Towards Interpretable Vision Systems

Peng Zhang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Devi Parikh, Chair
Jia-Bin Huang, Chair
Harpreet S. Dhillon
Bert Huang
Douglas Summers-Stay

October 16, 2017
Blacksburg, Virginia, USA

Keywords: failure prediction, portfolio vision system, interpretable vision systems

Towards Interpretable Vision Systems

Peng Zhang

ABSTRACT

Artificial intelligent (AI) systems today are booming and they are used to solve new tasks or improve the performance on existing ones. However, most AI systems work in a black-box fashion, which prevents the users from accessing the inner modules. This leads to two major problems: (i) users have no idea when the underlying system will fail and thus it could fail abruptly without any warning or explanation, and (ii) users' lack of proficiency about the system could fail pushing the AI progress to its state-of-the-art. In this work, we address these problems in the following directions. First, we develop a failure prediction system, acting as an input filter. It raises a flag when the system is likely to fail with the given input. Second, we develop a portfolio computer vision system. It is able to predict which of the candidate computer vision systems perform the best on the input. Both systems have the benefit of only looking at the inputs without running the underlying vision systems. Besides, they are applicable to any vision system. By equipped such systems on different applications, we confirm the improved performance. Finally, instead of identifying errors, we develop more interpretable AI systems, which reveal the inner modules directly. We take two tasks as examples, words semantic matching and Visual Question Answering (VQA). In VQA, we take binary questions on abstract scenes as the first stage, then we extend to all question types on real images. In both cases, we take attention as an important intermediate output. By explicitly forcing the systems to attend correct regions, we ensure the correctness in the systems. We build a neural network to directly learn the semantic matching, instead of using the relation similarity between words. Across all the above directions, we show that by diagnosing errors and making more interpretable systems, we are able to improve the performance in the current models.

Towards Interpretable Vision Systems

Peng Zhang

GENERAL AUDIENCE ABSTRACT

Researchers have made rapid progresses in artificial intelligence (AI). For example, AI systems were able to reach new state-of-the-art performance on object detection task in computer vision; AI systems were able to play games themselves, such as Alpha GO, which was never happened before. However, most of the AI systems work in a black-box fashion, which prevents users from accessing the inner modules. This could result in two problems. On one hand, users do not know when the underlying systems will fail. For example, in object detection task, users have no idea when the system could not recognize a cat in a cat image or when the system will recognize a dog as a cat. On the other hand, users have no access on how the system work, so it is hard for them to find the bottle neck and improve the overall performance. In this work, we tackle the above problems in two broad directions: diagnosing the errors and making interpretable systems. The first one can be addressed in two ways: identifying the erroneous inputs and identifying the erroneous systems. Thus, we build a failure prediction system and a portfolio computer vision system, respectively. Failure prediction system could raise a warning when the input is not reliable, while the portfolio system could pick predicted best-performing approach from candidates. Finally, we focus on developing more interpretable AI systems, which reveal the inner modules directly. We take two tasks as examples, words semantic matching and Visual Question Answering (VQA). VQA system produces an answer upon given image and question. We take attention as the important intermediate output, which mimics how humans solve this task. In semantic matching, we build a system to learn the semantic matching between words, instead of using the relation similarity between them. In both directions, we show the improved performance in a variety of applications.

Dedication

To my parents, Weizhen Zhang and Shufang Shan.

Acknowledgments

Firstly, I would like to say a big thank to my advisor, Prof. Devi Parikh, for the support and guidance during these four years. She showed me what real research should look like and taught me how to perform it. I could not remember how many times she helped me develop my half-baked ideas, clarify my confusion, strength my skills, to make me a competitive researcher. Her optimism and confidence in me made me not fear anything. She was the leader, walked me through the dark. Because of her, I really enjoyed this journey. It was my great honor to have her as my advisor. I was lucky enough to work close with Prof. Dhruv Batra. He is humorous, and very smart as well. He always proposed insightful ideas to my projects, making me dive deep. Thanks for doing that.

I had the opportunity to do my first internship at Army Research Lab, where I met my mentor Dr. Douglas Summers-Stay. I would like to thank him for introducing me to a new research field, natural language processing and helping me get familiar with it. I also want to express my appreciation to Dr. Leonid Sigal, who was my mentor at Disney Research in my second internship. His thought helped me make great progress in my research.

I would like to thank Prof. Jia-Bin Huang, Prof. Bert Huang and Prof. Harpreet Dhillon for serving in my committee. Without their valuable suggestions, this dissertation would not be such comprehensive at all.

During my PhD studies, I made a lot of friends, who helped me in various ways. I would like to thank Hao Zhang, Run Yu and Ji Wang, with whom we spent great time together in

Blacksburg. I was also fortunate enough to meet a bunch of smart guys in my internship in Pittsburgh, Zhiwei Deng, Fanyi Xiao, Zhiding Yu, Tong Wang, Ming Sun and Keyi Huang. Without them, I could not had such a great experience there. CVMLP lab at Virginia Tech has always been a family to me and I owe thanks to all my family members: Xiao Lin, Qing Sun, Jiasen Lu, Jianwei Yang, Stanislaw Antol, Ramakrishna Vedantam, Arjun Chandrasekaran, Ramprasaath Selvaraju, Yash Goyal and Aishwarya Agrawal. I miss the old days when we hanging out, discussing projects together. All these are going to be precious memory in my whole life.

Last but not the least, I have a very warm and loving family. Since my childhood, my parents taught me to be diligent, curious and confident, which played quite important roles in my life. Their love and support enabled me to chase my dream and achieve what I have. I want to express my biggest appreciation and thanks to them.

Contents

List of Figures

Chapter 1

Introduction

Artificial intelligent (AI) systems today are developing fast, because more and more techniques have been proposed to address new tasks. For example, Alpha Go [25], a computer program to play the game Go, is able to beat the human champion. Uber has developed self-driving cars, and they have been used in real life in San Francisco and Pittsburgh. Alexa, which has been applied in several Amazon products for years, is able to help people in real life. On the other hand, AI is also used to achieve a better performance on some standard public datasets, e.g. researchers were able to improve the classification accuracy on PASCAL 2012 dataset from 61.1 % to 86.5% [16]. On an even harder dataset, ImageNet datase [27], which contains 1000 object categories, 73.14 % detection accuracy was achieved [15]. Both of the two directions are promising and encouraging. While they are the primary focus of the community, reasoning how and why the systems work have been mostly ignored.

In most cases, the systems work in a black-box fashion, which take in inputs and output the predictions. For example, in image classification task, the system takes in an image as input and produces the most probable class label as output. While the prediction might be correct or not, we have no clue on why such results are given, since we have no access to the inner modules. This leads to two major problems. Firstly, we have no idea when the system will fail, i.e. making an incorrect prediction. Secondly, this might hinder further improvement

of the underlying system. Therefore, revealing the modules inside the systems and making them more interpretable can be very important.

To address this problem, we follow the directions as below:

(1) **Failure prediction**. Computer vision systems today fail frequently. They fail abruptly without warning or explanation. Even if the system has a high confidence about its prediction, the result might be still incorrect, which could be problematic. In addition, most computer vision systems work in a pipline fashion, in which the output from previous step is fed as input to the next. For example, the prediction from attribute classification can then be used to do object classification, image segmentation results can then be used to do robot path planning. Hence, any errors in the upstream applications could be propagated to the downstream ones. Thus, it would be very beneficial if we could identify them.

We introduce a solution ALERT, which is a surprisingly straightforward and general approach. It is used to predict the accuracy (or failure) of computer vision systems by only accessing to the input images. We apply the system onto a variety of applications: semantic segmentation, vanishing point and camera parameter estimation, and image memorability prediction. To evaluate the performance, we promote two metrics, Accuracy vs Declaration Rate (ADR) and Risk Averse Metric (RAM). In addition, we leverage ALERT to improve the performance of a downstream application of attribute prediction: zero-shot learning. We use our failure prediction system to predict which attribute classifier might not be trustworthy, so that at test time, we only use the ones that are predicted to be reliable.

(2) **Portfolio vision systems**. Multiple methods have been proposed to solve the same perception task. For example, [18,110,156] have been proposed to solve the image segmentation task, similarly, [66, 93, 113] are used to address object detection problem. To compare the performance, these methods are evaluated on the same benchmark datasets, i.e. PASCAL 2012 dataset. Evaluation metrics typically compute the average performance of a perception system across all test instances. Systems that have the highest (average) accuracy attract the most attention from the community, while others are ignored. However, recall that met-

rics typically only represent the average performance of each system. Multiple methods can be complementary. The method that performs the best on average need not to be the best on each input image. Different methods may be better suited for different input instances. Thus, given the inputs and candidates methods, if we were able to choose an input-specific approach, we should be able to reach a higher overall performance than each system individually. We propose PORTFOLIO vision system, which is able to automatically predict the best-performing approach for each input instance. It has the benefit that it only needs to analyze the input images, without ever running the underlying vision systems. We take two vision tasks as examples to show the effectiveness of our system, image segmentation, and saliency detection. For each of them, 4 candidate methods are provided.

(3) **Building interpretable intelligent models**. Neural networks today are widely applied in various tasks because of their great performance. However, due to their black-box setting, it is hard to find which module is the bottleneck in improving the overall performance. Therefore, we focus on two models towards more interpretable systems: one for words semantic matching and the other for Visual Question Answering (VQA).

Words semantic matching is the task that given two pairs of words, the machine would try to predict if they have the same relation or not. For example, two pairs of words <Beijing, China> and <Berlin, Germany> should be predicted as the same relation ("capitol-of"), while <cars, car> and <yen, Japan> should not (one is "pluralisation" and the other is "currency-of"). A well known approach to address this problem is word2vec model [101]. The ability to solve such analogies reliably allows us to apply a relation to a new input, a key problem in reasoning about knowledge graphs. While the simple arithmetic in the distributional vector space can handle a surprising variety of relations, relations without a one-to-one correspondence between the terms (such as the hyponym relation) require different techniques. For example, two pairs of words <Spoon, Utensil> and <Strawberry, fruit> has a "one to many" relation, this cannot be solved by simply arithmetic operation in word2vec space, except we set hand-crafted rules. We would like to build a general system to handle all kinds of relations but deal with each of them in a different way automatically.

VQA is a new AI-complete task that attracts more and more attention from the community. VQA system takes in an image and a natural language question as inputs, and outputs an answer.

The complex compositional structure of language makes problems at the intersection of vision and language challenging. But language also provides a strong prior that can result in a good superficial performance, without the underlying models truly understanding the visual contents. This can hinder progress in pushing state-of-the-art in the computer vision aspects of multi-modal AI.

We start by balancing the dataset on binary questions, where the questions are either "yes" or "no". For each question and image pair, we collected a complementary image, which has the similar scene as the original image, but on the same question, the answer is opposite, i.e. "yes" to "no" or "no" to "yes". So for each question, we have two scenes, with the answer is "yes" for one scene and "no" for the other one. By doing that, the language bias can be eliminated.

We take abstract images as the first step. These images were created from 150 clipart objects, with either "park" or "living room" scene background [4]. Abstract scenes play two roles: (1) They allow us to focus on the high-level semantics of the VQA task as opposed to the low-level recognition problems; (2) They provide us the modality to *balance* the dataset such that language priors are controlled, and the role of vision is essential.

We formulate this problem as *visual verification* of concepts inquired in the questions. Specifically, we convert the question to a tuple that concisely summarizes the visual concept to be detected in the image. If the concept can be found in the image, the answer to the question is "yes", and otherwise "no". For example, we can extract the tuple from the question "Is the woman holding the door?" as "primary object = woman, relation = holding, secondary object = door". Then we specifically extract image feature based on the primary and secondary objects. Only if we find the concept ("woman holding door") displayed in the image, a "yes" will be output as an answer, otherwise "no" will be given.

Next, we extend to work from binary questions to all kinds of questions, and from abstract images to real images. Attention-based models on VQA have been proved to be successful recently, which achieved high accuracy on VQA challenge. They learn to attend image regions based on question and image regions compatibility. This indeed shows the inner modules of the systems (from attention masks), revealing which image regions are fired by the given questions, but this schema has a major flaw. Across all the models, most attention masks are learned in an unsupervised learning setting, so that sometimes the system might not attend to the correct regions to answer the questions. For example, given the question: "What is the boy holding?", we would like the system to focus on the boy's hand, rather than some image region, i.e. sky, irrelevant foreground objects etc, so that the question can be correctly answered. In adapting to that, we use the collected human attention as an additional supervision, which allows us to bias attention regions towards those utilized by humans for the same task.

Therefore, we propose a novel loss, which combines the traditional VQA cross-entropy discriminative loss, as well as the attention loss. However, the human attention ground truth only contains a subset of the whole dataset, we cast this as a semi-supervised learning task, where the attention loss is considered only when the ground truth is available.

This dissertation is organized as below: failure prediction and portfolio vision systems are detailed described in chapter 2 and 3, respectively. Chapter 4 talks about how we access attention in binary VQA. The extended work for VQA is introduced in chapter 5. Lastly, the models for words semantic matching are described in chapter 6.

Chapter 2

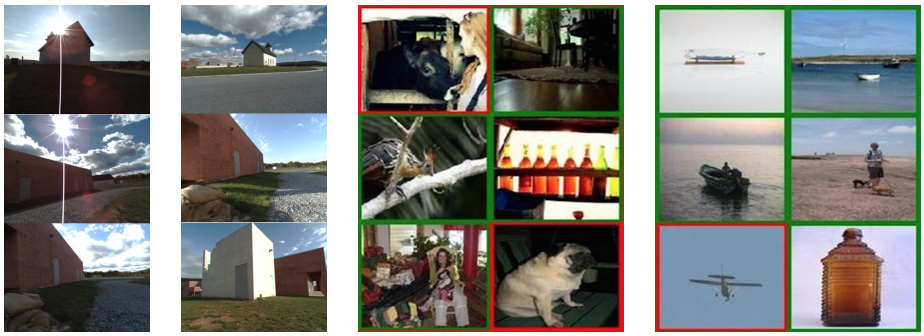
Failure Prediction

2.1 Introduction

Computer vision systems today are not perfect. Unfortunately, given the ambiguous nature of many vision problems, they will *never* be perfect. Our community's primary focus has been on making these systems – let's call them BaseSys[1] – more and more accurate. To encourage systematic progress, we have established benchmarks like Caltech 101 [41], PASCAL [36], SUN [146], etc. and we strive to minimize the failures of BaseSys on these benchmarks.

Computer vision as part of a system: It is crucial to keep in mind that in the real world, many applications involve *pipelines*, where the output of one system is fed into another as input. For instance, models trained to classify local image patches may be fed into probabilistic models for semantic segmentation [128]. Semantic segmentation may be fed to a robot's path planning algorithm for navigating through its environment [106]. Estimates of vanishing points in an image may be fed into a 3D layout estimation algorithm [61, 121]. Attribute predictions may be fed to recognition systems to classify previously unseen object

[1] BaseSys may be a segmentation engine, an attribute predictor, a vanishing point estimator, an iPhone app that predicts aesthetic quality, etc.



(a) (b) (c) (d)

Figure 2.1: Qualitative results of our proposed approach (ALERT) on two segmentation datasets: (a) and (c) show images predicted by ALERT as unreliable poor performing images; (b) and (d) show images predicted to be the best performing. For a ground robot (a,b), ALERT correctly flagged the images that are corrupted with strong shadows and over/under-exposure while retaining the images with good contrast. For PASCAL segmentation (c,d), the images for which BASESYS [14] generated poor segmentations are marked by a red border. ALERT clearly favored images with easy to segment isolated objects over complex images containing a large number of small regions.

categories [40, 80]. The subsequent "system" may even be a user: an image memorability predictor [67] may be used by a graphics designer.

In all these cases, it is of course desirable for BASESYS to fail infrequently. But arguably, it is equally desirable for BASESYS to fail *gracefully*, instead of failing abruptly without warning. While minimizing failures has been the primary focus of the community, embracing and effectively dealing with the failures has been mostly ignored. For a variety of reasons such as verifiable safety standards for autonomous systems, there is a recent push towards BASESYS that can explain their outputs and are interpretable. We take a small step in this direction.

We push for a capability where BASESYS can generate a warning "I am unable to make a reliable decision for this input". Not all applications may have the luxury of taking advantage of such a warning. We argue that *many* do. For instance, if BASESYS issues a warning that it

is unlikely to be able to accurately predict the presence of certain attributes in an image, the recognition system downstream can choose to ignore these unreliable attributes, and only use the reliable ones. This will make the overall recognition pipeline more robust as compared to one that accepts all attribute predictions from BaseSys at face value and propagates the errors. Consider a robot segmenting every frame in the video feed during a field experiment, where it often encounters poor quality images. If BaseSys can identify such frames where the segmentation output is likely to be inaccurate (See Figure 2.1(a)), the robot can choose to skip those frames all together, thus saving computational time, and simply wait for a later, more reliable frame. Depending on the application, a delayed but accurate decision may be preferred over a timely inaccurate decision, which may be catastrophic. If the user of an app can be warned that the output cannot be trusted for a particular input, (s)he can make an informed decision. In image search if a user is looking for "black high-heeled shiny shoes", it might be better to not return images where the "high-heel" predictor can not be trusted, and only return images where *all three* attributes can be reliably detected. Generally, the resultant higher precision may be worth the possibly lower recall.

How to detect failures? There are two ways to equip a BaseSys with such a warning capability. First, we could analyze the output of the vision system to assess its confidence on any given input. In this case, the confidence evaluation has to be designed specifically for each system. Instead, we propose Alert, which follows the alternative approach of evaluating the *input* itself in order to assess BaseSys's confidence. The main thesis of our approach is that while BaseSys may not be able to reliably predict the correct output for a certain input, predicting the *difficulty* or *ambiguity* of the input may still be feasible. Furthermore, this approach is applicable to *any* vision system because we generate the reliability estimate based on the input alone, without ever looking at the inner workings of BaseSys. Our approach has the added benefit of not having to run BaseSys – typically computationally expensive – to assess its confidence. Instead we can quickly reject inputs that are not reliable to begin with. Failure patterns may be different across domains. Designers of BaseSys can not foresee all scenarios in which it will be used. Hence, any confidence estimation or early-

rejection mechanisms built into BaseSys may not generalize. Alert provides a simple tool for the consumer of BaseSys to train a failure predictor catered to his domain.

Why is it feasible? The idea of analyzing the input to classification and detection algorithms to assess its suitability is not new and has been around in a variety of application-driven communities such as Automatic Target Recognition (ATR) [81, 145] and biometrics [54, 133]. In these signal and image processing fields, the distinction between errors due to inherent ambiguities in the feature space and errors due to corrupted input becomes critical. While both may be difficult to recover from, the latter can at least be detected. As the vision community inches closer to having real-world applications and away from hand-curated datasets such as Caltech 101 [41] or PASCAL [36], we argue that recognizing this distinction becomes critical in our community too. Note that, while analyzing performances of algorithms as a function of biases in these datasets has been addressed [74,136], we are advocating the distinct task of *predicting* the performance of a system on a given input. With this work we hope to bring the community's attention to building self-evaluating systems that can reliably predict their own failures.

A valid concern a reader may have is that if a system that can reliably predict failures of BaseSys can be trained, does that not mean BaseSys could have already been trained to be better? Assuming BaseSys has been trained well, should it not be impossible to train Alert with accuracy better than chance?[2] We argue that this reasoning is not applicable to a wide range of vision systems. First, many BaseSys are not learning-based approaches in the first place (e.g. vanishing point estimation as in [82]). Among those that are, applications where the output space is very large (e.g. exponential for segmentation or any structured output task), Alert identifying that the label predicted by BaseSys is incorrect provides little information about what the correct label in fact is. Further, the features used to train BaseSys (e.g. local image patches for segmentation) may be complementary to the features used to train Alert (e.g. image-level gist). It is not trivial to incorporate these features

[2]See [111] for a formal argument on "biometric-completeness".

