA triplets. Compared to Table 10.9, we see that the performance drop of DA from using all the training examples of the target domain to 1/16 of them is very small (mostly smaller than 0.3%), demonstrating the robustness of our algorithm under limited training data. On the other hand, the drop of **Within** is much more significant—for most of the (source, target) pairs, the drop is at least 10%. For most of the (source, target) pairs shown in Table 10.12, **Within** outperforms **Direct** and DA. The notable exceptions are (VG, Visual7W) and (VQA\(^-\), VQA\(^-\)). This is likely due to the fact that VG and Visual7W are constructed similarly while VG has more training examples than
Table 10.11: OE results (VQA → COCOQA, sub-sampled by 1/16).

<table>
<thead>
<tr>
<th></th>
<th>Direct</th>
<th>[Q+T+D]</th>
<th>Within</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16.7</td>
<td>24.0</td>
<td>26.9</td>
</tr>
</tbody>
</table>

Table 10.12: Transfer results (in %) across datasets (the decoys are generated according to Chapter 9). The setting for domain adaptation (DA) is on [Q+T+D] using all the training examples of the target domain.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct DA</td>
<td>52.0</td>
<td>46.1</td>
<td>58.1</td>
<td>30.1</td>
<td>48.8</td>
</tr>
<tr>
<td>Within</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Direct DA</td>
<td>45.6</td>
<td>53.7</td>
<td>52.6</td>
<td>34.4</td>
<td>51.0</td>
</tr>
<tr>
<td>Within</td>
<td>48.1</td>
<td>-</td>
<td>54.4</td>
<td>52.0</td>
<td>-</td>
</tr>
<tr>
<td>Direct DA</td>
<td>49.1</td>
<td>44.8</td>
<td>58.5</td>
<td>29.1</td>
<td>47.3</td>
</tr>
<tr>
<td>Within</td>
<td>49.6</td>
<td>47.9</td>
<td>58.5</td>
<td>33.4</td>
<td>49.6</td>
</tr>
<tr>
<td>Direct DA</td>
<td>58.0</td>
<td>59.0</td>
<td>65.5</td>
<td>75.8</td>
<td>60.3</td>
</tr>
<tr>
<td>Within</td>
<td>63.0</td>
<td>64.7</td>
<td>68.8</td>
<td>-</td>
<td>65.2</td>
</tr>
<tr>
<td>Direct DA</td>
<td>75.8</td>
<td>75.8</td>
<td>75.8</td>
<td>75.8</td>
<td>75.8</td>
</tr>
<tr>
<td>Within</td>
<td>75.8</td>
<td>75.8</td>
<td>75.8</td>
<td>75.8</td>
<td>75.8</td>
</tr>
</tbody>
</table>

Visual7W. The same fact applies to VQA2 and VQA. Therefore, the Visual QA model learned on the source domain can be directly applied to the target domain and leads to better results than Within.

10.5 Details on the proposed domain adaptation algorithm

10.5.1 Approximating the JSD divergence

As mentioned in Section 10.3.2, we use the Jensen-Shannon Divergence (JSD) to measure the domain mismatch between two domains according to their empirical distributions. Dependent on the domain adaptation (DA) setting, the empirical distribution is computed on the (transformed) questions, (transformed) correct answers, or both.

Since JSD is hard to compute, we approximate it by training a binary classifier WhichDomain(·) to detect the domain of a question Q, a correct answer T, or a QT pair, following the idea of Generative Adversarial Network [66]. The architecture of WhichDomain(·) is exactly the same as that used for Name that dataset!, except that the input features of examples from the target domain are after the transformations \(g_q(\cdot)\) and \(g_a(\cdot)\).

10.5.2 Details on the proposed algorithm

We summarize the proposed domain adaptation algorithm for Visual QA under Setting [Q+T+D] in Algorithm 1. Algorithms of the other settings can be derived by removing the parts corresponding to the missing information.

10.6 Summary

We study cross-dataset adaptation for visual question answering. We first analyze the causes of bias in existing datasets. We then propose to reduce the bias via domain adaptation so as
to improve cross-dataset knowledge transfer. To this end we propose a novel domain adap-
tation algorithm that minimizes the domain mismatch while leveraging the source domain’s Visual
QA knowledge. Through experiments on knowledge transfer among five popular datasets, we
demonstrate the effectiveness of our algorithm, even under limited and fragment target domain
information.

**Notations** Denote the features of Q, T, D by \(f_q, f_t, \) and \(f_d\). The D here stands for one
decoys.

**Goal** Learn transformations \(g_q(\cdot), g_a(\cdot)\) and a binary domain classifier \(\text{WhichDomain}(\cdot)\),
where \(\phi_q, \phi_a,\) and \(\theta\) are the parameters to learn, respectively. \(\text{WhichDomain}(\cdot)\) gives the
conditional probability of being from the source domain;

```
for number of training iterations do
    Initialize the parameters \(\theta\) of \(\text{WhichDomain}(\cdot)\);
    for \(k\) steps do
        Sample a mini-batch of \(m\) pairs \(\{Q_{SD}(j), T_{SD}(j)\}_{j=1}^m \sim SD\);
        Sample a mini-batch of \(m\) pairs \(\{Q_{TD}(j), T_{TD}(j)\}_{j=1}^m \sim TD\);
        Update \(\text{WhichDomain}(\cdot)\) by ascending its stochastic gradient;
        \[
        \nabla_{\theta} \left\{ \frac{1}{m} \sum_{j=1}^{m} \left[ \log \text{WhichDomain}(\{f_q(j), f_t(j)\}) + 
        \log(1 - \text{WhichDomain}(\{g_q(f_q(j)), g_a(f_t(j))\})) \right] \right\}
        \]
    end
    for \(l\) steps do
        Sample a mini-batch of \(m\) triplet \(\{Q_{TD}(j), T_{TD}(j), D_{TD}(j)\}_{j=1}^m \sim TD\);
        Update the transformations by descending their stochastic gradients;
        \[
        \nabla_{\phi_q, \phi_a} \left\{ \frac{1}{m} \sum_{i=1}^{m} \log(1 - \text{WhichDomain}(\{g_q(f_q(j)), g_a(f_t(j))\})) + 
        \lambda \left( \ell(\{g_q(f_q(j)), g_a(f_t(j))\}) + \ell(\{g_q(f_q(j)), g_a(f_d(j))\}) \right) \right\}
        \]
    end
```

Algorithm 1: The proposed domain adaptation algorithm for Setting[Q+T+D]. \(D_{TD}(j)\) denotes a
single decoy. When the decoys of the target domain are not provided (i.e., Setting[Q+T]), the \(\ell\)
term related to \(D_{TD}(j)\) is ignored.
Chapter 11

Learning Answer Embedding for Visual Question Answering

In this chapter we propose a novel probabilistic model for Visual QA. The key idea is to infer two sets of embeddings: one for the image and the question jointly and the other for the answers. The learning objective is to learn the best parameterization of those embeddings such that the correct answer has higher likelihood among all possible answers. In contrast to several existing approaches of treating Visual QA as multi-way classification, the proposed approach takes the semantic relationships (as characterized by the embeddings) among answers into consideration, instead of viewing them as independent ordinal numbers. Thus, the learned embedded function can be used to embed unseen answers (in the training dataset). These properties make the approach particularly appealing for transfer learning for open-ended Visual QA, where the source dataset on which the model is learned has limited overlapping with the target dataset in the space of answers. We have also developed large-scale optimization techniques for applying the model to datasets with a large number of answers, where the challenge is to properly normalize the proposed probabilistic models. We validate our approach on several Visual QA datasets and investigate its utility for transferring models across datasets. The empirical results have shown that the approach performs well not only on in-domain learning but also on transfer learning.