(e.g. gist) into BaseSys itself (e.g. vanishing point estimation or segmentation). Hence, generally speaking, research efforts towards systems such as Alert are orthogonal to efforts towards improving BaseSys. For applications such as attribute-predictors where BaseSys has a small label-space (binary) Alert may boil down to identifying images where the response can not be trusted (e.g. gray-scale images for predicting the attribute red), and not images where the response can be confidently classified as being wrong. This is a subtle but important difference. While a "good" classifier should ideally incorporate this, most existing training procedures do not. They optimize for assigning correct labels to training images, and not for the classifier's confidence being correlated with likelihood of failure. Alert provides a way to deal with the fact that discriminative classifiers tend to be overconfident.

In order to demonstrate the broad applicability of our approach we evaluate it on four diverse state-of-the-art BaseSys: attribute predictors (4 datasets), semantic segmentors (2 datasets), vanishing point estimators (2 datasets) and image memorability predictor (1 dataset). In all these cases, we show that Alert can predict the accuracy of these diverse systems with significant reliability: Alert improves the accuracy of BaseSys by allowing it to make fewer, but more accurate decisions. We introduce two metrics for evaluating failure prediction approaches, and show that Alert outperforms state-of-the-art systems in terms of these metrics. Finally, we use Alert on attribute-predictors on 4 datasets and outperform strong baselines at the conventional classification accuracy metric on a downstream task: zero-shot learning.

2.2 Related Work

Our work addresses an issue that is critical for real applications and has received attention in various communities.

Meta-recognition: Inspired by meta-cognition "knowing about knowing" [44], Scheirer *et al*. [126] coined the term "meta-recognition" for performance prediction methods that ana-

lyze post-recognition scores [141]. These methods have mainly been explored in biometrics with the recent exception of [126]. The analysis proceeds by examining the output of a matching system on a test instance. This output often consists of the test instance's similarity score to all instances in the dataset. The analysis can be used to automatically determine thresholds to make match / non-match decisions [126], intelligent fusion schemes [123, 124], etc. ALERT differs from this line of work in two prominent ways (1) ALERT does not require the output of BASESYS on a test instance to predict its performance – it only uses the input test instance itself. This is particularly desirable when BASESYS is computationally expensive. (2) Methods such as [125, 126] are only applicable to BASESYS that rely on similarity scores for recognition. Many computer vision systems (such as segmentation, vanishing point estimation, etc.) do not fall in this category. ALERT is broadly applicable to all such applications. In computer vision, Welinder *et al.* [144] and Aghazadeh and Carslsson [3] predict the global performance of a classifier on a corpus of test instances. ALERT instead provides an instance-specific reliability measure.

Image quality assessment: The biometrics community has been using image quality measures as predictors for matching performance [54, 133]. In this context, the concept of input quality is closely tied to BASESYS – poor quality simply refers to the inability of BASESYS to provide accurate results for that input. Our work follows the same philosophy. Such methods, as do we, only analyze the input instance (e.g. fingerprint scan) to assess the likely quality of match. We argue that a similar level of self-evaluation should be a part of all computer vision systems as they permeate a variety of real-world applications.

Predicting failures: Optimization algorithms can often detect when they have found the global optimum [131] or can indicate for which variables they are failing to find optimal assignments [58, 117]. In computer vision, Jammalamadaka *et al.* [68] recently introduced evaluator algorithms for human pose estimators (HPE). They used features specific to this application; ALERT on the other hand uses generic image appearance features for a wide variety of applications. While application-specific features can only boost ALERT's performance, we hope to promote this line of work by demonstrating the ease with which one can

build self-evaluating systems. Aodha *et al*. [95] trained a classifier that can select one of several optical flow algorithms for a given input video sequence, which resulted in improved performance over any one algorithm alone. Liao *et al*. [89] use application-specific features to train a classifier that can predict whether their region proposal algorithm would fail on an image or not. Methods that predict performance by analyzing statistics of the training *and test* data [11,142] are not applicable to our goal of predicting the reliability of individual test instances. Detecting errors has received a lot of attention in speech recognition [21,120]. Recovering from these errors in spoken dialogue systems is often interactive [43]. In a similar spirit, a system like ALERT can be used to actively elicit labels for instances likely to be misclassified as in [9] to improve performance. KWIK "Knows What It Knows" frameworks [87] allow for active exploration which is beneficial in reinforcement- and active-learning problems. Hoiem *et al*. [64] focus on analyzing the different sources of error in object detectors, and do not predict failure. [152,153] predict anomaly activities in network.

Estimating confidence of classifiers: The confidence of a classifier in its decision is often correlated to the likelihood of it being correct. Reliably estimating the confidence of classifiers has received a lot of attention in the pattern recognition community [35,78,104]. Applications such as spam-filtering [26], natural language processing [7,33], speech [69] and even computer vision [155] have leveraged these ideas. However, different classifiers require different methods to estimate their confidences, making their general applicability limited. Moreover, many computer vision systems (e.g. vanishing point estimators) are not classifiers. Hence, system- and application-specific means of estimating the confidence of BASESYS would be required. For instance, Haeusler *et al*. [56] use an ensemble of classifiers to combine different stereo matching confidence estimates. ALERT is BASESYS independent, making it broadly applicable. Methods that use cross-validation to select a robust model from a pool of models [12] completely discard unreliable models from being used in the future. ALERT makes local instance-level decisions about whether BASESYS can be trusted for *that* instance or not.

Rejection ("none-of-the-above"): Refusing to make a decision on an instance is related

to systems that identify an instance as belonging to none of the classes seen during training [29,83]. The scenario we consider is not concerned with unfamiliar instances arising from discrepancies in the distributions of the training and test data. It is concerned with familiar but difficult instances. Gao and Koller [51] learn a relaxed hierarchy of binary classifiers, where the difficult categories may be ignored early on and are dealt with further down in the hierarchy. This allows them to tradeoff speed for accuracy. The work of Deng *et al.* [28] makes decisions in a hierarchical fashion. Given a semantic hierarchy on the categories of interest (e.g. WordNet [103]), their approach trades off specificity for accuracy. They output the most specific label they can while being confident about it. ALERT is complementary to these approaches and trades off the declaration rate of BASESYS for its accuracy.

2.3 Approach Overview

Given a vision system BASESYS, we wish to learn ALERT: a model that will detect with some reliability whether BASESYS is likely to fail on a given test instance.

As training data, we are given N instances $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_N\}$ and the corresponding accuracies (or measure of error) $\{y_1, \ldots, y_i, \ldots, y_N\}$ of BASESYS on these instances. These instances $\boldsymbol{x}_i$ may be images, video frames, entire videos or outputs of any sensing modality – whatever it is that BASESYS takes as input. The accuracy (or error) $\{y_i\}$ may be any metric that is appropriate to evaluate the output of BASESYS for a given input instance $\boldsymbol{x}_i$. In this work we only consider scenarios where y_i is a scalar, but extensions to vector y_i are conceivable. If BASESYS is a classifier that predicts a binary label for a given input instance, y_i may be the 0-1 loss. If BASESYS produces a semantic segmentation for an image $\boldsymbol{x}_i$, y_i may be the proportion of pixels in $\boldsymbol{x}_i$ correctly assigned to their ground truth label. We represent each instance $\boldsymbol{x}_i$ with $d-$dimensional appearance features: $\boldsymbol{x}_i \in \mathcal{R}^d$ (overloading notation). These appearance features may be any features that are appropriate for the modality of $\boldsymbol{x}_i$. ALERT is a discriminatively trained classifier when y_i is binary (for instance when BASESYS

produces binary labels for $\boldsymbol{x}_i$), or a regressor when y_i is a continuous scalar.

Note that any sophisticated image descriptors, machine learning techniques and application- or BaseSys-specific features can be incorporated in Alert. For instance, if one anticipates specific failure modes of BaseSys given knowledge about its inner workings, one could design features that specifically represent these failure modes.

BaseScore: BaseSys often produce indicators of their confidence in their output e.g. the energy of a conditional random field model for semantic segmentation or the number of lines detected in an image that are poorly explained by the estimated vanishing points. We call such measures BaseScore. Not only is BaseScore BaseSys-specific (unlike Alert), it also requires us to run BaseSys at test time – which may be computationally expensive – and obtain its output before we can estimate the confidence. Our approach needs to run BaseSys only once to train Alert. At test time, Alert does not need to run BaseSys to estimate reliability of the input instance. BaseScore may be quite reliable since it is aware of the inner workings of BaseSys. Alert on the other hand has the benefit of capturing potentially orthogonal image features. Which one performs better may depend on the application. But note that BaseScore is less general than Alert and not preemptive – it does not allow for early rejection of unreliable input instances. If early rejection is not critical for an application, combining Alert with BaseScore (which involves running BaseSys) can yield a better failure predictor than both.

In Section 2.4 we describe how we use Alert to avoid failures in a variety of applications. In Section 2.5 we use Alert to predict the failure of attribute predictors at transferring knowledge to previously unseen categories. This allows us to improve the performance of a downstream application (zero-shot learning [80]) over strong baselines.

2.4 Avoiding Failures

We applied our approach to three diverse applications. For all three, ALERT was trained by combining multiple kernels for 14 generic image features such as dense SIFT, color, texton, gist, HOG, line histograms, local binary patterns, self-similarity and so on, using the implementation of [146]. In other words, the feature representation for $\boldsymbol{x}_i$ is the same for all three applications. We use half the images to train ALERT and the other half to test. We now provide the specific details of BASESYS, y_i and BASESCORE. Quantitative results on these applications are in Section 2.4.4.

2.4.1 Semantic Segmentation

We explored semantic segmentation in two domains. **Robot:** The first involves segmenting images collected by a robot. The original 1278x958 images were scaled and rectified to 320x240. Each pixel was assigned to one of 8 semantic labels (sky, tree, asphalt, grass, building, object, concrete and gravel) using the semantic segmentation algorithm of Munoz *et al*. [105] as BASESYS, which constructs a hierarchical segmentation of the image and predicts a distribution of labels for each segment by combining, at each level of the hierarchy, image features with the distribution of labels predicted at the previous level. We experimented with two different accuracy measures y_i. The first is the proportion of pixels assigned to their correct labels and the second is the proportion of pixels from each class assigned to their correct labels averaged across the classes. While the first measure favors classes that occur more frequently in images, this second measure normalizes for the class distribution. BASESYS provides a distribution over all the labels for each pixel in the input image. For BASESCORE, we compute the entropy of this distribution, and averaged it across all pixels in an image.

PASCAL: The second scenario is the PASCAL VOC 2012 segmentation challenge where each pixel in a diverse set of indoor and outdoor images is to be assigned to one of 20

semantic categories (person, chair, etc.) or background. As BaseSys we use the state-of-the-art approach of Carreira and Sminchisescu [13, 14]. It involves generating a large number of plausible object segment hypotheses using constrained parametric min-cuts and bottom up cues, followed by re-ranking them using mid-level cues. This re-ranking is done by training class-specific regressors which provide a high score if a segment is a good match for the category. In addition to the two accuracy metrics described above we also experiment with the standard PASCAL $\frac{\text{intersection}}{\text{union}}$ metric. As BaseScore, we average the score assigned to each segment in an image by the class-specific regressor of BaseSys corresponding to the label that the segment took in the segmentation. Figure 2.1 shows qualitative results.

2.4.2 Vanishing Point Estimation

We explored vanishing point estimation (VP) on 301 indoor scene images in Hedau *et al.* [61]'s dataset, and camera parameter (focal length and rotation) estimation on Satkin *et al.* [121]'s dataset containing 353 bedroom and 166 living room images from the SUN [146] dataset. BaseSys involves detecting lines in image $\boldsymbol{x}_i$ and clustering them in 3 directions (and outliers) [82] to estimate the VPs. Under the Manhattan world assumption, the VP estimates and the orthogonality constraint are used to compute the intrinsic camera parameters and the camera rotation matrix. The error e in each VP estimated in image $\boldsymbol{x}_i$ is measured as its distance from the lines associated with the corresponding ground truth VP. We used several measures for y_i: the minimum of the three e's, their mean, the sum of the two smallest errors, the difference between the estimated and ground truth focal length relative to the true focal length, and the error between the ground truth camera rotation matrix $R(\boldsymbol{x}_i)$ and that estimated by BaseSys is $\hat{R}(\boldsymbol{x}_i)$ measured as the geodesic distance on the 3D manifold of rotation matrices:

$$y_i = \frac{1}{\sqrt{(2)}} || \log(R(\boldsymbol{x}_i)^T \hat{R}(\boldsymbol{x}_i)) ||_F. \tag{2.1}$$

Figure 2.2: Qualitative examples for the camera rotation matrix estimation task. Left: images predicted as poor performing images by ALERT. We see that the three dominant directions are often not clearly visible (e.g. ceiling cropped out of the picture). Right: images predicted as best performing images by ALERT. The 3D orientation of the room is clearly evident.

As BASESCORE for VP estimation, we use the proportion of lines detected in the image that were assigned to the "outliers" cluster when estimating VPs. There is no natural BASESCORE for camera rotation and focal length estimation. See Figure 2.2 for qualitative results.

2.4.3 Image Memorability Prediction

We explored image memorability prediction, where the task is to predict how memorable an input image is [67] in the $[0, 1]$ range. We use the dataset of Isola *et al.* [67] containing 2222 images. As BASESYS we use their approach that trains a Support Vector Regressor using state-of-the-art image features such as HOG, gist, SIFT, self similarity and pixel histograms. We computed the error y_i in the prediction as the relative difference between the score predicted by BASESYS and the ground truth memorability score.[3] Since BASESYS is a regressor, there is no natural definition of BASESCORE for this application.

[3] ALERT regresses to this error. This can be thought of as analogous to one round of L2 boosting, which involves regressing to the error of a regressor. Also relevant is the notion of twicing in statistics [138].

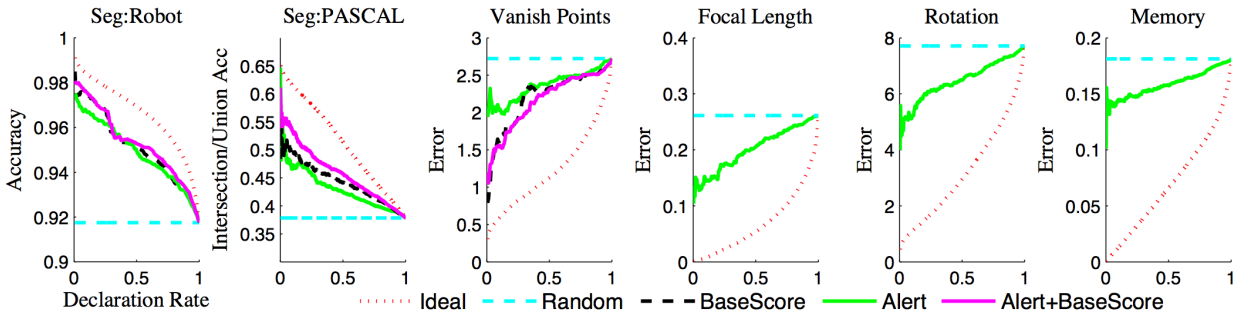


Figure 2.3: Accuracy or Error vs. Declaration Rate Curves. We used accuracy for robot segmentation, intersection/union for PASCAL, and the mean of three errors for vanishing point estimation. Results for other choices of y_i described in Section 2.3 are in supp. with generally similar trends. In fact, ALERT sometimes outperforms BASESCORE. Results are averaged over 10 random train/test splits. BASESCORE is not shown for focal length, rotation and memorability prediction because the first two are direct products of vanishing point estimates, and the inner workings of BASESYS for the latter do not lead to an obvious definition for BASESCORE. Best viewed in color.

2.4.4 Evaluation

We evaluate ALERT using two different metrics.

ADR: The first is an Accuracy vs. Declaration Rate (ADR) curve which is computed by sorting the test images in descending order of their reliability as estimated by ALERT. We then retain only a DR proportion of the test images (DR $\in [0, 1]$), and discard the rest. We compute the accuracy of BASESYS on these retained images and plot accuracy vs. DR. For some applications like vanishing point estimation, it is more natural to use error instead of accuracy. If ALERT were perfect, it would discard the worst performing images. Accuracy would be very high at low DR and then fall gracefully as DR approached 1. If ALERT performed at chance level, on average, the accuracy would remain constant with varying DR. We compare the performance of ALERT to these upper and lower bounds (Figure 2.3). We see that an approach as straightforward as ALERT can perform significantly better than

chance. As expected, BaseSys-specific BaseScore often performs better (see supp. for examples where Alert outperforms BaseScore). A simple summation of Alert and BaseScore improves performance of both. Clearly, Alert captures information orthogonal to BaseSys's beliefs. Some BaseSys may have more systematic failure modes than others, hence Alert helps in some cases more than others. BaseSys for image memorability prediction included many of the same features that Alert was trained on. Alert still predicts its failures reliably, but not as dramatically as in other applications.

RAM: Our second metric evaluates Alert's ability to trade-off the risk of making an incorrect decision with not making a decision at all. This Risk-Averse Metric (RAM) gives BaseSys +1.0 points for a correct answer, -0.5 points for an incorrect answer and 0 points if it makes no decision.[4]

We believe such a metric is crucial when dealing with real applications. By refusing to make a decision when Alert raises a warning, we expect BaseSys to gain more points by trading off incorrect decisions for no decisions. In applications where y_i is a continuous scalar accuracy measure, how high does y_i have to be for a decision to be considered to be "correct"? Different applications may place different thresholds T on y_i to define "good enough" (correct) vs. incorrect answers. If T is high (i.e. most images are incorrect), we would expect Alert to help a lot. If T is low (i.e. most images are correct), Alert will not be beneficial and BaseSys may as well make decisions on all images because it is likely to get them right anyway. Across the different applications, we pick a value of T such that 65% of the images are considered to be correctly classified. This is an (perhaps optimistic) assessment of how well current vision systems work. In Figure 2.4, we plot the average points (per test image) gained by BaseSys when equipped with Alert. We see in all cases, Alert can help BaseSys gain more points than it would if it did not use Alert (i.e. DR = 1.0). Operating at the optimum DR (can be determined via cross validation) can allow

[4]In fact, the ImageClef Robot Vision Challenge `http://www.imageclef.org/2013/robot` uses such a Risk-Averse Metric for recognition. This is similar to, but distinct from applications where false positives are less expensive than false negatives e.g. pedestrian detection.

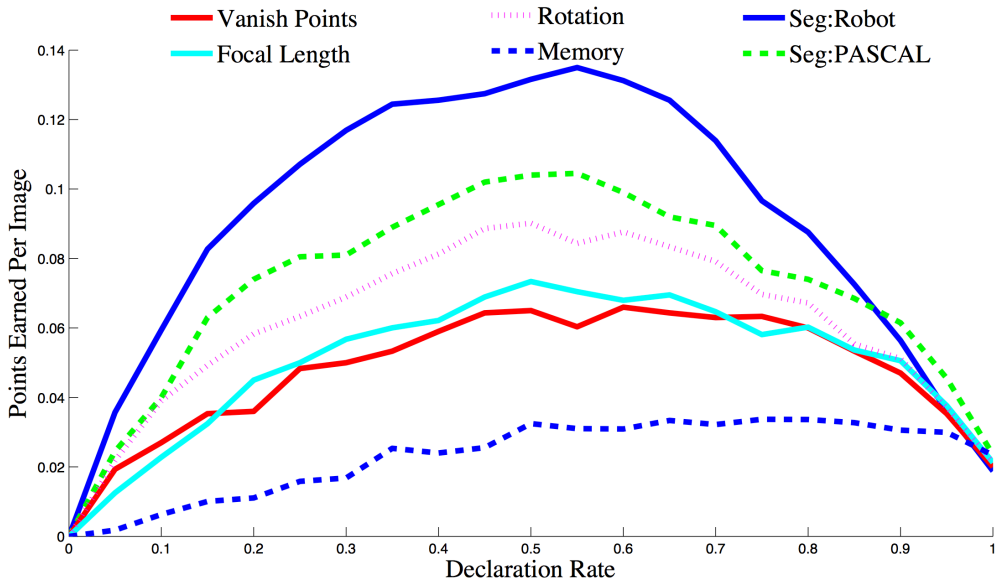


Figure 2.4: Risk-averse Metric: Points achieved by BaseSys when equipped with Alert at varying declaration rates (DR). Alert can help earn more points than the default strategy of always making a decision (DR = 1) or the other extreme of always refusing to make decisions (DR = 0).

BaseSys to gain as many as 7× the points it would without Alert (Figure 2.5).

To understand the intervals in which Alert is more helpful, let us consider the cost of an incorrect decision ($-C$) relative to that of a correct decision (1.0). In our experiments so far, we used $C = 0.5$. If C is very low or very high, Alert would have to be very accurate in predicting failures for BaseSys to benefit from it. Otherwise, DR = 1 and DR = 0 would be optimum for low C and high C respectively. For instance, if C is very high, even one mistake undetected by Alert can reduce the total points earned by BaseSys to a large negative value, and BaseSys would be better off not making any decisions at all (DR = 0). The quality of Alert can be assessed by the range in values of C for which it is still beneficial. In our experiments, we find that Alert can help BaseSys for C in $[0.14, 9]$, $[0.14, 2.34]$, $[0.41, 0.67]$, $[0.12, 0.82]$, $[0.16, 1.86]$ and $[0.16, 1.5]$ for the 6 applications in Figure 2.5. Since the optimum DR is to be determined via cross validation, while unable to boost performance

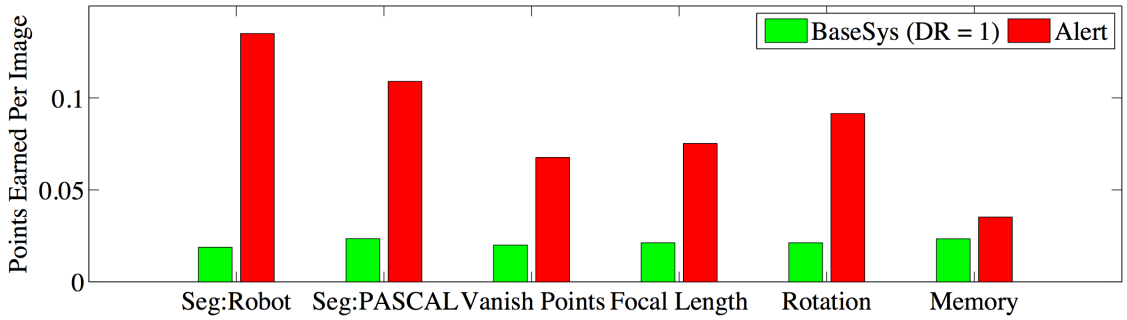


Figure 2.5: Risk-averse Metric: Summary of Figure 2.4. Points achieved by ALERT by picking the appropriate DR as compared to BASESYS.

outside these ranges, ALERT would default to DR = 0 or 1 and not hurt performance.

2.5 Benefiting A Downstream Application

Visual attributes are mid-level semantic image properties that are considered to be shareable across categories. Hence, attribute predictors are often used to transfer information from one set of categories to a previously unseen set. This involves transfer of knowledge. ALERT can be used to predict the potential failure of BASESYS in transferring knowledge from one domain to the other. This can in turn be used to benefit downstream applications, such as zero-shot learning [80] in the case of attribute predictors.

We experimented with attribute predictors in four domains: animals, faces, scenes and objects. We use the Animals With Attributes (AWA) dataset of Lampert *et al*. [80] containing 8609 images, the Public Figures Face Database (PubFig) of Kumar *et al*. [79] containing 42461 images, 7160 images from the SUN Attribute Database (SUN) of Patterson and Hays [109] and the aPascal+aYahoo (UIUC) dataset of Farhadi *et al*. [40] containing 8999 images. They contain 85, 73, 102 and 64 attributes respectively. We collected ground truth attribute annotations for PubFig (not available with dataset) using Amazon Mechanical Turk, and will make them publicly available.

Figure 2.6: Qualitative results for the "wearing sunglasses" attribute on PubFig. Left: Images predicted by ALERT as poor performing. Many of these have poor lighting and shadows especially in the eye region. Right: images predicted by ALERT as high performing. These are all taken under good lighting conditions. The images for which the BASESYS attribute predictor generated the incorrect output are marked by a red border.

As BASESYS we use the attribute predictors provided by the authors for AWA, PubFig and UIUC. For SUN, we use the authors' code to train the predictors. Each predictor is a binary classifier converted to a probabilistic estimate $\pi(\boldsymbol{x}_i)$. We use the 0-1 loss as y_i. If the attribute predictor is sure of the attribute presence/absence, but is wrong, its decision is considered to be a mistake. If it is sure and right, or unsure of its decision, its decision is not a mistake:

$$y_i = \begin{cases} 0, & \text{if } \hat{l}\left(\boldsymbol{x}_i\right) = l\left(\boldsymbol{x}_i\right) \\ 0, & \text{if } \pi(\boldsymbol{x}_i) \in [\eta_1, \eta_2] \\ 1, & \text{otherwise} \end{cases} \tag{2.2}$$

where $l\left(\boldsymbol{x}_i\right)$ is the true binary label of $\boldsymbol{x}_i$ and $\hat{l}\left(\boldsymbol{x}_i\right)$ is the label predicted by BASESYS. We set η_1 and η_2 to 0.25 and 0.75 respectively in all our experiments. We train one ALERT for every attribute. The threshold on the score beyond which ALERT raises a warning was chosen via cross validation. Since we already have trained attribute predictors, we use their outputs to represent each image to train ALERT. $\boldsymbol{x}_i$ is thus an attributes-based M-dimensional descriptor, where M is the number of attributes. We train ALERT on a held

out set of unseen categories, and test it on a disjoint set of categories.[5] Qualitative results are in Figure 2.6. For space considerations, ADR curves similar to Figure 2.3 are shown in supplementary material. ALERT significantly outperforms BASESCORE in most datasets, because ALERT is explicitly trained to detect attribute classifiers' failures to generalize to unseen categories – which BASESCORE is typically not aware of. We now demonstrate how a downstream application that uses outputs of attribute predictors as input can benefit from such an ALERT.

Zero-shot Learning: We leveraged ALERT to improve zero-shot learning performance [80]. Zero-shot learning involves learning novel categories simply from their binary attributes-based descriptions (e.g. zebras are stripped, black and white, have four legs, etc.). Each category c is described with a list of M attributes $[a_1, \ldots, a_M]$, $a_m \in \{0, 1\}$. The probability of an instance $\boldsymbol{x}$ belonging to class c is

$$p(c|\boldsymbol{x}) \propto \prod_{m=1}^{M} p(a_m|\boldsymbol{x}), \tag{2.3}$$

where $p(a_m = 1|\boldsymbol{x}) = \pi_m(\boldsymbol{x})$. These attribute predictors are trained on a set of previously seen categories, which are disjoint form the set of unseen test categories. The attribute predictors are expected to generalize across this domain transfer. The test instance $\boldsymbol{x}$ is assigned the unseen class with the highest probability. The standard approach of Lampert *et al.* [80] uses all M attributes in Equation 2.3 ignoring the variability in the reliability of each attribute prediction in an input image. Instead, when using ALERT, for each test image we ignore the attributes for which ALERT raises a warning.[6] The same set of attributes are ignored for all classes, so the probabilities $p(c|\boldsymbol{x})$ are comparable. Note that different attributes are ignored for different test images.

[5] See supplementary material for details on train/val/test splits.

[6] Notice that in this zero-shot learning application, ALERT is used on images from categories not seen by BASESYS during its training, making the "if you can train ALERT, you could have trained a better BASESYS in the first place" reasoning discussed in the introduction further inapplicable.

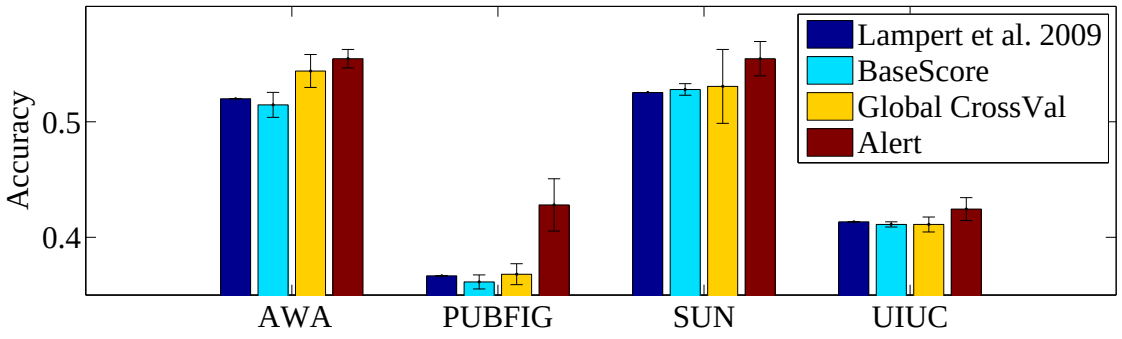


Figure 2.7: Zero-shot learning results.

We report our zero-shot learning results in Figure 2.7. In addition to comparing to the standard approach of using all attributes [80], we compare to two other baselines. They both ignore attributes as needed, but by using alternative strategies than ALERT. We set the parameters of both baselines so that they ignore on average the same number of attributes per image as ALERT[7] making comparisons fair. The first baseline (BASESCORE) ignores attributes that are not confident (i.e. $\pi_m(\boldsymbol{x}_i)$ is close to 0.5). The second baseline computes the accuracies of all attributes on a validation set, and ignores the least accurate ones (similar in spirit to how poselets are selected in [12]). Note that this baseline ignores the same attributes for all images, while ALERT adapts its decisions to individual input instances.

We see that ALERT outperforms all three baselines. Since the zero-shot model is probabilistic, its decision is not significantly influenced by under-confident attribute predictions anyway, making BASESCORE less effective. The cross-validation baseline also fails to improve performance consistently (except for AWA) due to its global nature. However, ALERT makes image-specific decisions regarding which attributes to ignore and consistently improves performance across all four datasets. PubFig has the largest number of unseen test categories making knowledge transfer and ZSL harder. Here ALERT shows the most improvement. Note that ALERT here uses attribute predictions themselves as features. So the improve-

[7] ALERT ignores 12 out of 85 attributes per test image for AWA, 12 out of 73 for PubFig, 32 out of 102 for SUN and 7 out of 64 for UIUC.

ment in performance is not because of access to additional visual information – it is because of explicitly reasoning about failures.

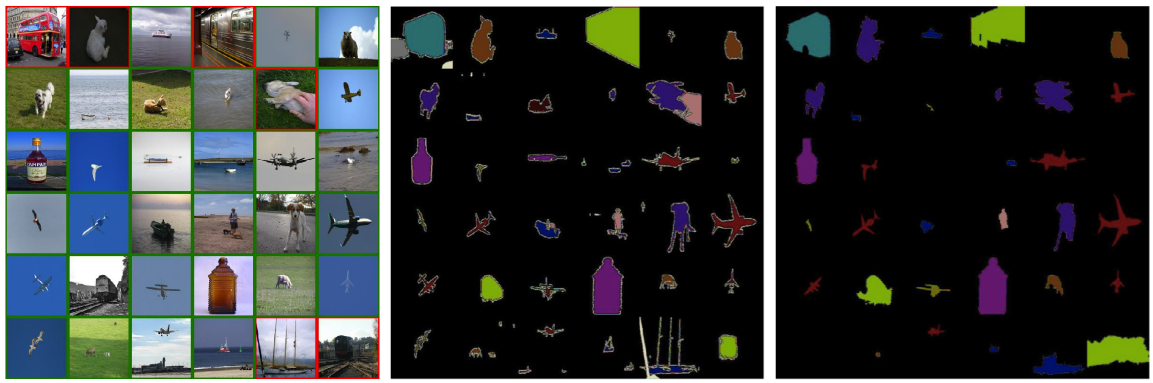
2.6 Conclusion

We take a step in the direction of building self-evaluating vision systems that fail gracefully, making them more usable in real world applications even with their existing imperfections. We introduce ALERT, a warning system that analyzes the input instance and predicts if a vision system is likely to produce an unreliable response. We demonstrate its generality by applying it to four diverse applications ranging from segmentation and 3D layout estimation to image memorability prediction and attributes-based scene and object recognition. We show that ALERT can predict failure reliably, and can help improve the performance of a downstream application. We advocate the use of two metrics (accuracy vs. declaration rate curves and risk-averse metrics) to facilitate further progress in failure predicting systems.

2.7 Appendix

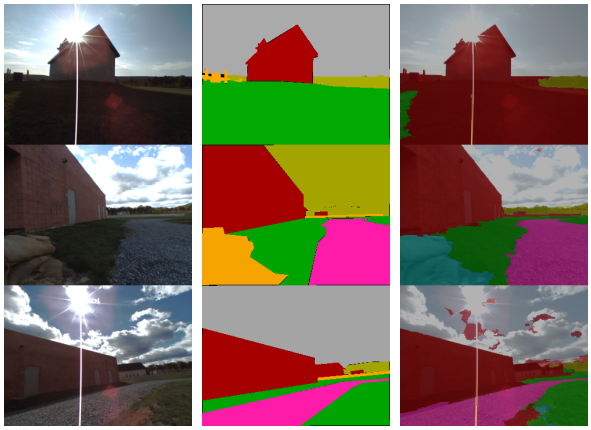
Details of train/val/test splits for attribute prediction experiments (Section 2.5) For each of the datasets, we select three sets of categories: a training set for ALERT, a validation set for choosing its parameters, and a test set. We use 50 (PubFig), 20 (AWA), 50 (SUN) and 20 (UIUC) images per category for the training set, and 15 and 15 images per category for the validation and test sets. The number of categories in the 3 sets are as follows: 40, 5, 5 out of 50 for AWA; 40, 10, 10 out of 140 in evaluation set of PubFig; 40, 5, 5 out of 358 for SUN; and 20, 6, 6 out of 20+12 (aPascal+aYahoo) for UIUC. We report the average accuracy over 5 validation+test splits.

(b) Ground truth (c) BaseSys output

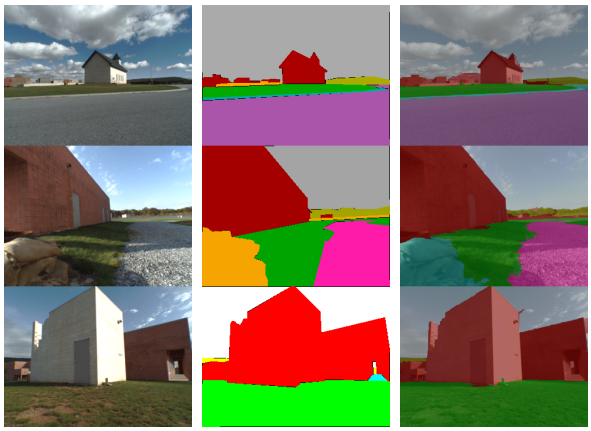


(d) Image (e) Ground truth (f) BaseSys output

Figure 2.8: Qualitative results of our proposed approach Alert on the PASCAL segmentation task. (a) shows images predicted by Alert as unreliable poor performing images; (d) shows images predicted to be the best performing. Alert clearly favored images with easy to segment isolated objects over complex images containing a large number of small regions. Segmentation accuracy is measured as the per-class $\frac{\text{intersection}}{\text{union}}$ metric averaged across all classes present in the ground truth (+ background). Images with accuracies higher than the median accuracy across all images are outline in green. Those below the median accuracy are outlined in red. There are more red outlines in (a) than in (d). Clearly, Alert is quite effective at identifying images where BaseSys will have poor accuracies, simply by processing the input image even before BaseSys is applied to the image. This Figure is an expanded version of Figure 2.1. Best viewed in color.



(a) Image (b) Ground truth (c) BaseSys output



(d) Image (e) Ground truth (f) BaseSys output

Figure 2.9: Qualitative results of our proposed approach Alert on a ground robot segmentation task. (a) shows images predicted by Alert as unreliable poor performing images; (d) shows images predicted to be the best performing. Notice that Alert flagged the images that are corrupted with strong shadows and over/under-exposure while retaining the images with good contrast. Indeed, the output segmentations (c) of images in (a) typically contain large errors when compared to the ground truth segmentations (b). The segmentations (f) of images in (d) are typically accurate when compared to ground truth segmentations (e). Of course, Alert's predictions of unreliability are not always accurate. Although the the bottom image in (a) is flagged by Alert as being unreliable, its segmentation was in fact quite accurate. Recall that Alert only examines the input image *before* running the segmentation engine BaseSys to assess if BaseSys is likely to produce an accurate response for the input instance or not. This Figure is an expanded version of Figure 2.1 .

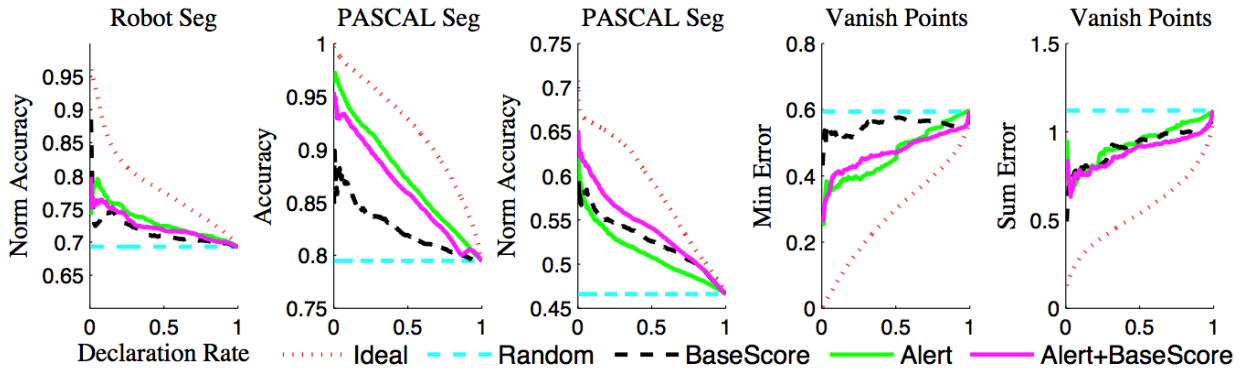


Figure 2.10: Accuracy or Error vs. Declaration Rate Curves. We used accuracy for robot segmentation, intersection/union for PASCAL, and the mean of three errors for vanishing point estimation for Figure 2.3. Other choices of y_i described in Section 2.4 are shown above. Results are averaged over 10 random train/test splits. ALERT performs much better than chance, and is often better than BASESCORE, which unlike ALERT is BASESYS-specific and utilizes the output of BASESYS on the input image to estimate its confidence. Best viewed in color.