11.1 Introduction

In Visual QA, the machine is presented with an image and a related question and needs to output a correct answer. There are several ways of “outputting”, though. One way is to ask the machine to generate a piece of free-form texts [59]. However, this often requires humans to decide whether the answer is correct or not. Thus, scaling this type of evaluation to assess a large amount of data (on a large number of models) is challenging.

Automatic evaluation procedures have the advantage of scaling up. There are two major paradigms. One is to use multiple-choice based Visual QA [230, 3, 150]. In this setup, for each pair of image and question, a correct answer is mixed with a set of incorrect answers and the learner optimizes to select the correct one. While popular, it is difficult to design good incorrect answers without shortcuts such that learners are not able to exploit (cf. Chapter 9).

The other paradigm that is amenable to automatic evaluation revises the pool of possible answers to be the same for any pair of image and question [67, 14], i.e., open-ended Visual QA. In particular, the pool is composed of most frequent $K$ answers in the training dataset. This has the advantage of framing the task as a multi-way classifier that outputs one of the $K$ categories, with the image and the question as the input to the classifier.
Q1: Where is the ball?

The little boy.
In the basket.
In the air.
The man in the white uniform.

Q2: Who is holding the bat?

The player.
The woman.

Figure 11.1: Conceptual diagram of our approach. We learn two embedding functions to transform image question pair \((i, q)\) and (possible) answer \(a\) into a joint embedding space. The distance (by inner products) between the embedded \((i, q)\) and \(a\) is then measured and the closest \(a\) (in red) would be selected as the output answer.

However, while alleviating the bias of introducing incorrect answers that are image and question specific, the open-end Visual QA approaches also suffer from several problems. First, treating the answers as independent categories (as entailed by the multi-way classification) removes the semantic relationship between answers. For example, the answers of “running” and “jogging” (to the question “what is the woman in the picture doing?”) are semantically close, so one would naturally infer the corresponding images are visually similar. However, treating “running” and “jogging” as independent categories “choice i” and “choice j” would not automatically regularize the learner to ensure the classifier’s outputs of visually similar images and semantically similar questions to be semantically close. In other words, we would desire the outputs of the Visual QA model express semantic proximities aligned with visual and semantic proximities at the inputs. Such alignment will put a strong prior on what the models can learn and prevent them from exploiting biases in the datasets, thus become more robust.

Secondly, Visual QA models learned on one dataset do not transfer to another dataset unless the two datasets share the same space of top \(K\) answers—if there is a difference between the two spaces (for example, as “trivial” as changing the frequency order of the answers), the classifier will make a substantial number of errors. This is particularly alarming unless we construct a system a priori to map one set of answers to another set, we are likely to have very poor transfer across datasets and would have to train a new Visual QA model whenever we encounter a new dataset. In fact, for two popular Visual QA datasets, about 10% answers are shared and of top-\(K\) answers (where \(K < 10,000\)), only 50% answers are shared. We refer readers to Section 11.3.5 and Table 11.6 for more results.

In this chapter, we propose a new learning model to address these challenges. Our main idea is to learn also an embedding of the answers. Together with the (joint embedding) features of image and question in some spaces, the answer embeddings parameterize a probabilistic model describing how the answers are similar to the image and question pair. We learn the embeddings for the answers as well as the images and the questions to maximize the correct answers’ likelihood. The learned model thus aligns the semantic similarity of answers with the visual/semantic similarity of the image and question pair. Furthermore, the learned model can also embed any
unseen answers, thus can generalize from one dataset to another one. Fig. 11.1 illustrates the main idea of our approach.

Our method needs to learn embeddings of hundreds and thousands of answers. Thus to optimize our probabilistic model, we overcome the challenge by introducing a computationally efficient way of adaptively sampling negative examples in a minibatch.

Our model also has the computational advantage that for each pair of image and question, we only need to compute the joint embedding of image and question for once, irrespective of how many candidate answers one has to examine. On the other end, models such as [80, 56] learn a joint embedding of the triplet (image, question and answer) needs to compute embeddings at the linear order of the number of candidate answers. When the number of candidate answers need to be large (to obtain better coverage), such models do not scale up easily.

While our approach is motivated by addressing challenges in open-end Visual QA, the proposed approach trivially includes multiple-choice based Visual QA as a special case and is thus equally applicable. We extensively evaluated our approach on several existing datasets, including Visual7W [230], VQA2 [67], and Visual Genome [100]. We show the gain in performance by our approach over the existing approaches that are based on multi-way classification. We also show the effectiveness of our approach in transferring models trained on one dataset to another. To our best knowledge, we are likely the first to examine the challenging issue of transferability in the open-end Visual QA task.

The rest of the chapter is organized as follows. Section 11.2.1 introduces the notation and problem setup. Section 11.2.2 presents our proposed methods. Section 11.3 shows our empirical results on multiple Visual QA datasets.

11.2 Methods

In what follows, we describe our approach in detail. We start by describing a general setup for Visual QA and introducing necessary notations. We then introduce the main idea, followed by detailed descriptions of the method and important steps to scale the method to handling hundreds of thousands negative samples.

11.2.1 Setup and notations

In the Visual QA task, the machine is given an image \( i \) and a question \( q \), and is asked to generate an answer \( a \). In this work, we focus on the open-ended setting where \( a \) is a member of a set \( \mathcal{A} \). This set of candidate answers is intuitively “the universe of all possible answers”. However, in practice, it is approximated by the top \( K \) most frequent correct answers in a training set [122, 56, 209], plus all the incorrect answers in the dataset (if any). Another popular setting is multiple-choice based. For each pair of \((i, q)\), the set \( \mathcal{A} \) is different (this set is either automatically generated (cf. Chapter 9) or manually generated [230, 3]). Without loss of generality, however, we use \( \mathcal{A} \) to represent both. Whenever necessary, we clarify the special handling we would need for \((i, q)\) specific candidate set.

\( ^1 \)Our work focuses on the transferability across datasets with different question and answer spaces. We leave visual transferability (e.g., by domain adaptation) as future work.
We distinguish two subsets in $\mathcal{A}$ with respect to a pair $(i, q)$: $\mathcal{T}$ and $\mathcal{D} = \mathcal{A} - \mathcal{T}$. The set $\mathcal{T}$ contains all the correct answers for $(i, q)$ — it could be a singleton or in some cases, contains multiple semantically similar answers to the correct answer (e.g., “policeman” to “police officer”), depending on the datasets. The set $\mathcal{D}$ contains all the incorrect (or undesired) answers.