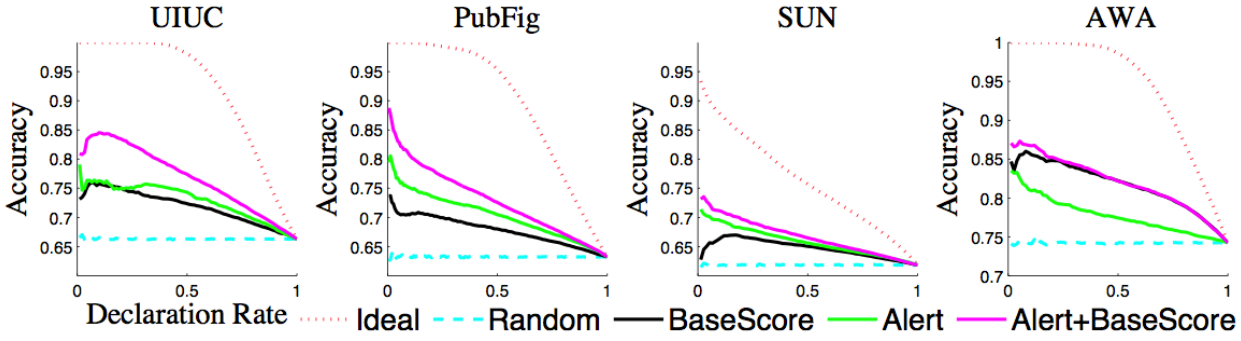


Figure 2.11: Accuracy vs. Declaration Rate Curves for attribute prediction as described in Section 2.5. ALERT generally outperforms BASESCORE. Attributes on the AWA dataset are category-specific attributes as opposed to image-specific attributes as in the remaining three datasets. A combination of ALERT and BASESCORE performs better than both in all cases. Best viewed in color.

Chapter 3

Portfolio Vision System

3.1 Introduction

Computer vision techniques today are booming. For a large variety of different tasks, a variety of approaches are competing to obtain better performance. Improvements have be witnessed on benchmark datasets such as Caltech101 [86], PASCAL [37], ImagNet [27], SUN [146] and Microsoft COCO [90]. Accuracies on such datasets are typically computed as the average performance a system had across all test instances. Note that a system that obtains the highest average performance need not be the most accurate on *each* test instance. Different systems are complementary, and may be better suited for different input instances. Clearly, given an input instance, if the best performing system can be automatically identified, the overall performance obtained will be higher than any individual system.

We propose such a system: PORTFOLIO. PORTFOLIO is able to automatically identify the (likely) best perception system for a given input from a candidate pool of perception systems. See Figure 3.1. PORTFOLIO is general and not catered to specific perception tasks. Notably, PORTFOLIO is designed to predict the likely performance for each candidate perception system by only looking at the input instances themselves, without even running

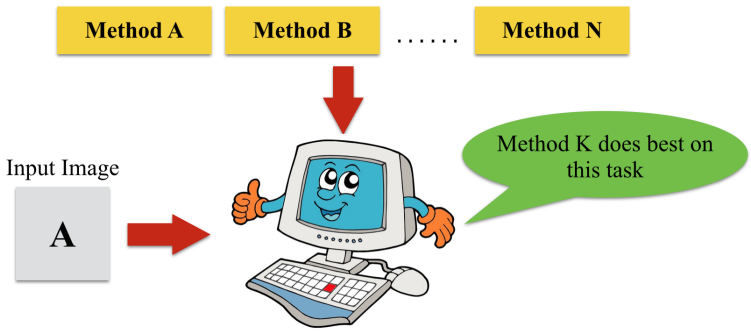


Figure 3.1: Our proposed PORTFOLIO system

the perception systems, which might be very time-consuming in some circumstances.

Zhang *et al*. proposed a failure prediction system [154] which given an input instance, was able to reliably predict whether a given perception system is likely to fail on that input instance or not. Our approach builds on that work to identify the perception system that is least likely to fail on an image (i.e., is most likely to succeed).

3.2 Related work

Robot Portfolio is an idea that comes from economy and finance. A robot has been designed and trained, and then it is used to concentrate on one or two industries in the stock market. By doing this, the robot will identify a number of promising stocks in those industries with some existing stock screen software. From this pool, the robot is able to pick the top 10 stocks with lowest PE ratios, that is, stocks that are more likely to make money in the near future. In this work, we have a similar scenario, in that we have a pool of approaches and our system is trained to predict the which one is most likely to provide the highest performance on a particular test instance. As a result, we call our system PORTFOLIO: a portfolio approach to machine perception.

Meta-recognition was proposed by Scheirer in 2011 [122]. It involves predicting the performance of a system by analyzing the scores of the system after it has been run on a test

instance. Our approach on the other hand involves predicting the performance of a system even *before* it has been run on an input instance. That is, our approach analyzes the input instance alone when predicting the likely performance of a perception system on that instance.

AdaBoost [47] learns a combined classifier from several weak classifiers. It assigns a set of weights to each classifier, which are set during training. The same set of weights are then use for each test instance. Our approach involves *instance-specific* selection of perception systems.

Identifying the best model is not a new topic in computer vision. The Deformable Part-based Model [42] identifies the model component with the highest score on (i.e., that best explains) an input instance; [23] tries to pick the best texture classification method from several candidate methods. They designed features specific to the task at hand to automatically identify the best-performing method. Given input images and saliency detection outputs from several methods, [96] aims to find the best saliency map. In addition, Aodha et al. proposed a decision-tree-based algorithm to pick method from candidates, demonstrating their idea in optical flow and other applications [6]. These works all either: (1) are specifically designed for the task at hand or (2) make predictions about the quality of the output *after* the perception system has been run (our approach makes these predictions *before* running the perception system). Our approach is generic and can be applied to any perception task without modification. It is an end-to-end system which only takes images as inputs, and does not require application-specific features.

3.3 Approach

Given a vision task and several approaches to solve it, we would like to learn PORTFOLIO: a system that can predict the best-performing method by analyzing the input instance.

We are given a set of training instances x and corresponding measures of accuracies from M

methods on each of training instances $\mathbf{y}$. The accuracy measure of a perception system can be a vector or simply a scalar. In this work, we only consider the scenario where the accuracy measure is a scalar, resulting in $\mathbf{y}$ being an M-dimensional vector. For each training sample, we sort the candidate approaches in descending order of their accuracies and use the best-performing approach index as the label for each instance. The accuracy can be set as any metric that is able to characterize the performance of candidate methods on the underlying vision task. For example, in semantic segmentation (assigning every pixel in an image to a category e.g., car, grass), the accuracy can be intersection over union, which is the standard evaluation metric.

We first start by evaluating the best performance our approach can achieve. That is, if an oracle told us which perception system is indeed the best system for each test instance, what overall performance would we get? We describe this Oracle approach first. We then describe PORTFOLIO, which *automatically* identifies the perception system that is likely to be the most accuracy for a test instance.

3.3.1 Oracle

We view this method as the best accuracy that we can achieve. For each test instance, we obtain its outputs from all the candidate methods. We compute the accuracy for each of them by comparing the outputs to ground truth annotations. We then pick the method with the highest accuracy for each input instance. After doing this for each instance in the dataset, we can compute the overall performance using a standard metric. Clearly, this approach utilizes ground truth annotations on test data, and is thus an upper bound on the performance PORTFOLIO can obtain.

3.3.2 Portfolio

We now describe PORTFOLIO, our system that automatically predict the best-performing method on an input instance. PORTFOLIO is an end-to-end classifier trained using a Convolution Neural Network (CNN). The inputs during training are images and the labels are the indices of the best-performing method on each input image. At test time, for each novel instance, our system outputs the method index, which is the predicted as the best-performing approach. That approach can then be run on the test instance to obtain the desired perception output.

Our CNN is composed of 6 layers, which has 2 convolution layers, 2 max pooling layers and 2 fully connected layers. For each convolution layer, we set the kernel size as 3 by 3. As output, there are 6 output channels and 12 output channels for each of them. For the fully connected layer, we set it as 200 nodes, and output 4 predictions, corresponding to the number of candidate methods. As inputs to the neural network, we resize the each image to 32 by 32. Notice that in most perception tasks, large datasets are not available. To avoid overfitting but have good results at the same time, we set the maximum training iteration as 200, while the learning rate was 1e-4. We implemented our model in Torch [134].

3.4 Experiments and results

To demonstrate the reliability of our methods, we apply them on two different applications, semantic segmentation and saliency detection on standard benchmark datasets, which are PASCAL 2012 dataset and MSRA10k dataset, respectively.

3.4.1 Semantic segmentation

Semantic segmentation involves classifying each pixel in the image as belonging to one of several categories e.g., plane, sofa, car. Pixels that belong to the same category are plotted

with the same color. Therefore, segments with the same color represent one object class across all the images. For the pixels that do not belong to any class, we leave them black and classify them as background.

Multiple methods have been proposed to tackle with this problem. In this work, we take 4 methods as candidate approaches, which are fisher codemap [88], SDS [60], re-ranking [91] and SSDL [22].

We use the PASCAL 2012 segmentation validation set as our testbed. It contains 1449 images. We randomly split the dataset into two halves, one for training and one for testing. We evaluate the performance over 5 such splits, and average the performance. To measure the accuracy, we follow the standard segmentation evaluation metric: Intersection Over Union (IOU), which is defined as the overlap between the prediction and the groundtruth divided by the union of the two.

We first run all 4 methods on on the training images. For each segmentation output, we compute its IOU accuracy by comparing it with the ground truth. Therefore, each training image has 4 accuracies. We then pick the method that has the highest accuracies and make its index the training label. We then train our PORTFOLIO system. At test time, PORTFOLIO predicts the approach that is likely to be the most accurate. On every test image, we run the predicted method and obtain its output. Overall performance can then be computed across the entire test set.

We also perform the oracle experiment described earlier on the dataset by selecting the true most accurate approach on each test instance, and computing overall performance.

Table 3.1 shows the comparison of the segmentation accuracy of the 4 candidate methods, the oracle method (upper bound), and our PORTFOLIO approach. Of course, oracle achieves the highest performance, with $\sim$13 % better than the best candidate method. This shows the potential gains in improvement that can be achieved by a good PORTFOLIO system. Note that this gain is possible just by leveraging the complementary nature of *existing* segmentation algorithms, without training more accurate segmentation algorithms. We see

that our PORTFOLIO approach performs better than any of the individual approaches. Some qualitative results are shown in Figure 3.2(a).

[60]	[88]	[22]	[91]	Oracle	Ours
50.09%	46.80%	47.75%	50.85%	64.46%	51.25%

Table 3.1: Comparison of segmentation accuracy for candidate methods and our approaches

3.4.2 Saliency detection

Saliency detection is the task of detect the most distinctive regions in an image. Given an image, the most salient regions are marked in white, while other regions are left as black.

We apply our methods on one of the benchmark datasets: MSRA10k [20], which contains 10000 images and annotated ground truth saliency maps. In this work, we randomly select 5000 images as our testbed. As described earlier, we run cross validation by randomly splitting the data into two halves, one for training and one for testing. We do this for 5 trails and report the average performance. We consider 4 state-of-the-art approaches in saliency detection, which are RC [20], CB [70], SWD [34], SEG [112].

To evaluate the performance, we use one of the most widely used metrics for saliency detection, area under ROC curve (AUC). The higher the AUC score is, the better.

The results for oracle is shown in Table 3.2. We witness the same trend for the oracle as in semantic segmentation. Oracle (upper bound) results in $\sim$4% improvement over the best candidate method. We also show some qualitative results in Figure 3.2(b), which contains the original image, the ground saliency detection results, output of the system selected by oracle

[20]	[70]	[34]	[112]	Oracle	Ours
75.51%	74.09%	70.70%	56.67%	80.26%	75.52%

Table 3.2: Comparison of saliency detection accuracy for candidate methods and our approach

3.5 Discussion

PORTFOLIO involves the idea of picking the most plausible solution from candidates by analyzing the input alone. We evaluated two approaches (oracle and PORTFOLIO) on two machine perception applications: semantic segmentation and saliency detection. Our approaches outperform the candidates on both these tasks. This indicates that the candidate approaches are complementary to each other. Indeed, the approach that has the best performance overall might not be the best approach on *each* test instance. In addition, our results show that our PORTFOLIO system is indeed capable of picking input-specific perception systems, just by analyzing the input instances.

As we discussed before, the oracle method has access to the ground truth knowledge about which perception system is going to be the most accurate for each test instance. It shows the potential improvement that our method can achieve. That is, if our PORTFOLIO system is able to predict the best-performing method correctly for all the test samples, it will have the same performance as the oracle approach.

We checked the prediction accuracy of PORTFOLIO on both the tasks. We found that it was able to identify the most accurate perception system for $\sim$35% of the input instances for semantic segmentation.

3.6 Conclusion

In this work, we proposed PORTFOLIO: an approach that leverages failure prediction to automatically identify the best-performing perception system for each input instance from a candidate of perception systems all solving the same perception task. We evaluated two approaches: an oracle approach that demonstrates the possible gain in performance a portfolio approach can achieve, and the performance of our trained PORTFOLIO system. Results on two perception tasks (semantic segmentation and saliency detection) showed improvements in performance over any of the candidate perception systems individually. PORTFOLIO is generic (can be applied to any perception task), and only involves analyzing the input instance (i.e., can be applied *before* running the different candidate perception systems on the input instance, which is often computationally expensive).

(a) Semantic Segmentation

(b) Saliency Detection

Figure 3.2: Qualitative results on semantic segmentation and saliency detection

Chapter 4

Binary VQA

4.1 Introduction

Problems at the intersection of vision and language are increasingly drawing more attention. We are witnessing a move beyond the classical "bucketed" recognition paradigm (e.g. label every image with categories) to rich compositional tasks involving natural language. Some of these problems concerning vision and language have proven surprisingly easy to take on with relatively simple techniques. Consider image captioning, which involves generating a sentence describing a given image [19,32,39,73,76,99,140]. It is possible to get state of the art results with a relatively coarse understanding of the image by exploiting the statistical biases (inherent in the world and in particular datasets) that are captured in standard language models.

For example, giraffes are usually found in grass next to a tree in the MS COCO dataset images [90]. Because of this, the generic caption "A giraffe is standing in grass next to a tree" is applicable to most images containing a giraffe in the dataset. The machine can confidently generate this caption just by recognizing a "giraffe", without recognizing "grass", or "tree", or "standing", or "next to". In general, captions borrowed from nearest neighbor

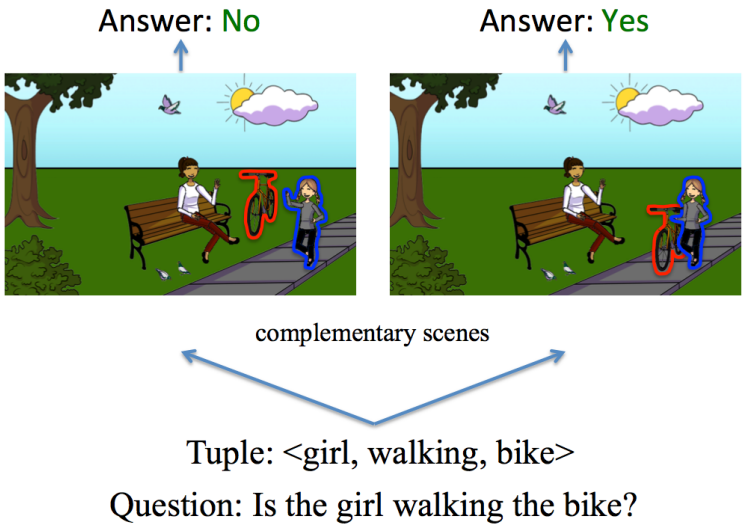


Figure 4.1: We address the problem of answering binary questions about images. To eliminate strong language priors that shadow the role of detailed visual understanding in visual question answering (VQA), we use abstract scenes to collect a *balanced* dataset containing pairs of *complementary* scenes: the two scenes have opposite answers to the same question, while being visually as similar as possible. We view the task of answering binary questions as a visual verification task: we convert the question into a tuple that concisely summarizes the visual concept, which if present, result in the answer of the question being "yes", and otherwise "no". Our approach attends to relevant portions of the image when verifying the presence of the visual concept.

images result in a surprisingly high performance [30].

A more recent task involving vision and language is Visual Question Answering (VQA). A VQA system takes an image and a free-form natural language question about the image as input (e.g. "What is the color of the girl's shoes?", or "Is the boy jumping?"), and produces a natural language answer as its output (e.g. "blue", or "yes"). Unlike image captioning, answering questions requires the ability to identify specific details in the image (e.g. color of an object, or activity of a person). There are several recently proposed VQA datasets on real images e.g. [4, 50, 97, 98, 115], as well as on abstract scenes [4]. The latter allows research

on semantic reasoning without first requiring the development of highly accurate detectors.

Even in this task, however, a simple prior can give the right answer a surprisingly high percentage of the time. For example, in the VQA dataset (with images from MS COCO) [4], the most common sport answer "tennis" is the correct answer for 41% of the questions starting with "What sport is". Similarly, "white" alone is the correct answer for 23% of the questions starting with "What color are the". Almost half of all questions in the VQA datatset [4] can be answered correctly by a neural network that ignores the image completely and uses the question alone, relying on systematic regularities in the kinds of questions that are asked and what answers they tend to have.

This is true even for binary questions, where the answer is either "yes" or "no", such as "Is the man asleep?" or "Is there a cat in the room?". One would think that without considering the image evidence, both answers would be equally plausible. Turns out, one can answer 68% of binary questions correctly by simply answering "yes" to all binary questions. Moreover, a language-only neural network can correctly answer more than 78% of the binary questions, without even looking at the image.

As also discussed in [135], such dataset bias effects can give a false impression that a system is making progress towards the goal of understanding images correctly. Ideally, we want language to pose challenges involving the visual understanding of rich semantics while not allowing the systems to get away with ignoring the visual information. Similar to the ideas in [52], we propose to unbias the dataset, which would force machine learning algorithms to exploit *image* information in order to improve their scores instead of simply learning to game the test. This involves not only having an equal number of "yes" and "no" answers on the test as a whole, but also ensuring that *each particular question is unbiased*, so that the system has no reason to believe, without bringing in visual information, that a question should be answered with "yes" or "no."

In this work, we focus on binary (yes/no) questions for two reasons. First, unlike open-ended questions (Q: "what is the man playing?" A: "tennis"), in binary questions (Q: "is

the man playing tennis?") all relevant semantic information (including "tennis") is available in the question alone. Thus, answering binary questions can be naturally viewed as *visual verification* of concepts inquired in the question ("man playing tennis"). Second, binary questions are easier to evaluate than open-ended questions.

Although our approach of visual verification is applicable to real images (more discussion in Section 4.6), we choose to use abstract images [4, 5, 157–159] as a test bed because abstract scene images allow us to focus on high-level semantic reasoning. They also allow us to balance the dataset by making changes to the images, something that would be difficult or impossible with real images.

Our main contributions are as follows: (1) We balance the existing abstract binary VQA dataset [4] by creating complementary scenes so that all questions[1] have an answer of "yes" for one scene *and* an answer of "no" for another closely related scene. We show that a language-only approach performs significantly worse on this balanced dataset. (2) We propose an approach that summarizes the content of the question in a tuple form which concisely describes the visual concept whose existence is to be verified in the scene. We answer the question by verifying if the tuple is depicted in the scene or not (See Figure 4.1). We present results when training and testing on the balanced and unbalanced datasets.

4.2 Related work

Visual question answering. Recent work has proposed several datasets and methods to promote research on the task of visual question answering [4, 8, 50, 52, 97, 98, 115, 137], ranging from constrained settings [52, 97, 115] to free-form natural language questions and answers [4, 8, 50, 98, 137]. For example, [52] proposes a system to generate binary questions from templates using a fixed vocabulary of objects, attributes, and relationships between

[1] nearly all. About 6% of test questions do not lend themselves to this modification. See Section 4.3 for details.

objects. [137] has studied joint parsing of videos and corresponding text to answer queries about videos. [97] studied VQA with synthetic (templated) and human-generated questions, both with the restriction of answers being limited to 16 colors and 894 object categories or sets of categories. A number of recent papers [4, 50, 98, 115] proposed neural network models for VQA composing LSTMs (for questions) and CNNs (for images). [4] introduced a large-scale dataset for free-form and open-ended VQA, along with several natural VQA models. [8] uses crowdsourced workers to answer questions about visual content asked by visually-impaired users.

Data augmentation. Classical data augmentation techniques (such as mirroring, cropping) have been widely used in past few years [65, 129] to provide high capacity models additional data to learn from. These transformations are designed to not change the label distribution in the training data. In this work, we "augment" our dataset to explicitly change the label distribution. We use human subjects to collect additional scenes such that *every question* in our dataset has equal number of 'yes' and 'no' answers (to the extent possible). In that sense, our approach can be viewed as semantic data augmentation. Several classification datasets, such as ImageNet [27] try to be balanced. But this is infeasible for the VQA task on real images because of the heavy-tail of concepts captured by language. This motivates our use of abstract scenes.

Visual abstraction + language. A number of works have used abstract scenes to focus on high-level semantics and study its connection with other modalities such as language [5,45,53, 92,139,157–159], including automatically describing abstract scenes [53], generating abstract scenes that depict a description [158], capturing common sense [45,92,139], learning models of fine-grained interactions between people [5], and learning the semantic importance of visual features [157, 159]. Some of these works have also taken advantage of visual abstraction to "control" the distribution of data, for example, [5] collects equal number of examples for each verb/preposition combinations, and [157] have multiple scenes that depict the exact same sentence/caption. Similarly, we balance the dataset by making sure that each question in the dataset has a scene for "yes" and another scene for "no" to the extent possible.

Visual verification. [119, 139] reason about the plausibility of commonsense assertions (men, ride, elephants) by gathering visual evidence for them in real images [119] and abstract scenes [139]. In contrast, we focus on visually-grounded image-specific questions like "Is the man in the picture riding an elephant?". [158] also reasons about relations between two objects, and maps these relations to visual features. They take as input a description and automatically generate a scene that is compatible with all tuples in the description and is a plausible scene. In our case, we have a single tuple (summary of the question) and we want to verify if it exists in a given image or not, for the goal of answering a free form "yes/no" question about the image.

Visual attention involves searching and attending to relevant image regions. [72, 148] uses alignment/attention for image caption generation. Input is just an image, and they try to describe the entire image and local regions with phrases and sentences. We address a different problem: visual question answering. We are given an image *and text* (a question) as input. We want to align parts of the question to regions in the image so as to extract detailed visual features of the regions of the image being referred to in the text.

4.3 Datasets

We first describe the VQA dataset for abstract scenes collected by [4]. We then describe how we balance this dataset by collecting more scenes.

4.3.1 VQA dataset on abstract scenes

Abstract library. The clipart library contains 20 "paperdoll" human models [5] spanning genders, races, and ages with 8 different expressions. The limbs are adjustable to allow for continuous pose variations. In addition to humans, the library contains 99 objects and 31 animals in various poses. The library contains two different scene types – "indoor" scenes,

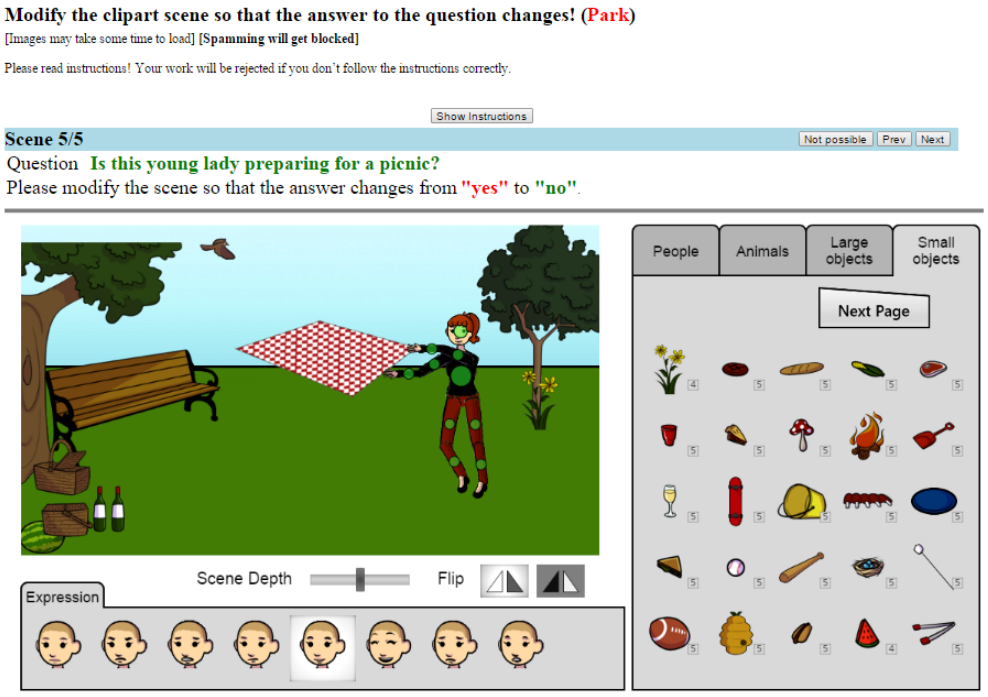


Figure 4.2: A snapshot of our Amazon Mechanical Turk (AMT) interface to collect complementary scenes.

containing only indoor objects, e.g. desk, table, etc., and "outdoor" scenes, which contain outdoor objects, e.g. pond, tree, etc. The two different scene types are indicated by different background in the scenes.

VQA abstract dataset consists of 50K abstract scenes, with 3 questions for each scene, with train/val/test splits of 20K/10K/20K scenes respectively. This results in total 60K train, 30K validation and 60K test questions. Each question has 10 human-provided ground-truth answers. Questions are categorized into 3 types – 'yes/no', 'number', and 'other'. In this work, we focus on 'yes/no' questions, which gives us a dataset of 36,717 questions- 24,396 train and 12,321 val questions. Since test annotations are not publicly available, it is not possible to find the number of 'yes/no' type questions in test set. We use the binary val questions as our unbalanced test set, a random subset of 2,439 training questions as our unbalanced validation set, and rest of the training questions as our unbalanced train set.

4.3.2 Balancing abstract binary VQA dataset

We balance the abstract VQA dataset by posing a counterfactual task – given an abstract scene and a binary question, what would the scene have looked like *if the answer to the binary question was different*? While posing such counterfactual questions and obtaining corresponding scenes is nearly impossible in real images, abstract scenes allow us to perform such reasoning.

We conducted the following Mechanical Turk study – given an abstract scene, and an associated question from the VQA dataset, we ask subjects to *modify the clipart scene such that the answer changes* from 'yes' to 'no' (or 'no' to 'yes'). For example, for the question "Is a cloud covering the sun?", a worker can move the 'sun' into open space in the scene to change the answer from 'yes' to 'no'. A snapshot of the interface is shown in Figure 4.2.

We ask the workers to modify the scene as little as possible. We encourage minimal changes because these complementary scenes can be thought of as hard-negatives/positives to learn subtle differences in the visual signal that are relevant to answering questions. This signal can be used as additional supervision for training models such as [10, 31, 108, 151] that can leverage explanations provided by the annotator in addition to labels. Our complementary scenes can also be thought of as analogous to good pedagogical techniques where a learner is taught concepts by changing one thing at a time via contrasting (e.g., one fish vs. two fish, red ball vs. blue ball, etc.).

Note that there are some (scene, question) pairs that do not lend themselves to easy creation of complementary scenes with the existing clipart library. For instance, if the question is "Is it raining?", and the answer needs to be changed from 'no' to 'yes', it is not possible to create 'rain' in the current clipart library. Fortunately, these scenes make up a small minority of the dataset (e.g., 6% of the test set).

To keep the balanced train and test set comparable to unbalanced ones in terms of size, we collect complementary scenes for ∼half of the respective splits – 11,760 from train and 6,000

from test set. Since Turkers indicated that 2,137 scenes could not be modified to change the answer because of limited clipart library, we do not have complementary scenes for them. In total, we have 10,295 complementary scenes for the train set and 5,328 complementary scenes for test, resulting in balanced train set containing 22,055 samples and balanced test set containing 11,328 samples. We further split a balanced set of 2,202 samples from balanced train set for validation purposes. Examples from our balanced dataset are shown in Figure 4.1 and Figure 4.4.

We use the publicly released VQA evaluation script in our experiments. The evaluation metric uses 10 ground-truth answers for each question to compute performance. To be consistent with the VQA dataset, we collected 10 answers from human subjects using AMT for all complementary scenes in the balanced test set.

We compare the degree of balance in our unbalanced and balanced datasets. We find that 92.65% of the (scene, question) pairs in the unbalanced test set do not have a corresponding complementary scene (where the answer to the same question is the opposite). Only 20.48% of our balanced test set does not have corresponding complementary scenes. Note that our dataset is not 100% balanced either because there are some scenes which could not be modified to flip the answers to the questions (5.93%) or because the most common answer out of 10 human annotated answers for some questions does not match with the intended answer of the person creating the complementary scene (14.55%) either due to inter-human disagreement, or if the worker did not succeed in creating a good scene.

4.4 Approach

We present an overview of our approach before describing each step in detail in the following subsections. To answer binary questions about images, we propose a two-step approach: (1) Language Parsing: where the question is parsed into a tuple, and (2) Visual Verification: where we verify whether that tuple is present in the image or not.

Our language parsing step summarizes a binary question into a tuple of the form <P, R, S>, where P refers to primary object, R to relation and S to secondary object, e.g. for a binary question "Is there a cat in the room?", our goal is to extract a tuple of the form: <cat, in, room>. Tuples need not have all the arguments present. For instance, "Is the dog asleep" → <dog, asleep, >, The primary argument P is always present. Since we only focus on binary questions, this extracted tuple captures the entire visual concept to be verified in the image. If the concept is depicted in the image, the answer is "yes", otherwise the answer is "no".

Once we extract <P, R, S> tuples from questions (details in Section 4.4.1), we align the P and S arguments to objects in the image (Section 4.4.2). We then extract text and image features (Section 4.4.4), and finally learn a model to reason about the consistency of the tuple with the image (Section 4.4.3).

4.4.1 Tuple extraction

In this section, we describe how we extract <P, R, S> tuples from raw questions. Existing NLP work such as [38] has studied this problem, however, these approaches are catered towards statements, and are not directly applicable to questions.

Parsing: We use the Stanford parser to parse the question. Each word is assigned an entity, e.g. nominal subject ("nsubj"), direct object ("dobj"), etc. We remove all characters other than letters and digits before parsing.

Summarizing: As an intermediate step, we first convert a question into a "summary", before converting that into a tuple. First, we remove a set of "stop words" such as determiners ("some", "the", etc.) and auxillary verbs ("is", "do", etc.). Next, following common NLP practice, we remove all words before a nominal subject ("nsubj") or a passive nominal subject ("nsubjpass"). For example, "Is the woman on couch petting the dog?" is parsed as "Is(aux) the(det) woman($nsubj$) on($case$) couch($nmod$) petting($root$) the(det) dog($dobj$)?".

The summary of this question can be expressed as (woman, on, couch, petting, dog).

Extracting tuple: Now that we have extracted a summary of each question, next we split it into PRS arguments. Ideally, we would like P and S to be noun phrases ("woman on couch", "dog") and the relation R to be a verb phrase ("petting") or a preposition ("in") when the verb is a form of "to be". For example, <dog, in, room>, or <woman on couch, petting, dog>. Thus, we apply the Hunpos Part of Speech (POS) tagger [57] to assign words to appropriate arguments of the tuple.

4.4.2 Aligning objects to primary (P) and secondary (S) arguments

In order to extract visual features that describe the objects in the scene being referred to by P and S, we need to align each of them with the image.

We extract PRS tuples from all binary questions in the training data. Among the three arguments, P and S contain noun phrases. To determine which objects are being referred to by the P and S arguments, we follow the idea in [158] and compute the mutual information[2] between word occurrence (e.g. "dog"), and object occurrence (e.g. clipart piece #32). We only consider P and S arguments that occur at least twice in the training set. At test time, given an image and a PRS tuple corresponding to a binary question, the object in the image with the highest mutual information with P is considered to be referred by the primary object, and similarly for S. If there is more than one instance of the object category in the image, we assign P/S to a random instance. Note that for some questions with ground-truth answer 'no', it is possible that P or S actually refers to an object that is not present in the image (e.g. Question: "Is there a cat in the image?" Answer: "no"). In such cases, some other object from images (say clipart #23, which is a table) will be aligned with P/S.

[2]We compute MI separately for indoor and outdoor scenes. More details about scene types can be found in Section 4.3.1.

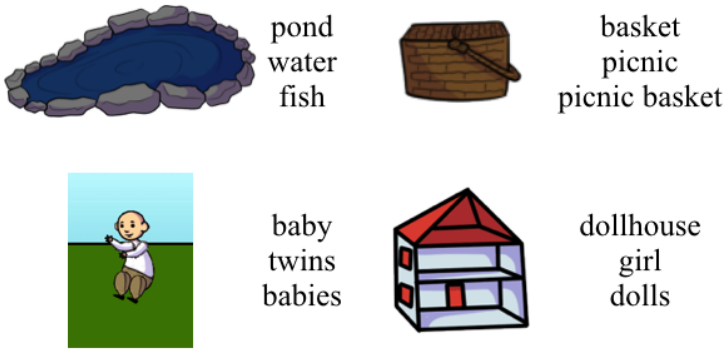


Figure 4.3: Most plausible words for an object determined using mutual information.

However, since the category label ('table') of the aligned object is a feature, the model can learn to handle such cases, i.e., learn that when the question mentions 'cat' and the aligned clipart object category is 'table', the answer should be 'no'.

We found that this simple mutual information based alignment approach does surprisingly well. This was also found in [158]. Figure. 4.3 shows examples of clipart objects and three words/phrases that have the highest mutual information.

4.4.3 Visual verification

We have extracted PRS tuples and aligned PS to the clipart objects in the image, we can now compute a score indicating the strength of visual evidence for the concept inquired in the question. Our scoring function measures compatibility between image and text features (described in Section 4.4.4).

Our model is an ensemble of two similar models– Q-model and Tuple-model, whose common architecture is inspired from a recently proposed VQA approach [4]. Specifically, each model takes two inputs (image and question), each along a different branch. The two models (Q-model and Tuple-model) use the same image features, but different language features. Q-model encodes the sequential nature of the question by feeding it to an LSTM and using its 256-dim hidden representation as a language embedding, while Tuple-model focuses on

the important words in the question and uses concatenation of word2vec [101] embeddings (300-dim) of P, R and S as the language features. If P, R or S consist of more than one word, we use the average of the corresponding word2vec embeddings. This 900-dimensional feature vector is passed through a fully connected layer followed by a *tanh* non-linearity layer to create a dense 256-dim language embedding.

The image is represented by rich semantic features, described in Section 4.4.4. Our binary VQA model converts these image features into 256-dim with an inner-product layer, followed by a *tanh* layer. This fully-connected layer learns to map visual features onto the space of text features.

Now that both image and text features are in a common space, they are point-wise multiplied resulting in a 256-dim *fused* language+image representation. This fused vector is then passed through two more fully-connected layers in a Multi-Layered Perceptron (MLP), which finally outputs a 2-way softmax score for the answers 'yes' and 'no'. These predictions from the Q-model and Tuple-model are multiplied to obtain the final prediction. Both the models are learned separately and end-to-end (including LSTM) with a cross-enptropy loss. Our implementation uses Keras [1]. Learning is performed via SGD with a batch-size of 32, dropout probability 0.5, and the model is trained till the validation loss plateaus.

At test time, given the question and image features, we can perform visual verification simply by performing forward pass through our network.

4.4.4 Visual Features

We use attention-based as well as holistic visual features that describe the entire scene layout.

Attention-based features. Our attention-based features have been derived from [92]. These visual features describe the objects in the image that are being referred to by the P and S arguments, their interactions, and the context of the scene within which these objects are present. In particular, the feature vector for each scene has 1432 dimensions, which

are composed of 563 dimensions for each primary object and secondary object, encoding object category (e.g., cat vs. dog vs. tree), instance (e.g., which particular tree), flip (i.e., facing left or right), absolute location modeled via GMMs, pose (for humans and animals), expression, age, gender and skin color (for humans), 48 dimensions for relative location between primary and secondary objects (modeled via GMMs), and 258 dimensions encoding which other object categories and instances are present in the scene around P and S.

Holistic features. The holistic features include a bag-of-words for clipart objects occurrence (150-dim), human expressions (8-dim), and human poses (7-dim). The 7 human poses refer to 7 clusters obtained by clustering all the human pose vectors (concatenation of (x, y) location and global angles of all 15 deformable parts of human body) in the training set. We extract these 165-dim holistic features for the complete scene and for four quadrants, and concatenate them together to create a 825-dim vector. These holistic image features are similar to decaf features for real images, which are good at capturing what is present where, but (1) does not attend to different parts of the image based on the questions, and (2) may not be capturing intricate interactions between objects.

4.5 Experiments

4.5.1 Baselines

We compare our models with several strong baselines including language-only models as well as a state-of-the-art VQA method.

PRIOR: Predicting the most common answer in the training set, for all test questions. The most common answer is "yes" in the unbalanced set, and "no" in the balanced set.

Blind-Q+Tuple: A language-only baseline which has a similar architecture as our approach except that each model only accepts language input and does not utilize any visual information. Comparing our approach to Blind-Q+Tuple quantifies to what extent our model has

succeeded in leveraging the image to answer questions correctly.

SOTA Q+Tuple+H-IMG: This VQA model has a similar architecture as our approach, except that it only uses holistic image features (described in Section 4.4.4) instead of focusing on specific regions in the scene as determined by P and S. This model is analogous to the state-of-the-art models presented in [4, 50, 98, 115], except applied to abstract scenes.

Comparing our model to SOTA Q+Tuple+H-IMG quantifies the improvement in performance by attending to specific regions in the image as dictated by the question being asked , and explicitly capturing the interactions between the relevant objects in the scene. In other words, we quantify the improvement in performance obtained by pushing for a deeper understanding of the image than generic global image descriptors. Thus, we name our model Q+Tuple+A-IMG, where A is for attention.

4.5.2 Evaluation on the original (unbalanced) dataset

In this subsection, we train all models on the train splits of both the unbalanced and balanced datasets, and test on our unbalanced test set. The results are shown in Table 4.1.

	Training set	
	Unbalanced	Balanced
PRIOR ("yes")	68.67	68.67
Blind-Q+Tuple	78.90	60.80
SOTA Q+Tuple+H-IMG	79.40	70.98
Ours Q+Tuple+A-IMG	**79.68**	**75.07**

Table 4.1: Evaluation on unbalanced test set. All accuracies are calculated using the VQA [4] evaluation metric.

We draw the following key inferences:

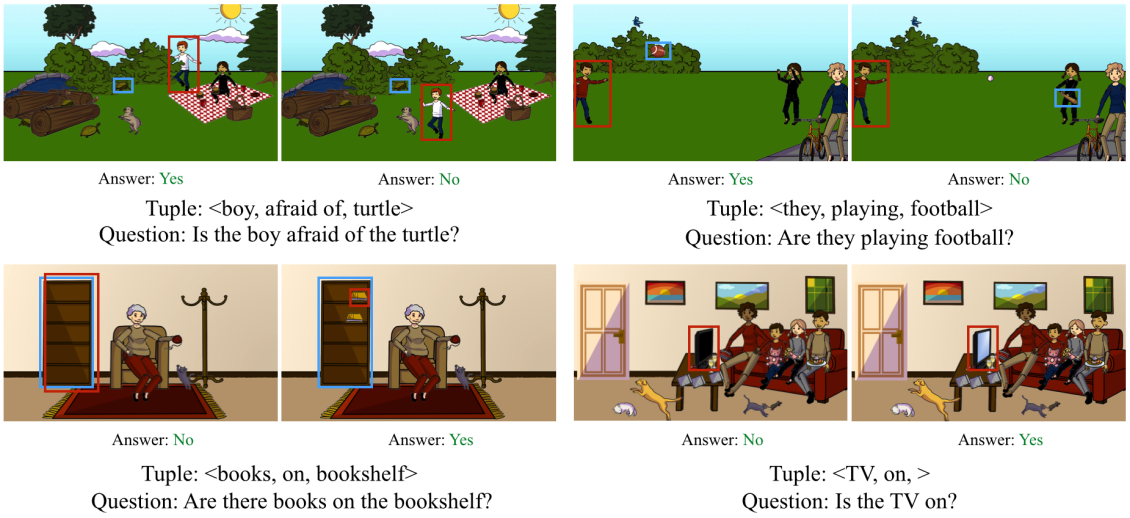


Figure 4.4: Qualitative results of our approach. We show input questions, complementary scenes that are subtle (semantic) perturbations of each other, along with tuples extracted by our approach, and objects in the scenes that our model chooses to attend to while answering the question. Primary object is shown in red and secondary object is in blue.