A training dataset is thus denoted by a set of $N$ distinctive triplets $\mathcal{D} = \{(i_n, q_n, T_n)\}$ when only the correct answers are given, or $\tilde{\mathcal{D}} = \{(i_n, q_n, \mathcal{A}_n = \mathcal{T}_n \cup \mathcal{D}_n)\}$ when both the correct and incorrect answers are given.

Note that by $i$, $q$ or $a$, we refers to their “raw” formats (an image in pixel values, and a question or an answer in its textual forms).

### 11.2.2 Main idea

Our main idea is motivated by two deficiencies in the current approaches for open-ended Visual QA [3]. In those methods, it is common to construct a $K$-way classifier so that for each $(i, q)$, the classifier outputs $k$ that corresponds to the correct answer (i.e., the $k$-th element in $\mathcal{A}$ is the correct answer).

However, this classification paradigm cannot capture all the information encoded in the dataset for us to derive better models. First, by equating two different answers $a_k$ and $a_l$ with the ordinal numbers $k$ and $l$, we lose the semantic kinship between the two. If there are two triplets $(i_m, q_m, a_k \in \mathcal{T}_m)$ and $(i_n, q_n, a_l \in \mathcal{T}_n)$ having similar visual appearance between $i_m$ and $i_n$ and similar semantic meaning between $q_m$ and $q_n$, we would expect $a_k$ and $a_l$ to have some degrees of semantic similarity. In a classification framework, such expectation cannot be fulfilled as the assignment of ordinal numbers $k$ or $l$ to either $a_k$ or $a_l$ can be arbitrary such that the difference between $k$ and $l$ does not preserve the similarity between $a_k$ and $a_l$. However, observing such similarity at both the inputs to the classifier and the outputs of the classifier is beneficial and adds robustness to learning.

The second flaw with the multi-way classification framework is that it does not lend itself to generalize across two datasets with little or no overlapping in the candidate answer sets $\mathcal{A}$. Unless there is a prior defined mapping between the two sets, the classifier trained on one dataset is not applicable to the other dataset.

We propose a new approach to overcome those deficiencies. The key idea is to learn embeddings of all the data. The embedding functions, when properly parameterized and learned, will preserve similarity and will generalize to unseen answers (in the training data).

**Embeddings** We first define a joint embedding function $f_\theta(i, q)$ to generate the joint embedding of the pair $i$ and $q$. We also define an embedding function $g_\phi(a)$ to generate the embedding of an answer $a$. We will postpone to later to explain why we do not learn a function that generates the joint embedding of the triplet.

The embedding functions are parameterized by $\theta$ and $\phi$, respectively. In this work, we use deep learning models such as multi-layer perceptron (MLP) and Stacked Attention Network (SAN) [209, 93] (after removing the classifier at the last layer). In principle, any representation network can be used — our focus is on how to use the embeddings.
Probabilistic Model of Compatibility (PMC) Given a triplet \((i_n, q_n, a) \in \mathcal{T}_n\) where \(a\) is a correct answer, we define the following probabilistic model

\[
p(a | i_n, q_n) = \frac{\exp(f_\theta(i_n, q_n)^\top g_\phi(a))}{\sum_{a' \in A} \exp(f_\theta(i_n, q_n)^\top g_\phi(a'))}
\]

(11.1)

Discriminative Learning with Weighted Likelihood Given the probabilistic model, it is natural to learn the parameters to maximize its likelihood. In our work, we have found the following weighted likelihood is more effective

\[
\ell = - \sum_n \sum_{a \in \mathcal{T}_n} \sum_{d \in \mathcal{A}} \alpha(a, d) \log P(d | i_n, q_n),
\]

(11.2)

where the weighting function \(\alpha(a, d)\) measures how much the answer \(d\) could contribute to the objective function. A nature design is

\[
\alpha(a, d) = \mathbb{I}[a = d],
\]

(11.3)

where \(\mathbb{I}[\cdot]\) is the binary indicator function, taking value of 1 if the condition is true and 0 if false. In this case, the objective function reduces to the standard cross-entropy loss if \(\mathcal{T}_n\) is a singleton. However, in Section 11.2.4, we discuss several different designs.

11.2.3 Large-scale stochastic optimization

The optimization of eq. (11.2) is very challenging on real Visual QA datasets. There, the size of \(\mathcal{A}\) can be as large as hundreds of thousands\(^2\). Thus computing the normalization term of the probability model is a daunting task.

We use a minibatch based stochastic gradient descent procedure to optimize the weighted likelihood. Specifically, we choose \(B\) triplets randomly from \(D\) (the training dataset defined in Section 11.2.1) and compute the gradient of the weighted likelihood.

Within a minibatch \((i_b, q_b, \mathcal{T}_b)\) or \((i_b, q_b, \mathcal{T}_b \cup \mathcal{D}_b)\) for \(b = 1, 2, \cdots B\), we construct a minibatched-universe

\[
\mathcal{A}_B = \bigcup_{b=1}^N (\mathcal{T}_b \cup \mathcal{D}_b)
\]

(11.4)

Namely, all the possible answers in the minibatch are used.

However, this “mini-universe” might not be a representative sampling of the true “universe” \(\mathcal{A}\). Thus, we augment it with negative sampling. First we compute the set

\[
\bar{\mathcal{A}}_B = \mathcal{A} - \mathcal{A}_B
\]

(11.5)

and sample \(M\) samples from this set. These samples (denoted as \(\mathcal{A}_o\)) are mixed with \(\mathcal{A}_B\) to increase the exposure to incorrect answers (i.e. negative samples) encountered by the triplets in a

\(^2\)In the Visual Genome dataset [100], for example, we have more than 201,000 possible answers.
minibatch. In short, we use $A_0 \cup A_B$ in lieu of $A$ in computing the posterior probability $p(a|i, q)$ and the likelihood.

### 11.2.4 Defining the weighting function $\alpha$

We can take advantage of the weighting function $\alpha(a, d)$ to incorporate external or prior semantic knowledge. For example, $\alpha(a, d)$ can depend on semantic similarity scores between $a$ and $d$. Using the WUPS score [196, 124], we define the following rule

$$
\alpha(a, d) = \begin{cases} 
1 & \text{if } \text{WUPS}(a, d) > \lambda, \\
0 & \text{otherwise}, 
\end{cases}
$$

where $\lambda$ is a threshold (e.g., 0.9 as in [124]). $\alpha(a, d)$ can also be used to scale triplets with a lot of semantic similar answers in $T$ (for instance, “apple”, ”green apple”, ”small apple” or “big apple” are good answers to “what is on the table?”):

$$
\alpha(a, d) = \frac{1}{|T|} \mathbb{I}[a = d]
$$

such that each of these similar answers only contributes to a fraction of the likelihood to the objective function. The idea of eq. (11.7) has been exploited in several recent work [226, 79, 93] to boost the performance on VQA [14] and VQA2 [67].

### 11.2.5 Prediction

During testing, given the learned $f_\theta$ and $g_\phi$, for the open-ended setting we can apply the following decision rule

$$
a^* = \arg \max_{a \in A} f_\theta(i, q)^\top g_\phi(a),
$$

to identify the answer to the pair $(i, q)$.