Vision helps. We observe that models that utilize visual information tend to perform better than "blind" models, especially when trained on the balanced dataset. This is because the lack of strong language priors in the balanced dataset forces the models to focus on the visual understanding.

Attending to specific regions is important. When trained on the balanced set where visual understanding is critical, our proposed model Q+Tuple+A-IMG, which focuses on only a specific region in the scene, outperforms all the baselines by a large margin.

Bias is exploited. As expected, the performance of all models trained on unbalanced data is better than the balanced dataset, because these models learn the language biases while training on unbalanced data, which is also present in the unbalanced test set.

4.5.3 Evaluation on the balanced dataset

We also evaluate all models trained on the train splits of both the unbalanced and balanced datasets, by testing on the balanced test set. The results are summarized in Table 4.2.

	Training set	
	Unbalanced	Balanced
PRIOR ("no")	63.85	63.85
Blind-Q+Tuple	65.98	63.33
SOTA Q+Tuple+H-IMG	67.47	71.28
Ours Q+Tuple+A-IMG	**67.48**	**75.52**

Table 4.2: Evaluation on balanced test set. All accuracies are calculated using the VQA [4] evaluation metric.

Here are the observations from this experiment:

Training on balanced is better. It is clear from Table 4.2 that both language+vision models trained on balanced data perform better than the models trained on unbalanced data. This may be because the models trained on balanced data *have* to learn to extract visual information to answer the question correctly, since they are no longer able to exploit language biases in the training set. Where as models trained on the unbalanced set are blindsided into learning strong language priors, which are then not available at test time.

Blind models perform close to chance. As expected, when trained on unbalanced dataset, the "blind" model's performance is significantly lower on the balanced dataset (66%) than on unbalanced (79%). Note that the accuracy is higher than 50% because this is not binary classification accuracy but the VQA accuracy [4], which provides partial credit when there is inter-human disagreement in the ground-truth answers.

Attention helps. When trained on balanced dataset (where language biases are absent),

our model Q+Tuple+A-IMG is able to outperform all baselines by a large margin. Specifically, our model gives improvement in performance relative to the state-of-the-art VQA model from [4] (Q+Tuple+H-IMG), showing that attending to relevant regions and describing them in detail helps, as also seen in Section 4.5.2.

Role of balancing. We see clear improvements by reasoning about vision in addition to language. Note that in addition to the lack of language bias, the visual reasoning is also harder on the balanced dataset because now there are pairs of scenes with fine-grained differences but with opposite answers to the same question. So the model really needs to understand the subtle details of the scene to answer questions correctly. Clearly, there is a lot of room for improvement and we hope our balanced dataset will encourage more future work on detailed understanding of visual semantics towards the goal of accurately answering questions about images.

Classifying a pair of complementary scenes. We experiment with an even harder setting – a test point consists of a pair of complementary scenes and the associated question. Recall, that by construction, the answer to the question is "yes" for one image in the pair, and "no" for the other. This test point is considered to be correct only when the model is able to predict both its answers correctly.

Since language-only models only utilize the textual information in the question ignoring the image, and therefore, predict the same answer for both scenes, their accuracy is zero in this setting[3]. The results of the baselines and our model, trained on balanced and unbalanced datasets, are shown in Table 4.3. We observe that our model trained on the balanced dataset performs the best. And again, our model that focuses on relevant regions in the image to answer the question outperforms the state-of-the-art approach of [4] (Q+Tuple+H-IMG) that does not model attention.

[3]Note that to create this pair-level test set, we only consider those pairs where the answers were opposites. We removed all scenes that workers were unable to create complementary scenes for due to a finite clipart library, as well as those scenes for which the majority answer from 10 workers did not agree with the intended answer of the creator of the scene.

	Training set	
	Unbalanced	Balanced
Blind-Q+Tuple	0	0
Q+Tuple+H-IMG	15.17	26.76
Q+Tuple+A-IMG	**16.19**	**37.58**

Table 4.3: Classifying a pair of complementary scenes. All accuracies are percentage of test pairs that have been predicted correctly.

4.5.4 Analysis

Our work involves three steps: tuple extraction, tuple and object alignment, and question answering. We conduct analyses of these three stages to determine the importance of each of the three stages. We manually inspected a random subset of questions, and found the tuple extraction to be accurate 86.3% of the time. Given perfect tuple extraction, the alignment step is correct 95% of the time. Given perfect tuple extraction and alignment, our approach achieves VQA accuracy of 81.06% as compared to 79.68% with imperfect tuple extraction and alignment. Thus, ∼1.5% in VQA accuracy is lost due to imperfect tuple extraction and alignment.

4.5.5 Ablation Study

We conducted ablation studies to analyze the importance of the two kinds of language features– LSTM for question vs. word2vec for tuple, as well as to compare attention-based vs. holistic image features across dataset distributions. For "blind" (language only) models trained and tested on unbalanced datasets, we found that the combination (Q+Tuple) performs better than each of the individual methods. Specifically, Q+Tuple achieves a VQA accuracy of 78.9% as compared to 77.87% (Q-only) and 77.54% (Tuple-only). Similarly, for our

models trained and tested on balanced datasets, the combination of attention-based+holistic image features performs better (VQA accuracy of 75.52%) as compared to only attention-based (74.65%) and only holistic (71.28%) image features.

4.5.6 Qualitative results

Figure 4.4 shows qualitative results for our approach. We show a question and two complementary scenes with opposite answers. We find that even though pairs of scenes with opposite ground truth answers to the same questions are visually similar, our model successfully predicts the correct answers for both scenes. Further, we see that our model has learned to attend to the regions of the scene that seem to correspond to the regions that are most relevant to answering the question at hand. The ability to (correctly) predict different answers to scenes that are subtle (semantic) perturbations of each other demonstrates visual understanding.

4.6 Discussion

The idea of balancing a dataset can be generalized to real images. For instance, we can ask MTurk workers to find images with different answers for a given question. The advantage with clipart is that it lets us make the complementary scenes very fine-grained forcing the models to learn subtle differences in visual information. The differences in complementary real images will be coarser and therefore easier for visual models. Overall, there is a trade-off between clipart and real images. Clipart is easier (trivial) for low-level recognition task, but hard because it can introduce fine-grained semantic differences. Real is harder for low-level recognition task, but may be an easier balanced dataset because it will have coarse semantic differences.

4.7 Role of language priors

In this section, we analyze the language priors present in the (unbalanced) abstract VQA dataset [4]. To do this, we implemented an n-gram baseline model to predict answers. In our implementation, we used n = 4[4]. During training, the model extracts all n-grams that start the questions in the training set, and remembers the most common answer for each n-gram. At test time, the model extracts an n-gram from the beginning of each test question, and predicts the most common answer corresponding to the extracted n-gram for that question. In case, the extracted n-gram is not present in the model's n-gram memory, the model predicts the answer "yes" (which has a higher prior than "no").

We found that this model is able to achieve a VQA accuracy of 75.11%, which is substantially high as compared to All YES baseline accuracy of 68.67%. Qualitatively, the most frequently occurring n-gram in validation split of VQA dataset, "is the little girl" (135 out of total 12,321 questions) has an accuracy of 82.59% with the answer "yes". Similarly, the n-grams "is there a fire" (61 questions) and "is the young man" (35 questions) have accuracies of 97.21% and 91.28% respectively with the answer "yes". In some cases, the bias is towards answering "no". For instance, the n-gram "are there leaves on" has an accuracy of 98.18% by predicting "no" for all 22 questions it occurs in. This example clearly demonstrates that humans tend to ask about leaves on trees only when the trees in the images do not have them. There are many more images which contain trees full of leaves, but humans don't ask such questions for those images.

In some cases, the bias in the dataset is because of the limited clipart library. For instance, the predicted answer "no" is always correct for the n-grams "is the door open?" and "is it raining?" because the clipart library has no open doors, and has no rain. Similarly, the n-gram "is it daytime?" gets 100% accuracy with the answer "yes".

We believe that LSTMs, known to perform well at remembering sequences, are able to exploit

[4]If question has less than 4 words, we use n = length of question.

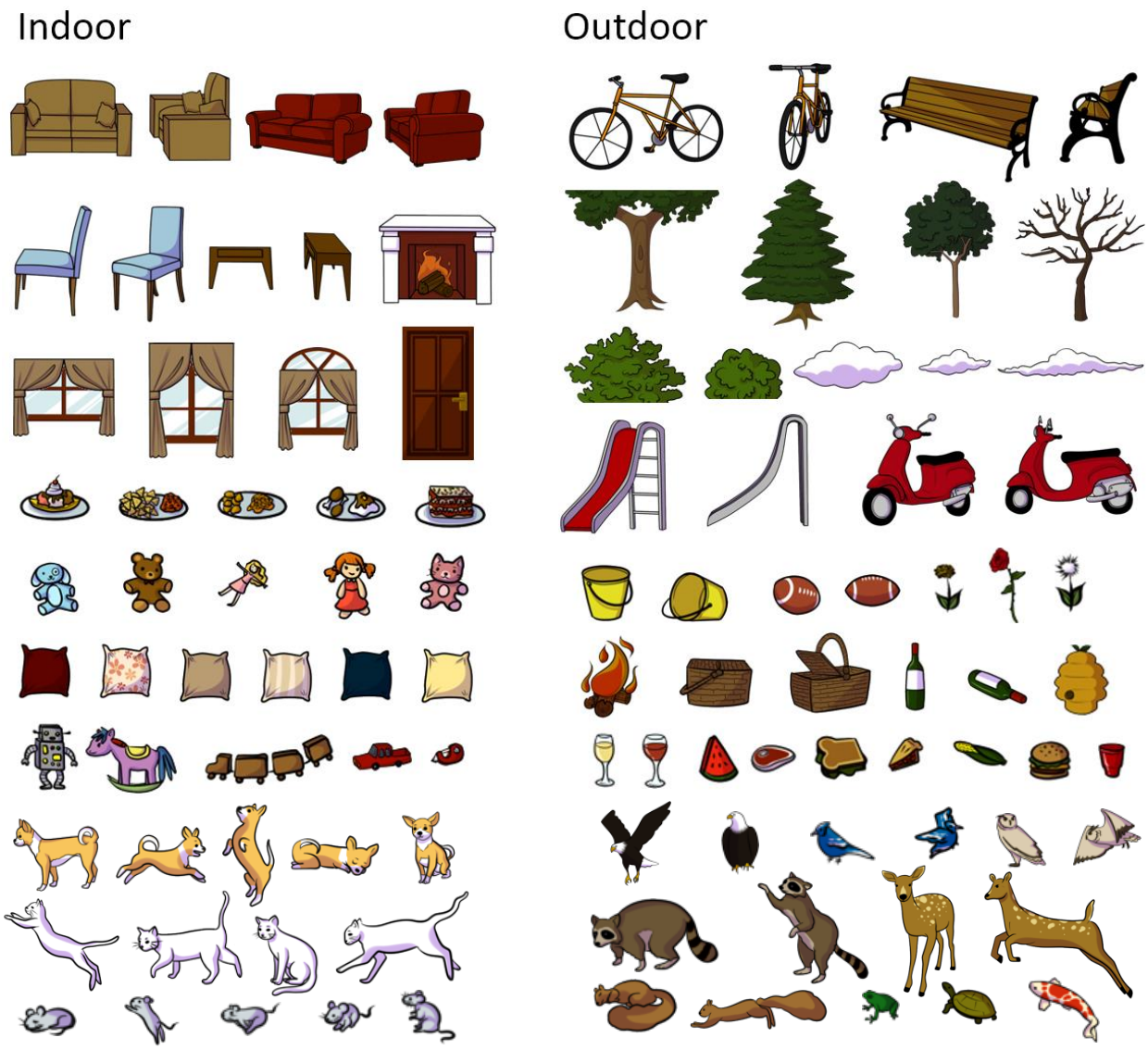


Figure 4.5: A subset of the clipart objects present in the abstract scenes library.

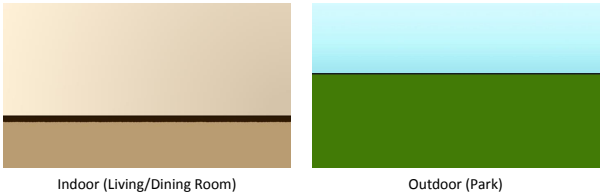


Figure 4.6: Backgrounds in indoor (left) and outdoor (right) scenes.

Modify the clipart scene so that the answer to the question changes! (Living/Dining Room)

[Images may take some time to load] **[Spamming will get blocked]**

Please read instructions! Your work will be rejected if you don't follow the instructions correctly.

Hide Instructions

We will show you a clipart scene and an associated question.

We will ask you to change the clipart scene so that the answer to the question changes either from **"yes"** to **"no"**, or from **"no"** to **"yes"**. **Each scene will have a different requirement.** So please **look at the instruction above each scene** carefully.

Important Instructions

1. Modify the scene **as little as possible**. Just sufficient to change the answer. Try modifications in the following order:
   - change facial expressions of people, flip objects, or change their poses
   - move objects
   - add or remove objects
2. The modified scene should still be **realistic**.
3. The question should **still be a valid question** for the scene. For example, if a scene contains a woman sitting on the couch, the question is "is the woman sitting on a couch?", and the answer has to be changed from "yes" to "no", then you can't just remove the woman from the scene to change the answer, because then the question "is the woman sitting on a couch?" doesn't make sense for the new scene that doesn't even have a woman. Instead, you would need to move the woman somewhere else (change her pose to standing, or make her sit on something else in a realistic fashion).
4. On the rare occasion where the **person writing the question clearly made a mistake** when looking at the scene, if it is easy to fix the scene so the question makes sense, please do so. Note that we don't expect this to happen very frequently, if at all.
5. If because of the limited clipart library, it is **not possible at all** to modify the scene to change the answer while making sure the question is valid, only then click on the "Not possible" button.

If you create unrealistic scenes, or if the new answer as instructed is not correct for the modified scene, or if you modify the scene more than necessary, or if the question doesn't make sense for the new scene you created, **your work will be rejected**. Also, please **do not create potentially offensive (e.g., sexually suggestive) scenes**.

Clipart objects (5 instances each) may be added by dragging them onto the scene and removed by dragging them off. In addition to the buttons, there are also some keyboard commands to adjust the objects:

Option	Command
Smaller	(CTRL + a)
Larger	(CTRL + z)
Flip	(CTRL + c)
Send backward	(CTRL + s)
Bring forward	(CTRL + x)

You will be asked to work on 5 **different/independent** scenes. You can go back and forth between all of your scenes by pressing "Prev" and "Next". When you finish your last one, a pop-up will ask you to submit. We'd love to hear any feedback you have about the usability of the interface, any bugs you encounter, or the HIT in general, so feel free to leave a comment.

Thanks for your work!

Figure 4.7: The instructions given to AMT workers while collecting complementary scenes.

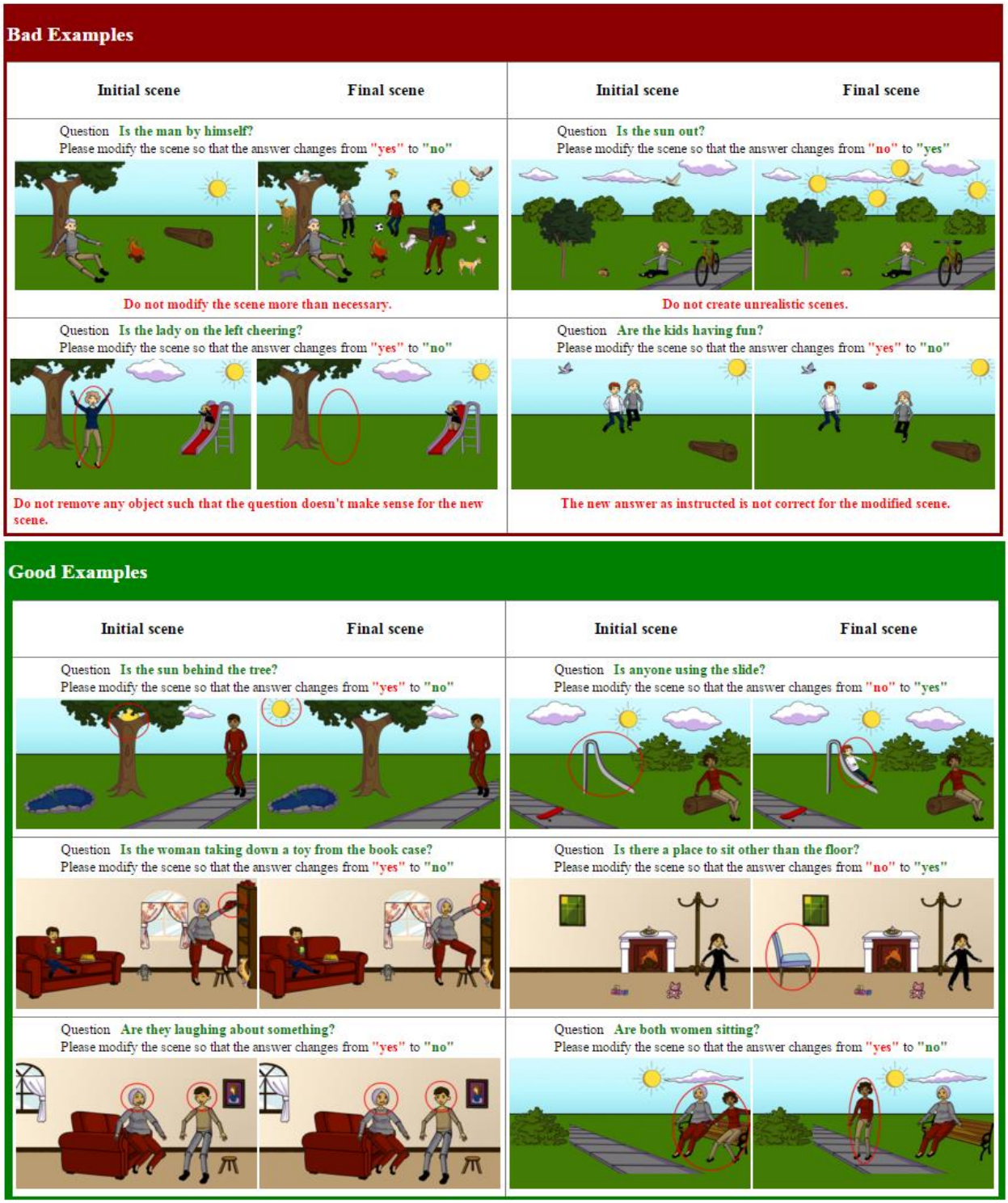


Figure 4.8: The good and bad examples shown to AMT workers while collecting complementary scenes.