Note that we have the freedom to choose $A$ again: it can be the same as the “universe of answers” constructed for the training (i.e., the collection of most frequent answers), or a union with all the answers in the validation or testing set. The flexibility is afforded here by using the embedding function $g_\phi$ to embed any texts. Note that in existing open-ended Visual QA, the set $A$ is constrained to the most frequent answers, reflecting the limitation of using multi-way classification as a framework for Visual QA tasks.

This decision rule readily extends to the multiple-choice setting, where we just need to set $A$ to include the correct answer and the incorrect answers in each testing triplet.

### 11.2.6 Comparison to existing algorithms

Most existing Visual QA algorithms (most working on the open-ended setting on VQA [14] and VQA2 [67]) train a multi-way classifier on top of the $f_\theta$ embedding. The number of classes are set to 1,000 for VQA [56] and around 3,000 for VQA2 [56, 226, 93] of the top-frequency correct answers. These top-frequent answers cover over 90% of the training and 88% of the training
and validation examples. Those training examples whose correct answers are not in the top-$K$ frequent ones are simply disregarded during training.

There are some algorithms also learning a tri-variable compatibility function $h(i, q, a)$ [80, 56, 168]. And the correct answer is inferred by identify $a^*$ such that $h(i, q, a^*)$ is the highest. This type of learning is particularly suitable for multiple-choice based Visual QA. Since the number of candidate answers is small, enumerating all possible $a$ is feasible. However, for open-ended Visual QA tasks, the number of possible answers is very large—computing the function $h()$ for every one of them is costly.

Note that our decision rule relies on computing $f_\theta(i, q) \top g_\phi(a)$, a factorized form of the more generic function $h(i, q, a)$. However, precisely due to this factorization, we only need to compute $f_\theta(i, q)$ just once for every pair $(i, q)$. For $g_\phi(a)$, as long as the model is sufficiently simple, enumerating over many possible $a$ is less demanding than what a generic (and more complex) function $h(i, q, a)$ requires. Indeed, in practice we only need to compute $g_\phi(a)$ once for any possible $a$.

See Section 11.3.9 for details.

### 11.3 Empirical studies

We validate our approach on several Visual QA datasets. We start by describing these datasets and the empirical setups. We then report our results. The proposed approach performs very well. It outperforms the corresponding multi-way classification-based approaches where the answers are modeled as independent ordinal numbers. Moreover, it outperforms those approaches in transferring models learned on one dataset to another one.

#### 11.3.1 Datasets

We apply the proposed approach to four datasets. Table 11.1 summarizes their characteristics. We call the revised Visual7W in Chapter 9 ascV7W, and call the multiple-choice version of Visual Genome (VG) as qaVG. Note that each $(i_n, q_n)$ pair in VQA2 is answered by 10 human annotators (i.e., $|T_n| = 10$). The most frequent one is selected as the single correct answer $t_n$.

Please see Chapter 9 for more details.

---

3The answer embedding $g(a)$ for all possible answers (say 100,000) can be pre-computed. At inference we only need to compute the embedding $f(i, q)$ once for an $(i, q)$ pair and perform 100,000 inner products. In contrast, methods like [80, 56, 168] need to compute $h(i, q, a)$ for 100,000 times. Even if such a function is parameterized with a simple MLP, the computation is much more intensive than an inner product when one has to perform 100,000 times.
Table 11.2: The answer coverage of each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of unique answers</th>
<th>triplets covered by top $K$ = 1,000</th>
<th>triplets covered by top $K$ = 3,000</th>
<th>triplets covered by top $K$ = 5,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQA2</td>
<td>22K/13K/-/29K</td>
<td>88%</td>
<td>93%</td>
<td>96%</td>
</tr>
<tr>
<td>Visual7W</td>
<td>63K/31K/43K/108K</td>
<td>57%</td>
<td>68%</td>
<td>71%</td>
</tr>
<tr>
<td>VG</td>
<td>119K/57K/79K/201K</td>
<td>61%</td>
<td>72%</td>
<td>76%</td>
</tr>
</tbody>
</table>

**Answer Coverage within Each Dataset.** In Table 11.2, we show the number of unique answers in each dataset on each split, together with the portions of question and answer pairs covered by the top-$K$ frequent correct answers from the training set. We observe that the qaVG contains the largest number of answers, followed by Visual7W and VQA2. In terms of coverage, we see that the distribution of answers on VQA2 is the most skewed: over 88% of training and validation triplets are covered by the top-1000 frequent answers. On the other hand, Visual7W and qaVG needs more than top-5000 frequent answers to achieve a similar coverage.

Thus, a prior, Visual7W and qaVG are “harder” datasets, where a multi-way classification-based open-ended Visual QA model will not perform well unless the number of categories is significantly higher (say $\gg 5000$) in order to be able to encounter less frequent answers in the test portion of the dataset—the answers just have a long-tail distribution.

### 11.3.2 Experimental setup

**Our Model.** We use two different models to parameterize the embedding function $f_{\theta}(i, q)$ in our experiments—Multi-layer Perceptron [80] (MLP) and Stacked Attention Network [209, 93] (SAN). For both models, we first represent each token in the question by the 300-dimensional GloVe vector [143], and use the ResNet-152 [73] to extract the visual features following the exact setting of [93]. Detailed specifications of each model are as follows.

- **Multi-layer Perceptron (MLP):** We represent an image by the 2,048-dimensional vector form the top layer of the ResNet-152 pre-trained on ImageNet [157], and a question by the average of the GloVe vectors after a linear transformation followed by tanh non-linearity and dropout. We then concatenate the two features (in total 2,348 dimension), and feed them into a one-layer MLP (4,096 hidden nodes and intermediate dropout), with the output dimensionality of 1,024.

- **Stacked Attention Network (SAN):** We represent an image by the $14 \times 14 \times 2048$-dimensional tensor, extracted from the second last layer of the ResNet-152 pre-trained on ImageNet [157]. See [209] for details. On the other hand, we represent a question by a one layer bidirectional LSTM over GloVe word embeddings. Image and question features are then inputed into the SAN structure for fusion. Specifically, we follow a very similar network architecture presented in [93], with the output dimensionality of 1,024.

For parameterizing the answering embedding function $g_{\phi}(a)$, we adopt two architectures: 1) Utilizing a one-layer MLP on average GloVe embeddings of answer sequences, with the output dimensionality of 1,024. 2) Utilizing a two-layer bidirectional LSTM (bi-LSTM) on top of
GloVE embeddings of answer sequences. We use MLP for computing answer embedding by default. We denote method with bi-LSTM answer embedding with a postfix \( \star \) (e.g. SAN\( \star \)).

In the following, we denote our factorized model applying PMC for optimization as \( fPMC \) (cf. eq (11.1)). We consider variants of \( fPMC \) with different architectures (e.g. MLP, SAN) for computing \( f_\theta(i, q) \) and \( g_\phi(a) \), named as \( fPMC(\text{MLP}) \), \( fPMC(\text{SAN}) \) and \( fPMC(\text{SAN}\star) \).