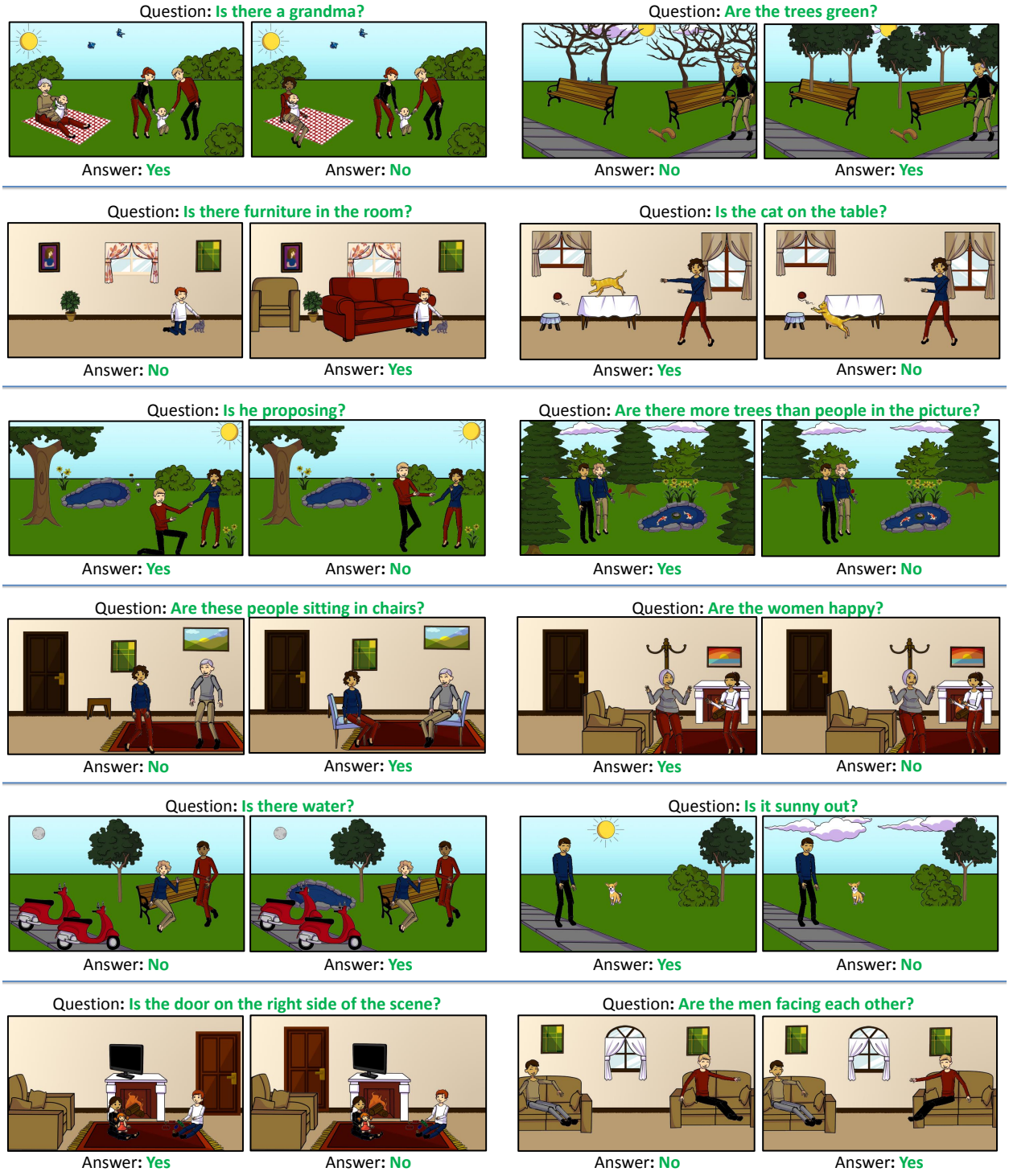


Figure 4.9: Example complementary scenes from our balanced dataset. For each pair, the left scene is from VQA dataset [4], and the right scene is the modified version created by AMT workers to flip the answer to the given question.

this dataset bias by remembering the most common answer for such n-grams. Moreover, LSTMs get even higher accuracy (VQA accuracy of 78.3%) than our n-gram baseline because they probably learn to remember meaningful words in the question instead of the first n words.

4.8 Abstract library

The abstract library consists of two scene types- indoor ("Living/Dining room") and outdoor ("Park") scenes. Each scene type is indicated by a different background as shown in Fig. 4.6, and consists of clipart objects which fit in the respective setting. For instance, indoor scenes can contain indoor objects such as "pillow", "TV", "fireplace", etc., while outdoor scenes can contain outdoor objects such as "eagle", "bike", "football", etc. All clipart objects are categorized into 4 types –

1) "human": There are 20 paperdoll human models spanning genders, 3 different races and 5 different ages including 2 babies. Each human model can take any one of the available 8 expressions. The limbs are adjustable to allow continuous pose variations. All human models are present in both scene types.

2) "animal": There are 10 animal models in indoor scenes, and 31 in outdoor scenes. To keep the scenes realistic, wild animals such as "deer", "raccoon", "eagle", etc. are not available to create indoor scenes. Some of the animal models have been shown in Fig. 4.5.

3) "large object": There are 23 large objects present in indoor scenes (e.g. "door", "fireplace", etc.), and 17 large objects in outdoor scenes (e.g. "tree", "cloud", etc.). A subset of large objects have been shown in Fig. 4.5.

4) "small object": There are 40 small objects present in indoor scenes (e.g. "toy", "pillow", etc.), and 34 small objects in outdoor scenes (e.g. "pail", "flower", etc.). A subset of small objects have been shown in Fig. 4.5.

4.9 Dataset collection

Full instructions of our Amazon Mechanical Turk (AMT) interface to collect complementary scenes, can be seen in Fig. 4.7. To make the task more clear, we also show some good and bad examples (Fig. 4.8) to AMT workers. Finally, our AMT interface has been shown in Fig. 4.2.

Some of the complementary scenes from our balanced dataset have been shown in Fig. 4.9.

4.10 Qualitative results

We show some qualitative results of our approach in Fig. 4.10, including some failure cases. In each scene, the primary and secondary objects have been marked in red and blue boxes respectively.

4.11 Issue of Negation

Since our approach focuses on the meaningful words in the question, it leads to the issue of poorly handling negative questions. For example, for the questions "Is the cat on the ground?" and "Is the cat not on the ground?", the extracted tuple would be the same <cat, on, ground>, but their answers should ideally be opposite. This problem of negation is hard to deal with even in NLP research, but is often ignored because such negative sentences are rarely spoken by humans. For example, in the VQA training dataset, less than 0.1% of the binary questions contain the word "not", "isn't", "aren't", "doesn't", "don't", "didn't", "wasn't", "weren't", "shouldn't", "couldn't", or "wouldn't".

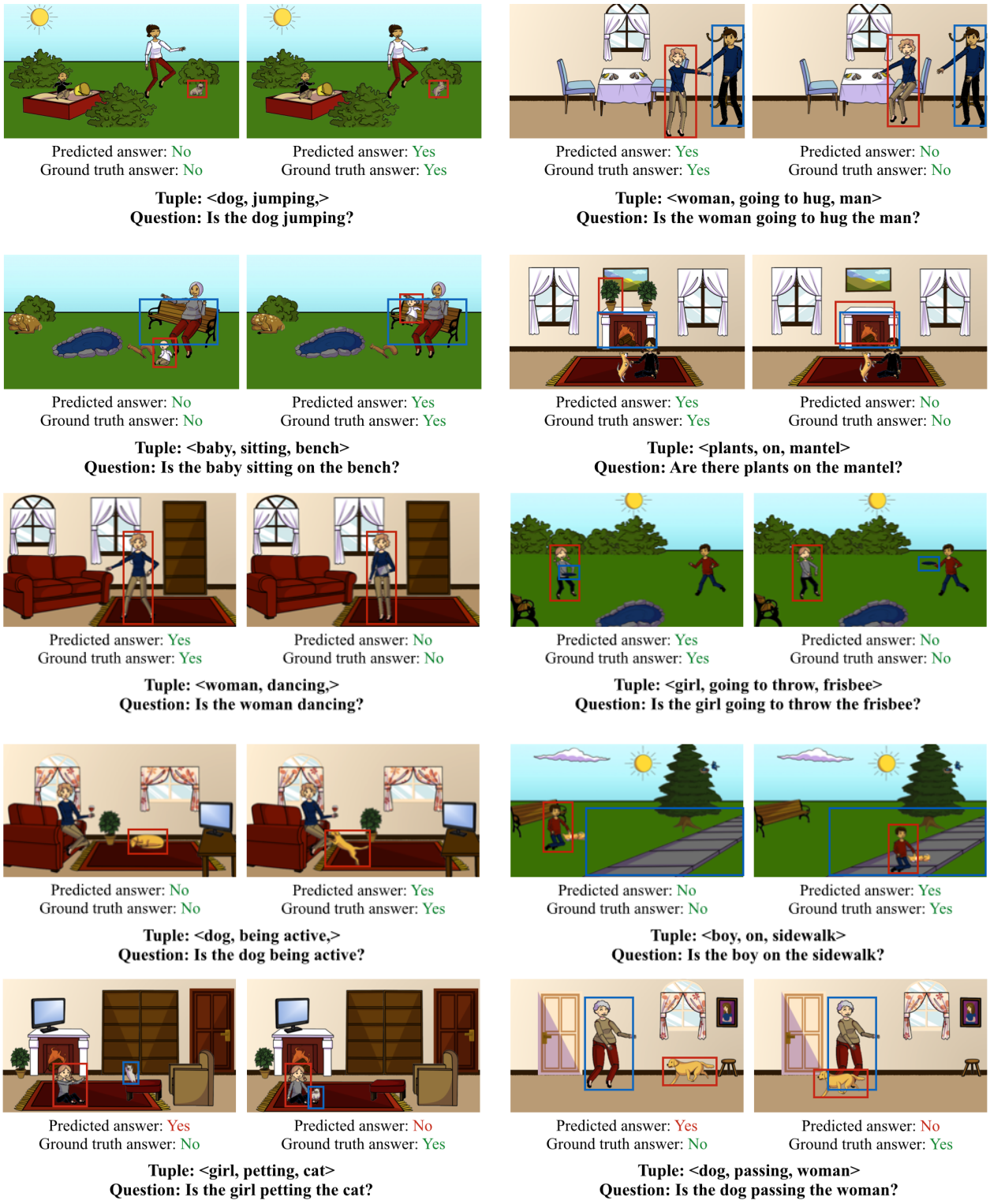


Figure 4.10: Some qualitative results of our approach. The last row shows failure cases. The primary and secondary objects have been shown in red and blue boxes respectively.

4.12 Image features

The image features in our approach are composed of the following 4 parts:

- primary object (P) features (563 dimensions)
- secondary object (S) features (563 dimensions)
- relative location features between P and S (48 dimensions)
- scene-level features (258 dimensions)

P and S features consist of the following parts:

- category ID (4 dimensions): category that the object belongs to – human, animal, large object or small object
- instance ID (254 dimensions): instance index of the object in the entire clipart library
- flip attribute (1 dimension): facing left or right
- absolute locations (50 dimensions): modeled via Gaussian Mixture Model (GMM) with 9 components in 5 different depths separately
- human features (244 dimensions): composed of age (5 dimensions), gender (2 dimensions), skin color (3 dimensions), pose (224 dimensions), and expressions (10 dimensions)
- animal features (10 dimensions): pose occurrence for 10 possible discrete poses

Relative location feature is modeled via another GMM, which is composed of 24 different components.

Scene level features contain the presence of the other objects, i.e. which object is present in the scene, which is not.

4.13 Details of tuple extraction from raw binary questions

4.13.1 Pre-processing

We first do the following pre-processing on the raw questions:

1. We only keep the letters from a to z (both lower cases and upper cases) and digits.
2. We drop some phrases – "do you think", "do you guess", "can you see", "do you see", "could you see", "in the picture", "in this picture", "in this image", "in the image", "in the scene", "in this scene", "does it look like", "does this look like", "does he look like", and "does she look like" – to avoid extra non-meaningful semantic relations in questions.
3. We make all the letters lower case, but capitalize the first letter. Then we add a question mark at the end of each question.

4.13.2 Summarization

As a first step, we parsed the processed question from Section 4.13.1 using the Stanford parser [17], resulting in each word in the question being assigned a grammatical entity such as nominal subject ("nsubj"), adverb modifier ("admod"), etc. Then we follow these steps:

1. We create a list of all the entities that we would like to keep. The full list is: entity list = ["nsubj", "root", "nsubjpass", "case", "nmod", "xcomp", "compound", "dobj", "acl", "advmod", "ccomp", "advcl", "nummod", "dep", "amod", "cc"]. In cases where there are more than one word that has been assigned to the same entity, we assign the words different names, for example, "nsubj" and "nsubj-1".

2. As we discussed in the main paper, the extracted summary usually starts with entity nominal subject ("nsubj") or passive nominal subject ("nsubjpass"). So from each parsing result, we first check if "nsubj" or "nsubjpass" exists. We would like the words that have entity "nsubj" or "nsubjpass" to be nouns or pronouns. Therefore, we run POS tagger on the whole sentence, specifically we use HunposTagger [57].

   If the word assigned as "nsubj" or "nsubjpass" is tagged as noun, then we drop all the words before it. Otherwise, we search for nouns or pronouns from nearby words.

   If the word with entity "nsubj" (or "nsubjpass") is tagged as pronoun, we would only like to keep more meaningful pronouns such as "he", "she", etc., rather than "it", "this" etc. So, we created a stop word list for pronouns: ["it", "this", "that", "the"]. Therefore, if the word with "nsubj" or "nsubjpass" entity is tagged as pronoun and is not present in the above stop list, we keep the question fragments from there.

   Example1: Given the question "Are the children having a good time?", the parser assigns the word "children" as "nsubj" and Hunpos tagger tags it as a noun. So we drop all the words before it and output "children having a good time".

   Example2: Given the question "Is she playing football?", the word "she" is assigned entity "nsubj" by Stanford parser, and is tagged as a pronoun by Hunpos tagger. Then we verify its absence from the stop word list for pronouns. Therefore, we drop all words before it, resulting in "she playing football" as summary.

3. Cases where there is neither "nsubj" nor "nsubjpass" in the question, fall into one of the following three cases: 1) starting with entity "root", 2) starting with entity "nmod", 3) starting with entity "nummod". We directly look for the first noun or pronoun (not in the stop word list) and drop words before it.

   Example: Given the question "Are there leaves on the trees?", there is no "nsubj" or "nsubjpass" in it. The word "leaves" is tagged as "root" and it is the first noun. So, we drop all the words before "leaves" and output "leaves on the trees".

4. Now that we have fragments of the sentence, we check the entity for each word and delete words whose entity is not in the entity list (described in step 1).

   Example: Given a binary question: "Is the girl pushing the cat off the stool?", the parsing result from Stanford parser is: Is (aux) the (det) girl (nsubj) pushing (root) the (det) cat (dobj) off (case) the (det) stool (nmod)? First, we search for "nsubj", and find the word "girl". POS tagger tags it as "PRP", which means pronoun, so we drop all the words before it. This results in the fragments "girl pushing the cat off the stool". Lastly, we check if there are any words with entities that are not in the entity list and delete the word "the" (entity is "det"). Therefore, we have the extracted summary as "girl pushing cat off stool".

4.13.3 Tuple extraction

Now that we have the extracted summaries from raw binary questions, we would like to split them into tuples in the form of primary object (P), relation (R) and secondary object (S). We will describe each of them separately.

Splitting P argument

1. From the extracted summary, we first look for words with entity "nsubj" or "nsubjpass". If it exists, then we split all the words from the beginning until it as P.

   Example: If the extracted summary is "girl pushing cat off stool", the word "girl" has entity "nsubj", so we make P = "girl".

2. In some cases, we would like to have a noun phrase instead of one noun word as P. For example, if the summary is "lady on couch", we would like to have "lady" as P, while for "lady on couch petting dog", we would like to split "lady on couch" as P. In the later case, the summary has the common structure: subject word (e.g. lady)

+ preposition word (e.g. on) + noun word (e.g. couch). Usually, noun words in two categories can be used: "real object" category, which refer to objects in clipart library, for example, desk, chair etc., or "location" category, for example, middle, right, etc. Therefore, we created two such lists, as shown below:

Real object list = ["toy", "bird", "cup", "scooter", "bench", "bush", "bike', "dining chair", "plate", "bluejay', "cat', "blanket", "dollhouse", "yarn", "watermelon", "pillow", "bread", "bat", "monkey bars", "slide", "pet bed", "stool", "frog", "seesaw", "sandwich", "tape", "finch", "picture", "flower", "door", "sun", "rug", "moon", "campfire', "rabbit", "utensil", "sofa", "corn", "chair", "baseball", "butterfly", "sidewalk", "turtle", "steak", "doll", "coat rack", "mouse", "ribs", "skateboard", "end table", "paper", "rat", "koi", "cheese", "shovel", "camera", "apple", "marshmallow", "pigeon", "book", "lilypad", "cloud", "log", "stapler', "notebook","bookshelf", "dog", "hawk", "fireplace", "raccoon", "footstool", "mushroom", "pie", "building toy", "tea set", "bottle", "duck", "grill", "soccer", "tree", "pen", "cd", "game system", "scissors", "lily pad" "hamburger", "puppy", "couch", "pond", "window", "eagle", "plant", "squirrel", "tv", "dining table", "desk", "robin", "frisbee", "pail", "pencil", "nest", "football", "kitten", "bee", "owl", "bone", "chipmunk", "deer", "tongs", "beehive", "sandbox", "bottle", " basket", "table", "bed", "bar", "pad", "shelf", "house", "ground", "cartoon", "rope", "footstool"]

location list = ["left", "right", "center", "top", "front", "middle", "back"]

If we find a word tagged as "nsubj" or "nsubjpass" by Stanford parser, and is tagged as noun by POS tagger, we check if the next word has entity "IN" (meaning proposition word), and the next (or two) word(s) after that are belonging to either the object word list or location word list. Lastly, we check if this is the end of the summary, if not, then we group all the words till here as P, otherwise, we only assign "nsubj" or "nsubjpass" as P.

Example:, If the extracted summary is "lady on couch petting dog", the entity "nsubj"

corresponds to the word "lady", and the next word "on" is tagged as "IN", followed by the word "couch", which is in the object list. Moreover, we check that the word "couch" is not the last word of the summary. So we assign "lady on couch" to P.

3. If we could not find "nsubj" or "nsubjpass" in the summary, we directly look for nouns in the summary. And we assign all the consecutive nouns to P.

   Example: For the question "Is it night time?", the extracted summary is "night time", in which both the words are nouns. So we assign both the words to P.

Splitting S argument

1. We drop the words included in P from the extracted summary, and look for nouns in the remaining summary. Once we locate a noun, all the words after it are kept as S.

   Example: If the extracted summary is "lady on couch petting dog", we assign "lady on couch" to P, so we drop them from the summary and thus, we are left with "petting dog" from the summary. "petting" is not a noun, so we move to the next word. And "dog" is a noun. We stop here and make everything after it as S, which in this case is only "dog". So S = "dog".

2. In some cases, there are some words which modify the nouns, for example, "pretty girl", "beautiful flowers", etc.. In such cases, we would like the adjectives to be included with the nouns in S. To do so, we look for adjectives, which are tagged by POS tagger, and if there is a noun after this, we keep both of them as S. Otherwise, we only keep the nouns as S.

   Example: If the summary is "scooters facing opposite directions", P = "scooters", so we drop it from the summary, this leaves us with "facing opposite directions". "opposite" is an adjective and after it, "directions" is a noun. SO we keep "opposite directions" as S.

3. A minor special case is the occurrence of the phrases – "in front of" and "having fun with". These are two phrases that we qualitatively noticed cause issues. In these cases,

"front" and "fun" are tagged as nouns, which confuses our system. So if we detect "in front of" or "having fun with", we skip them and move to the next word.