**Competing Methods.** We compare our model to multiway classification-based (CLS) models which take either MLP or SAN as \( f_\theta \). We denote them as \( \text{CLS}(\text{MLP}) \) or \( \text{CLS}(\text{SAN}) \). We set the number of output classes for CLS model to be top-3,000 frequent training answers for VQA2, and top-5,000 for Visual7W and VG. This is a common setup for open-ended Visual QA [3].

Meanwhile, we also re-implement approaches that learn a scoring function \( h(i, q, a) \) with its input as \((i_n, q_n, T_n)\) triplets [80]. As such methods are initially designed for multiple-choice datasets, the calibration between positive and negative samples needs to be carefully tuned. It is challenging to adapt to ‘open-end’ settings where the number of negative answers scaled up. Therefore, we adapt them to also utilize our PMC framework for training, which optimize stochastic multi-class cross-entropy with negative answers sampling. We name such methods as \( uPMC \) (un-factorized PMC) and call its variants as \( uPMC(\text{MLP}) \) and \( uPMC(\text{SAN}) \). We also compare to reported results from other state-of-the-art methods.

**Evaluation Metrics** The evaluation metric for each dataset is different. For VQA2, the standard metric is to compare the selected answer \( a^\star \) of a \((i, q)\) pair to the ten corresponding human annotated answers \( T = \{s_1, \cdots, s_{10}\} \). The performance on such an \((i, q)\) pair is set as follows

\[
\text{acc}(a^\star, T) = \max \left\{ 1, \frac{\sum I[a^\star = s_l]}{3} \right\}.
\]  

(11.9)

We report the average performance over examples in the validation split and test split.

For Visual7W (or V7W), the performance is measured by the portion of correct answers selected by the Visual QA model from the candidate answer set. The chance for random guess is 25% (or 14.3%). For VG, we focus on the multiple choice evaluation (on qaVG). We follow the settings in Chapter 9 and measure multiple choice accuracy. The chance for random guess is 14.3%.

### 11.3.3 Results on individual Visual QA datasets

Table 11.3 gives a comprehensive evaluation for most state-of-the-art approaches on four different settings over VQA2 (test-dev), Visual7W, V7W and qaVG\(^4\). Among all those settings, our proposed \( fPMC \) model outperform the corresponding classification model by a noticeable margin. Meanwhile, \( fPMC \) outperforms \( uPMC \) over all settings. Comparing to other state-of-the-art methods, we show competitive performance against most of them.

In Table 11.3, note that there are differences in the experimental setups in many of the comparison to state-of-the-art methods. For instance, MLP [80] used either better text embedding

\(^4\)The omitted ones are due to their missing in the corresponding work. In fact, most existing work only focuses on one or two datasets.
Table 11.3: Results (%) on Visual QA with different settings: open-ended (Top-$K$) and multiple-choice (MC) based for different datasets. The omitted ones are due to their missing in the corresponding work.

<table>
<thead>
<tr>
<th>Method</th>
<th>Visual7W</th>
<th>V7W</th>
<th>VQA2</th>
<th>qaVG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MC [230]</td>
<td>MC Chapter 9</td>
<td>Top-3k [67]</td>
<td>MC Chapter 9</td>
</tr>
<tr>
<td>LSTM [230]</td>
<td>55.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MLP Chapter 9</td>
<td>65.7</td>
<td>52.0</td>
<td>-</td>
<td>58.5</td>
</tr>
<tr>
<td>MLP [80]</td>
<td>67.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C+LSTM [67]</td>
<td>-</td>
<td>-</td>
<td>54.1</td>
<td>-</td>
</tr>
<tr>
<td>MCB [67]</td>
<td>62.2</td>
<td>-</td>
<td>62.3</td>
<td>-</td>
</tr>
<tr>
<td>MFB [227]</td>
<td>-</td>
<td>-</td>
<td>65.0</td>
<td>-</td>
</tr>
<tr>
<td>BUTD [10]</td>
<td>-</td>
<td>-</td>
<td>65.6</td>
<td>-</td>
</tr>
<tr>
<td>MFH [226]</td>
<td>-</td>
<td>-</td>
<td>66.8</td>
<td>-</td>
</tr>
<tr>
<td>Multi-way Classification Based Model (CLS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLS(MLP)</td>
<td>51.6</td>
<td>40.9</td>
<td>53.5</td>
<td>46.9</td>
</tr>
<tr>
<td>CLS(SAN)</td>
<td>53.7</td>
<td>43.6</td>
<td>62.4</td>
<td>53.0</td>
</tr>
<tr>
<td>Our Probabilistic Model of Compatibility (PMC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uPMC(MLP)</td>
<td>62.4</td>
<td>51.6</td>
<td>51.4</td>
<td>54.5</td>
</tr>
<tr>
<td>uPMC(SAN)</td>
<td>65.3</td>
<td>55.2</td>
<td>56.0</td>
<td>61.3</td>
</tr>
<tr>
<td>fPMC(MLP)</td>
<td>63.1</td>
<td>52.4</td>
<td>59.3</td>
<td>57.7</td>
</tr>
<tr>
<td>fPMC(SAN)</td>
<td>65.6</td>
<td>55.4</td>
<td>63.2</td>
<td>62.6</td>
</tr>
<tr>
<td>fPMC(SAN+)</td>
<td>66.0</td>
<td>55.5</td>
<td>63.9</td>
<td>63.4</td>
</tr>
</tbody>
</table>

or more advanced visual feature, which benefits their result on Visual7W significantly. Under the same configuration, our model has obtained improvement. Besides, most of the state-of-the-art methods on VQA2 fall into the category of classification model that accommodates specific Visual QA settings. They usually explore better architectures for extracting rich visual information [230, 10], or better fusion mechanisms across multiple modalities [67, 227, 226]. We notice that our proposed PMC model is orthogonal to all those recent advances in multi-modal fusion and neural architectures. More advanced deep learning models can be adapted into our framework as $f_{\theta}(i, q)$ (e.g. fPMC(MFH)) to achieve superior performance across different settings. This is particularly exemplified by the dominance of SAN over the vanilla MLP model. We leave this for future work.

11.3.4 Ablation studies

Importance of Negative Sampling Our approach is probabilistic, demanding to compute a proper probability over the space of all possible answers. (In contrast, classification-based models limit their output spaces to a pre-determined number, at the risk of not being able to handle unseen answers).

In Section 11.2.3, we describe a large-scale optimization technique that allows us to approximate the likelihood by performing negative sampling. Within each mini-batch, we create a mini-universe of all possible answers as the union of all the correct answers (i.e., $A_B$). Additionally, we randomly sample $M$ answers from the union of all answers outside of the mini-batch, creating
Table 11.4: The effect of negative sampling \((M = 3,000)\) on fPMC. The number is the accuracy in each question type on VQA2 (val).

<table>
<thead>
<tr>
<th>Method</th>
<th>Mini-Universe</th>
<th>Y/N</th>
<th>Number</th>
<th>Other</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>(A_B)</td>
<td>70.1</td>
<td>33.0</td>
<td>38.7</td>
<td>49.8</td>
</tr>
<tr>
<td>SAN</td>
<td>(A_o)</td>
<td>78.2</td>
<td>37.1</td>
<td>45.7</td>
<td>56.7</td>
</tr>
<tr>
<td>MLP</td>
<td>(A_o \cup A_B)</td>
<td>76.6</td>
<td>36.1</td>
<td>43.9</td>
<td>55.2</td>
</tr>
<tr>
<td>SAN</td>
<td>(A_o \cup A_B)</td>
<td>79.0</td>
<td>38.0</td>
<td>51.3</td>
<td>60.0</td>
</tr>
</tbody>
</table>

Figure 11.2: Detailed analysis on the size of negative sampling to fPMC(MLP) and fPMC(SAN) at each mini-batch. The reported number is the accuracy on VQA2 (val).

“an other world” of all possible answers \(A_o\). The \(A_o\) provides richer negative samples to \(A_B\) and is important to the performance of our model, as shown in Table 11.4.

We further conducted detailed analysis on the effects of negative sample sizes as shown in Fig. 11.2. With the number of negative samples increasing from 0 to 3,000 for each mini-batch, we observe a increasing trend from the validation accuracy. A significant performance boost is obtained comparing methods with a small number of negative samples to no additional negative samples. The gain then becomes marginal after \(A_o\) is greater than 2,000.

The Effect of Incorporating Semantic Knowledge in Weighted Likelihood

In Section 11.2.2, we have introduced the weighting function \(\alpha(a, d)\) to measure how much an incorrect answer \(d\) should contribute to the overall objective function. In particular, this weighting function can be used to incorporate prior semantic knowledge about the relationship between a correct answer \(a\) and an incorrect answer \(d\).

We report in Table 11.5 the ablation study on using different weight function \(\alpha(a, d)\) in the weighted likelihood formulation (cf. Eq. 11.2). We compare three different types of \(\alpha(a, d)\) on VQA2:

- **one-hot**: Denote \(t_n\) as the dominant answer in \(T_n\). We set \(T_n \leftarrow \{t_n\}\) (i.e., now \(T_n\) becomes a singleton) and apply

  \[
  \alpha(a, d) = \mathbb{I}[a = d] \quad (\text{cf. Eq. 11.3}).
  \]
Table 11.5: Detailed analysis of different $\alpha(a, d)$ for weighted likelihood. The reported number is the accuracy on VQA2 (validation).

<table>
<thead>
<tr>
<th>Method</th>
<th>Weighting Criterion</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>fPMC(SAN)</td>
<td>one-hot</td>
<td>58.0</td>
</tr>
<tr>
<td></td>
<td>multi-hot</td>
<td>60.0</td>
</tr>
<tr>
<td>WUPS</td>
<td></td>
<td>57.8</td>
</tr>
</tbody>
</table>

Table 11.6: The # of common answers across datasets (training set)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Top-$K$ most frequent answers</th>
<th>Total # of unique answers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1K  3K  5K  10K  all</td>
<td></td>
</tr>
<tr>
<td>VQA2, Visual7W</td>
<td>451 1,262 2,015 3,585 10K</td>
<td>137K</td>
</tr>
<tr>
<td>VQA2, qaVG</td>
<td>495 1,328 2,057 3,643 11K</td>
<td>149K</td>
</tr>
<tr>
<td>Visual7W, qaVG</td>
<td>657 1,890 3,070 5,683 27K</td>
<td>201K</td>
</tr>
</tbody>
</table>

In this case, only one answer is considered positive to a ($i, q$) pair. No extra semantic relationship is encoded.

- **multi-hot**: We keep the given $T_n$ (the ten user annotations collected by VQA2; i.e. $|T_n| = 10$) and apply

  $$\alpha(a, d) = \mathbb{I}[a = d] \text{ (cf. Eq. 11.3)}$$

  to obtain a multi-hot vector $\sum_{a \in T_n} \alpha(a, d)$ for soft weighting, leading to a loss similar to [93, 79].

- **WUPS**: We again consider $T_n \leftarrow \{t_n\}$, but utilize the WUPS score [196, 124] (the range is $[0, 1]$) together with Eq. 11.6 to define $\alpha(a, d)$. We set $\lambda = 0.9$ and give $d$ which has $\text{WUPS}(a, d) = 1$ a larger weight (i.e., 8).

  The results suggest that the multi-hot vector computed from multiple user annotations provides the best semantic knowledge among answers for learning the model.

### 11.3.5 Transfer learning across datasets

One important advantage of our method is to be able to cope with unseen answers in the training dataset. This is in stark contrast to multi-way classification based models which will have to skip on those answers as the output categories are selected as top-$K$ most frequent answers from the training dataset.

Thus, classification based models for Visual QA are not amenable to transfer across datasets where there is a large gap between different spaces of answers. Table 11.6 illustrates the severity by computing the number of common answers across datasets. On average, about 7% to 10% of the unique answers are shared across datasets. If we restrict the number of answers to consider to top 1,000, about 50% to 65% answers are shared. However, top 1000 most frequent answers are in general not enough to cover all the questions in any dataset. Hence, we arrive at the unexciting observation—we can transfer but we can only answer a few questions!
Table 11.7: Results of cross-dataset transfer using either classification-based models or our models (PMC) for Visual QA. \( (f_\theta = SAN) \)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Visual7W</th>
<th>CLS</th>
<th>uPMC</th>
<th>fPMC</th>
<th>fPMC*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual7W</td>
<td>53.7</td>
<td>65.6</td>
<td>66.0</td>
<td>↑</td>
<td>19.1</td>
</tr>
<tr>
<td>VQA2</td>
<td>45.8</td>
<td>56.8</td>
<td>60.2</td>
<td>↑</td>
<td>59.4</td>
</tr>
<tr>
<td>qaVG</td>
<td>58.9</td>
<td>66.0</td>
<td>68.4</td>
<td>↑</td>
<td>25.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CLS</th>
<th>uPMC</th>
<th>fPMC</th>
<th>fPMC*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual7W</td>
<td>19.1</td>
<td>18.5</td>
<td>19.8</td>
<td>↑</td>
</tr>
<tr>
<td>VQA2</td>
<td>59.4</td>
<td>56.0</td>
<td>60.0</td>
<td>↑</td>
</tr>
<tr>
<td>qaVG</td>
<td>25.6</td>
<td>23.6</td>
<td>25.8</td>
<td>↑</td>
</tr>
</tbody>
</table>

Table 11.8: Transferring is improved on the VQA2 dataset without Yes/No answers (and the corresponding questions) \( (f_\theta = SAN) \).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CLS</th>
<th>uPMC</th>
<th>fPMC</th>
<th>fPMC*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual7W</td>
<td>31.7</td>
<td>29.5</td>
<td>33.1</td>
<td>↑</td>
</tr>
<tr>
<td>qaVG</td>
<td>42.6</td>
<td>39.3</td>
<td>43.0</td>
<td>↑</td>
</tr>
</tbody>
</table>

In Table 11.7, we report our results of transferring learned Visual QA model from one dataset (row) to another one (column). For VQA2, we evaluate the open-end accuracy using top-3000 frequent answer candidates on validation set. We evaluate multiple-choice accuracy on the test set of Visual7W and qaVG.