Splitting R argument

Since we have already split P and S, we simply assign everything else left in the summary as R.

Example: If the extracted summary is "lady on couch petting dog", P = "lady on couch" and S = "dog", thus we have R = "petting".

4.14 Definition of some Stanford typed entities

We used the Stanford parser in this work. To have better understanding of Section 4.13.2, we list some of the Stanford parser entities here for reference.

nsubj: nominal subject A nominal subject is a noun phrase which is the syntactic subject of a clause.

nsubjpass: passive nominal subject A passive nominal subject is a noun phrase which is the syntactic subject of a passive clause.

dobj: direct object The direct object of a verb phrase is the noun phrase which is the (accusative) object of the verb.

root: root The root grammatical relation points to the root of the sentence.

xcomp: open clausal complement An open clausal complement (xcomp) of a verb or an adjective is a predicative or clausal complement without its own subject.

advmod: adverb modifier An adverb modifier of a word is a (non-clausal) adverb or adverb-headed phrase that serves to modify the meaning of the word.

ccomp: clausal complement A clausal complement of a verb or adjective is a dependent

clause with an internal subject which functions like an object of the verb, or adjective.

advcl: adverbial clause modifier An adverbial clause modifier of a verb phrase or sentence is a clause modifying the verb (temporal clause, consequence, conditional clause, purpose clause, etc.).

dep: depend A dependency is labeled as "dep" when the system is unable to determine a more precise dependency relation between two words.

amod: adjectival modifier An adjectival modifier of an noun phrase is any adjectival phrase that serves to modify the meaning of the NP

cc: coordination A coordination is the relation between an element of a conjunct and the coordinating conjunction word of the conjunct.

4.15 Conclusion

We take a step towards the AI-complete task of Visual Question Answering. Specifically, we tackle the problem of answering binary questions about images. We balance the existing abstract binary VQA dataset by augmenting the dataset with complementary scenes, so that nearly all questions in the balanced dataset have an answer "yes" for one scene and an answer "no" for another closely related scene. For an approach to perform well on this balanced dataset, it *must* understand the image.

We propose an approach that extracts a concise summary of the question in a tuple form, identifies the region in the scene it should focus on, and verifies the existence of the visual concept described in the question tuple to answer the question. Our approach outperforms the language prior baseline and a state-of-the-art VQA approach by a large margin on the balanced dataset. We also present qualitative results showing that our approach attends to relevant parts of the scene in order to answer the question.

Chapter 5

Semi-supervised Attention for Visual Question Answering

5.1 Introduction

Acting as a multi-modal task, VQA attracts more and more attention from the community. VQA systems take information from images and questions, after further reasoning between them, answers are predicted.

Approaches have been proposed to address this task, including [4,48,75,94,127,147]. Most of them encode images with convolution neural network (CNN) [2,130], which was pretrained on ImageNet for image classification task. Questions are encoded by recurrent neural network, i.e. Long Short Term Memory network (LSTM) [62]. However, how these signals are handled affects the final performance most.

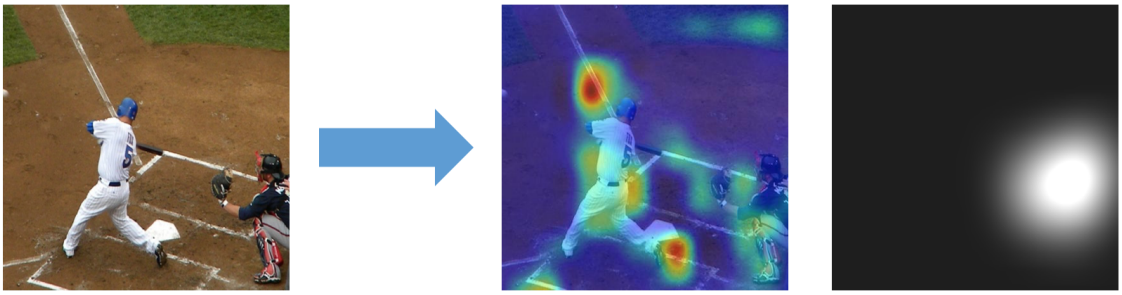
The VQA challenge leader board [4] shows a complete list of performance from all the existing approaches, ranging from 29.72% to 71.48%. After further analyzed each method, we found that all the top-performing models used attention neural network, which tried to learn the compatibility between image regions and question.

As humans, given the question and image, we first try to understand the meaning of the question, then we look for relevant image regions. For example, for the question "What is the color of the cup?", we know it is asking about the cup, then we look for "cup" region from the image. Next, we can reason about its color and give the answer. The attention-based models try to mimic such schema, preventing the system from being distracted by irrelevant objects and propose a nonsense answer.

Specifically, in attention neural network, the system tries to learn an attention mask for each question-image pair. All the entries in the mask are normalized between 0 and 1, showing the correlation between each image region and the question. The higher the value is, the more compatible they are. For example, in the above example, we want the "cup" region to have the highest values, while others have lower values. Attention mask is then further combined with the image to generate image feature.

This idea advocated a lot of attention-based approaches, which showed better performance than the ones without such mechanism. Each of them might equipped a different way to learn the attention mask, but all of them share the same property. That is, attention mask was treated as a latent variable and it was learned in an unsupervised manner. Because of this, the attention mask might not always attend to the correct regions as we want, since there is no direct supervision on it. Figure 5.1 shows an example, we feed one question "what is the guy on the right doing?" and an image into an existing attention-based VQA system, Hierarchical Co-Attention model [94]. Then we obtain the attention mask and answer. In the attention mask, dark regions show where the system is looking at. Apparently, it looks at the incorrect regions. Because of that, it outputs an incorrect answer: batting. If we were able to have the system look at "the person on right", probably it could predict the correct answer: "catching". To accomplish this, we need to have a direct supervision to the attention mask. Such label is different from the ones we give to train the traditional VQA model, i.e. answer for each question-image pair. In this case, we need to give per pixel label for the attention mask, as Figure 5.1 shows.

Figure 5.1: Machine learned attention map from [94]. The system attends an incorrect image region and leads to an incorrect answer. We would like to add human attention as a secondary supervision to guide the attention generation, so that the system can predict the correct answer, e.g. changing from "batting" to "catching".



Human attention vs. Machine generated attention Humans have extraordinary ability in reasoning between image and question. Locating the places of interest from the image plays an important role. We thus want the system to "look at" the same image regions as humans do. This can be effectively tackled by providing human attention as direct supervision. One obvious concern on this setting is, machine might not look at the same regions as humans when it outputs a correct answer. But we believe more human-like attention should be important. For example, in order to answer "how many apples in the basket" correctly, the system is required to put attention on "apple" rather than irrelevant regions. Human attention groundtruth can correct failures that are typically hidden during sequential reasoning. Therefore, we expect systems that are supervised by human-like attention should further boost performance.

End-to-end model vs. Compositional model Given the training data and labels, it is straightforward to build a supervised learning model. However, the existing attention groundtruth data was only collected on a *subset* of the whole VQA dataset. It is hard to directly train both an attention module and a VQA system in a full supervised learning model. There are two ways to solve this. One is we cast this as a two-step supervised learning task. In that case, we need to train them sequentially with two different optimization

functions. And the other is we take this as a semi-supervised learning model, so that it can be trained in an end-to-end fashion. We argue that two-step training might loss too much search space, resulting in less optimal overall performance. It could be over-kill compared with the one-step end-to-end setting. Therefore, we prefer to the second setting with a novel loss function, giving the whole model relatively large search space and incline to propose correct answers.

In this work, our contributions are: 1) We propose to use human attention ground truth to guide generated attention, which results in improved accuracy on VQA. 2) We cast our model as a new semi-supervised end-to-end framework. 3) Our method is general, which can be used in all kinds of attention-based VQA model. The rest of this work is organized as follows: we first go over existing methods in section 5.2. Then we detailed explain our method section 5.3. The datasets that we use are introduced in section 5.4 and lastly we show the results in section 5.5.

5.2 Related work

Attention neural network has been proved successful in almost all AI-related areas. Mostly because it coincides with how humans recognize the world. In VQA, questions and images are used to generate attention masks first, as in [48, 75, 94, 147, 149, 150]. Image regions that are most compatible with questions are highlighted in the masks. We follow the same idea here to generate our attention masks. Then new image features can be obtained by combing with the images.

Human attention. In all attention networks, attention mask is latent variable and is generated in an unsupervised way. So it might not be the same as what humans expected. But we still do not know how much the difference is or what the correlation is. Therefore, [24] first collected human attention on a subset of VQA data to understand such difference. For each image, humans are required to highlight regions that are necessary to answer the given

question. By comparing with the machine generated attention maps and human collected attention maps, [24] showed that machine generated attention does not align with human attention. However, in this work, we prove that by using human attention maps as a secondary supervision, the overall performance can be improved.

One recent work is [49], which has the similar idea as ours. But they collected annotations on the whole VQA dataset. For each question-image pair, they annotated boundaries of the whole relevant objects with curves. But in our case, the attention map only attends to corresponding object parts, rather than the full objects. [116] also takes in human attention map as a secondary supervision, but this is applied in object detection task.

Another series of works are [71, 77]. They were trying to predict where humans looking at on a given image. Based on the predicted attention maps, further applications could be performed. In their works, only one attention map was generated for each image since it only captured the most salient parts. However, in our case, we generate different attention maps upon different questions.

5.3 Approach

Figure 5.2 shows an overview of our model, which takes images and questions as inputs. They pass through the attention module to produce attention mask. It is then combined with the image to obtain the image feature. Finally we use multi-layered perception (MLP) module to predict an answer.

We extract image features from VGG 19-layer network, which was pretrained on ImageNet [27] dataset for classification task. All the images are rescaled and center-cropped into size 224×224 before fed into the network. The output from conv5 layer is taken as feature.

On question side, we tokenize all the questions into words and build a vocabulary. Each

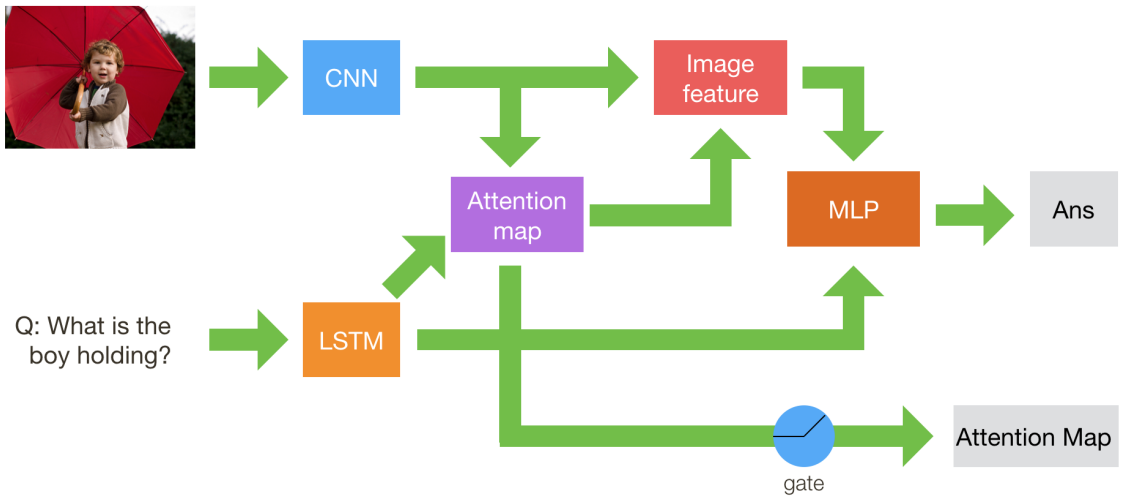


Figure 5.2: Proposed VQA model

word from every question is then one-hot encoded and right-aligned. We set the question length as the same. For questions that have less words, they are zero-padded from left. The encoded questions are fed into a lookup table to get embeddings, which are then applied to LSTM. The output from the last hidden layer is taken as the feature.

5.3.1 Attention module

We follow [94, 147] to generate attention mask. Assuming the extracted image feature and question feature are $I \in R^{M \times K}$ and $Q \in R^{1 \times N}$, which are first projected into a space with the same dimensionality.

$$I = I \cdot D_1 + b_1, Q = Q \cdot D_2 + b_2$$

$$I = Tanh(I), Q = Tanh(Q)$$

where $D_1 \in R^{K \times d}$ and $D_2 \in R^{N \times d}$ are projection matrices, b_1 and b_2 are biases. The two features are then fed into Tanh nonlinear function.

I stacks the extracted image feature at each grid. Therefore, the correlation between question

and each image grid can be obtained by dot-product. Then the correlation matrix is given as:

$$C = I \cdot Q^T$$

where $C \in R^{M \times N}$.

Thus, question feature can then be projected on the image space by following:

$$Q_C = Q \cdot C$$

which is then be added onto the image feature as:

$$Q_I = Q_C + I$$

where $Q_I \in R^{M \times d}$ is the fused feature from image and question.

The fused feature also stack of feature across all the grids, which contains image and question correlation at each position. So the attention mask can be obtained by:

$$A = Softmax(Q_I \cdot D_3 + b_3)$$

where D_3 is the projection matrix, b_3 is the bias. And softmax layer is used to normalize the values between 0 and 1.

Lastly, we can obtain the attention image feature as:

$$I_{atten} = I \cdot A$$

We do not take question attention feature, but directly feed the original question feature into the downstream module.

5.3.2 Losses

Following the existing work, we cast this as a classification task to predict an answer. From training data, the top most common N answers can be found, which are then be set as

candidate answers. Then for each question-image pair, this is an N-class classification task. The most probable class is the answer. As a classification task, cross-entropy loss is used to guide the training process. Meanwhile, during backpropagation, compatible images regions with question are fired and thus attention mask can be learned.

In addition to that, we also introduce ground truth attention loss to the system, which comes from the comparison between generated attention mask and human attention. We define attention loss on each question-image pair to be the mean-squared error between human attention and learned attention:

$$AttentionLoss_i = MSE(gt_i, mask_i)$$

where gt_i denotes the human attention for the i^{th} training sample, and $mask_i$ is the learned attention.

However, as mentioned above, such ground truth is only available to a subset of training examples. Thus, in cases where there is no such supervision, the attention loss is 0.

For each training sample $x_i, i \in \{1, 2, \ldots, N\}$, we associate it with an indicator $y_i, i \in \{1, 2, \ldots, N\}, y_i \in \{0, 1\}$, where 0 means the attention ground truth is not available. The attention loss can then be computed as:

$$Loss_{attention_i} = y_i AttentionLoss_i$$

where $AttentionLoss_i$ is the original attention loss. For samples that do not have human attention ground truth, a mask with all zero is set as the attention ground truth.

Therefore, to combine both of them and train an end-to-end model, the overall loss in our system is:

$$Loss = Loss_{cross-entropy} + \lambda Loss_{attention}$$

where λ is a hyper-parameter, which is used to emphasis the importance of attention loss.

5.4 Dataset

VQA 1.0 dataset contains 82,783 training images, 40,504 validation images and 81,434 test images. On each image, 3 questions are associated. This results in 248,349 / 121,512 / 244,302 questions in each split, respectively. All the images were taken under natural scene conditions. And the questions range across all the possible types.

As a secondary supervision, we use the VQA human attention (VQA-HAT) dataset, which was collected by Amazon Mechanical Turks. Each Turk was given a blurring image and a question about it. They were asked to sharp the image regions, so that it will help them answer the question correctly. They were doing in a smooth, click-and-drag, coloring motion with the mouse. The sharpening is gradual: successively scrubbing the same region progressively sharpens it [24]. This process was only done on a subset of VQA question-image pairs. Each question-image pair from training set was "labeled" by one human subject, and on validation set, each was "labeled" by three subjects. This leads to 58,475 attention maps from training set and 4,122 attention maps from validation set.

5.5 Results

LSTM and VGG networks are used to extract questions and image features, respectively. All the features were embedded into 512 dimensions before fed into MLP. We used Adam to perform optimization, with learning rate as 5e-5, batch size as 128. The maximum number of epoch was set as 250. We performed early stop if the validation loss / accuracy had been kept the same for 3 epochs.

We focused on OpenEnded questions from VQA 1.0 dataset. The top 1000 common answers from training set were used as candidates. So given each question at test time, we predicted which one of the 1000 classes fits most.

We first demonstrated the importance of adding attention module to VQA system. Based

on the proposed model, we obtained two results. One came from the model without human attention supervision, and the other came from the model with such supervision. We show the comparison between them per answer types in Table 5.1. Attention module was able to improve the overall accuracy to 55.20%. Moreover, by adding human attention, we were able to get the results improved from 55.20% to 56.01%. Systems with human attention did better when the answers were either "yes/no" nor "number" (others). By further combining these two models, we got an even improved performance.

	Yes / No	Num	Other	All
No attention	79.80	32.90	40.70	54.30
Without human attention	80.25	34.28	41.71	55.20
With human attention	80.38	34.10	43.27	56.01
Combined model	**80.65**	**34.36**	**43.98**	**56.50**

Table 5.1: Evaluation on VQA validation set for each answer type. All accuracies are calculated using the VQA [4] evaluation metric.

As we discussed before, we put the parameter λ to emphasis the importance of the two loss. We trained 5 models on the VQA 1.0 training set, with λ as 10, 20, 30, 40 and 50, to have a better sense about how this parameter affected the overall performance. These models were then tested on the validation set. We were able to get the comparison, as shown in Figure 5.3. We found that by emphasizing more on the attention loss, better accuracy could be expected.

5.6 Conclusion

Attention neural network has been applied in all AI-related areas and achieved good performance. Acting as the intermediate output, attention map reveals the inner workings of the system. But it is treated as a latent variable and learned in an unsupervised way in most

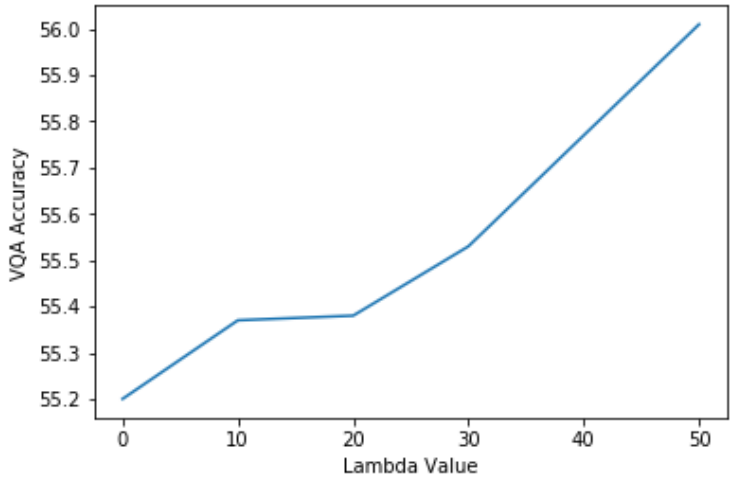


Figure 5.3: The accuracy of our proposed model as parameter λ changes.

approaches. Therefore, the generated attention mask might not what humans expected. And the difference between the learned attention and human attention has been mostly ignored. In this work, we took human perceived attention to guide the generation of attention mask. By checking the correctness of such output, we can ensure our system works as we expect. The attention groundtruth only accounts for a subset of VQA dataset. We cast this as a semi-supervised learning with a novel loss, so it can be trained in end-to-end fashion. On the same benchmark dataset, we improved the overall performance with all the above settings.

Chapter 6

Find Better Analogies

6.1 Introduction

Distributional semantic vector spaces (distributional vectors), such as `word2vec` and Glove, capture a great deal of information about the relationships between words. They encode words [1] as vectors in such a way that words occurring in a similar context will be represented as similar vectors. Such an encoding implicitly captures a great deal about the meaning of such words. [102] points out that "the sum of two word vectors is related to the product of the two context distributions. The product works here as the AND function: words that are assigned