The classification models (CLS) clearly fall behind the performance of our method (uPMC and fPMC)—the red upper arrows signify improvement. In some pairs the improvement is significant (e.g., from 42.8% to 54.8% when transferring from Visual7W to qaVG). Furthermore, we noticed that fPMC outperforms uPMC in all transfer settings.

However, VQA2 seems a particular difficult dataset to be transferred to, from either V7W or qaVG. The improvement from CLS to fPMC is generally small. This is because VQA2 contains a large number of Yes/No answers. For such answers, learning embeddings is not advantageous as there are little semantic meanings to extract from them.

We perform another study by removing those answers (and associated questions) from VQA2 and report the transfer learning results in Table 11.8. In general, both CLS and fPMC transfer better. Moreover, fPMC improves over CLS by a larger margin than that in Table 11.7.

### 11.3.6 Analysis with seen/unseen answers

To gain a deeper understanding towards which component brings the advantage in transfer learning, we performed additional experiments to analyze the difference on seen/unseen answers. Specifically, we study the transfer learning result from VQA2 and qaVG to Visual7W. Here, **seen** (S) refers to those multiple choices where at least one candidate answer is seen in the training vocabulary, and **unseen** (U) refers to those multiple choices where all the candidate answers are not observed in the training vocabulary. As shown in Table 11.9, we see that our fPMC model performs better than the CLS model on both seen and unseen answer set. While CLS model obtains random performance (the random chance is 25 %) on the unseen answer set, our fPMC model achieved at least 20% (in absolute value) better performance. In general, uPMC is also working well comparing to CLS. This performance improvement is gain mostly by taking answer semantics from the word vectors into account.
Table 11.9: Analysis of cross dataset performance over Seen/Unseen answers using either CLS or PMC for Visual QA

<table>
<thead>
<tr>
<th>Target</th>
<th>VQA2</th>
<th>Visual7W</th>
<th>qaVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>CLS(SAN)</td>
<td>uPMC(SAN)</td>
<td>fPMC(SAN)</td>
</tr>
<tr>
<td>VQA2</td>
<td>S</td>
<td>U</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>59.8</td>
<td>25.0</td>
<td>45.8</td>
</tr>
<tr>
<td>qavG</td>
<td>63.4</td>
<td>25.0</td>
<td>58.9</td>
</tr>
</tbody>
</table>

Table 11.10: Results for the baseline method that fix answer embedding as GloVe. (We show results with SAN as $f_\theta(i, q)$).

<table>
<thead>
<tr>
<th>Target</th>
<th>VQA2</th>
<th>Visual7W</th>
<th>qaVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Fixed Learning</td>
<td>Fixed Learning</td>
<td>Fixed Learning</td>
</tr>
<tr>
<td>VQA2</td>
<td>57.5</td>
<td>60.0</td>
<td>47.5</td>
</tr>
</tbody>
</table>

11.3.7 Visualization on answer embeddings

We provide the t-SNE visualization of the answer embedding. To better demonstrate the effectiveness of learning answer embedding, we re-train the answer embedding model with randomly initialized answer vectors. We provide visualization on both the initial answer embedding and learned answer embedding, to reflect the preservation of semantics and syntactics in the learned embedding.

According to Fig. 11.3, we can observe that a clear structure in the answer embedding are obtain in our learned embedding. While the random initialization of the embedding remains chaos, our learned embedding successfully provide both semantic and syntactic similarities between answers. For example, semantically similar answers such as “airplane” and “motorcycle” are close to each other, and syntactically similar answers like “in an office” and “on the porch” are close. Besides, we also observe that answers are clustered according to its majority question type, which meets our expectation for the answer embedding’s structure. Here we take majority because one answer can be used for multiple questions of different types.

11.3.8 Analysis on answer embeddings

We provide results for an additional baseline algorithm where $f_\theta(i, q)$ directly maps to the fixed space of average GloVe answer representations. Here we need to keep the GloVe embedding fixed to enable transferability. Table 11.10 shows the results on the VQA2 dataset. We compare its performance to our approach of learning answer embedding with MLP as $g_\phi(a)$ in terms of both in-domain and transfer learning performance—learning answer embeddings outperforms this simple baseline in all cases. Associated with the previous visualization results, we can conclude that learning answer embedding can effectively capture the semantic relationship between answers and image-question pairs while obtaining superior performance on both within-domain performance and transfer learning performance.
Figure 11.3: **t-SNE visualization.** We randomly select 1000 answers from Visual7W and visualize them in the initial answer embedding and learned answer embeddings. Each answer is marked with different colors according to their question types. (e.g. when, how, who, where, why, what). To make the figure clear for reading, we randomly sub-sampled the text among those 1000 answers to visualize.
Table 11.11: Efficiency study among CLS(MLP), uPMC(MLP) and fPMC(MLP). The reported numbers are the average inference time of a mini-batch of 128 (\(|T| = 1000\)).

<table>
<thead>
<tr>
<th>Method</th>
<th>CLS(MLP)</th>
<th>uPMC(MLP)</th>
<th>fPMC(MLP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (ms)</td>
<td>22.01</td>
<td>367.62</td>
<td>22.14</td>
</tr>
</tbody>
</table>

Figure 11.4: Inference time Vs. Mini-batch index. fPMC(MLP) and CLS(MLP) model are 10x faster than uPMC(MLP) (use PyTorch v0.2.0 + Titan XP + Cuda 8 + Cudnnv5).

11.3.9 Inference efficiency

Next we study the inference efficiency of the proposed fPMC, uPMC (i.e., triplet based approaches \([80, 56, 168]\) with PMC) models with the CLS model. For fair comparison, we use the one-hidden-layer MLP model for all approaches, keep \(|T| = 1000\) and mini-batch size to be 128 (uPMC based approach is memory consuming. More candidates require reducing the mini-batch size). We evaluate models on the VQA2 validation set (\(\sim 2200\) mini-batches) and report the (average) mini-batch inference time. Fig. 11.4 and Table 11.11 show that fPMC(MLP) obtains similar performance to CLS(MLP), with at least 10 times faster than uPMC(MLP).

11.4 Summary

We propose a novel approach of learning answer embeddings for the visual question answering (Visual QA) task. The main idea is to learn embedding functions to capture the semantic relationship among answers, instead of treating them as independent categories as in multi-way classification-based models. Besides improving Visual QA results on single datasets, another significant advantage of our approach is to enable better model transfer. The empirical studies on several datasets have validated our approach.

Our approach is also “modular” in the sense that it can exploit any joint modeling of images and texts (in this case, the questions). An important future direction is to discover stronger multimodal modeling for this purpose.
Part IV

Conclusion
Chapter 12

Conclusion

My thesis is towards developing intelligent systems for vision and language understanding in the wild. Many recent successes on vision and language understanding are based on statistical machine learning, which is founded on the assumption—the data of the training and test environments must be from the same feature and label spaces, and have the same distribution. This assumption, however, will not lead to systems that can perform well in the wild—to recognize object categories not seen during training (e.g., rare or newly-defined objects), and to answer unfamiliar visual questions (e.g., different language styles by users). In my thesis, I thus strive to develop transfer learning algorithms to leverage external information so that the learned models can be properly transferred and generalized across categories and users (domains).

To recognize unseen objects, we work on a transfer learning paradigm called zero-shot learning (ZSL), which aims to transfer discriminative knowledge learned from seen objects, of which we have labeled training data, to unseen ones with the help of external class semantic representations. My thesis provides a comprehensive set of insights and techniques to improve zero-shot learning (ZSL)—from effectively leveraging the semantic representations in relating classes (cf. Chapter 4), to revisiting and revising the ZSL settings towards real-world applications (cf. Chapter 5), to unifying ZSL with one-shot and few-shot learning (cf. Chapter 6), and to improving the class semantic representations by incorporating domain knowledge (cf. Chapter 7).

To answer unfamiliar questions, we work on domain adaptation, another transfer learning paradigm to match the data distributions of training (source) and testing (target) domains. We present a framework by adapting users’ language styles to what the learned Visual QA model has been trained on so that we can re-use the model without re-training (cf. Chapter 10). My thesis further revisits and revises existing datasets (cf. Chapter 9) and introduces a probabilistic and factorization framework to leverage answer semantics (cf. Chapter 11), providing a series of analysis and techniques to improve knowledge transfer across domains for Visual QA.

12.1 Remarks on future work

In this section, I will discuss my plans for future research towards advanced transfer learning. For the short term, I hope to advance zero-shot learning by exploring more informative class semantic representations and establishing the theoretical foundation. On the other hand, I want to improve the performance of visual question answering, and generalize the solutions to and benefit general AI tasks. For the long run, I will strive to unify different concepts and algorithms
of transfer learning and advance the area with brand new thinking by introducing more principled frameworks and approaches instead of relying on ad hoc decisions or domain-specific insights.

12.1.1 Advanced zero-shot learning

The performance of ZSL algorithms largely depends on the quality of class semantic representations. Human-annotated attributes (e.g., colors, shapes) convey more domain (e.g., visual) information than word vectors and lead to better performance. However, they are costly to define and acquire, especially on a massive number of categories. I will develop algorithms to automatically mine attributes from documents that are able to faithfully describe and discriminate a large amount of categories. Moreover, I will develop algorithms that can incorporate multiple sources of semantic representations even beyond the vectorized forms, e.g., taxonomies, knowledge graphs and bases, so as to take the best advantage of available information.

Besides algorithm design, I hope to establish the theoretical foundation that so far has been missing in zero-shot learning. Theories underlying other transfer learning paradigms such as domain adaptation have been gradually developed and provide novel insights to inspire algorithm design. For example, the idea of adversarial domain adaptation [58] has been indicated in the theories by Ben-David et al. [17]. I believe that similar benefits can be brought to ZSL. While theories are usually founded on simple models (e.g., linear models or nearest neighbors), the fact that simple models do work for ZSL [26, 28, 27] suggests their immediate applicability and impact once being developed. As a starting point, I am dedicated to establishing the theoretical guarantee—under what relationships between the data and class semantic representations the ZSL algorithms will succeed or fail—to guide exploring better semantic representations.

12.1.2 Advanced transfer learning for AI

My future plans on transfer learning for AI tasks are three-folded. First, I notice that Visual QA datasets that use real images and collect questions and answers from humans usually end up with little need of reasoning (e.g., “who is in the image?”). On the other hand, datasets that synthesize images and questions often require higher-level of reasoning (e.g., “what is the color of the big ball to the left of the small red triangle?”). I will thus develop transfer learning algorithms that takes advantage of the synthetic datasets to learn high-level reasoning for the real environments.

Secondly, I plan to extend the insights in advancing datasets and cross-dataset transfer for Visual QA to other AI tasks that involve information of multiple modalities. For example, the above line of research has potential to benefit autonomous driving and reinforcement learning, in which collecting real data is extremely costly. Moreover, I hope to awake the community’s attention in systematic and rigorous design of datasets and evaluation metrics so that the research progress will not deviate from practice.

Finally, I hope to build up powerful AI systems to help advance other vision tasks that indeed require human studies for evaluation. One exemplar is video summarization—a good video summary should conveys sufficient information of the original video. We can apply learned Visual QA models to the summary, together with questions relevant to the original videos, to judge the quality of the summary, replacing the currently used metrics that simply measure visual or temporal overlaps to the human-generated summary.
12.1.3 Principled frameworks for transferable machine learning

While transfer learning has been studied over decades, it still remains as a fundamental challenge of machine learning and without proper organization. Different transfer learning paradigms, such as zero-shot learning, few-shot learning, domain adaptation, and multi-task learning, are usually developed independently, lacking strong connections to inspire and benefit each other. On the other hand, learning-based models, instead of rule-based ones, are believed to be the most promising ways to develop intelligence systems for real-world applications, built upon the fact that they have achieved human-level performance in many constrained tasks of computer vision and natural language processing. To effectively bring the in-laboratory success to reality, we need to advance transfer learning with brand new and more principled thinking to account for various types of mismatch in the environment.

From my point of view, the core of transfer learning is to identify “why”, “what”, and “how” to transfer. “Why” corresponds to different situations of environmental mismatch. “What” corresponds to different sources that can be transferred, ranging from data to feature representation [139] to models (architectures and parameters) [158, 11] and to meta information like initialization, hyper-parameters [49], or even the optimizers [12]. Note that these sources are where algorithms of different paradigms share common concepts. Finally, “how” corresponds to the ways to execute the transfer, in which the standard one is to determine a static objective and optimize it on currently available data. While increasing effort has been put into the community, so far it is we humans that identify the combination of “why”, “what”, and “how” for the problem at hand, mostly according to domain-specific insights or even ad hoc decisions. Can we have a principled transfer learning framework in which a machine can automatically and dynamically adjust the combination and its model complexity by interacting with the real-world environment?

I plan to approach this goal via Bayesian approach. As mentioned earlier, Bayesian approach can intrinsically incorporate prior knowledge (i.e., belief) and uncertainty—corresponding to the past experience and the current interaction with the environment, respectively. More concretely, I will incorporate and take advantage of the streaming [77, 23], nonparametric [22], and hierarchical [177] methods of Bayesian approach for transfer learning.

- **Streaming**: Streaming (or more broadly stochastic and online) methods allow Bayesian approach to dynamically update the posterior with respect to the current environment and provide the up-to-date belief for the future one.

- **Nonparametric**: Nonparametric methods can automatically adjust the model complexity with respective to the data and environment the machine encounters.

- **Hierarchical**: Different sources in “what” to transfer can be arranged into a hierarchy—data and feature representation at the bottom, models at the middle, and meta-information at the top.

By treating the sources in “what” to transfer as nodes in a probabilistic graphical model, together with the streaming and nonparametric methods, Bayesian approach can systematically achieve automatic and dynamic transfer learning—(1) The observability, space, and distribution of data (and the corresponding labels) can identify “why” to transfer. (2) The likelihood on the data and labels can tell the uncertainty of them, providing cues to adjust the objective (i.e., “how” to transfer) so as to focus on hard instances. (3) Given the data, we can infer and update the
relationship of nodes. The posterior probability on variables of each node given the data in a new environment can indicate the transferability of the corresponding source, making identifying “what” to transfer an automatic data-driven process without ad hoc decisions.
Part V

Bibliography
Bibliography


References:


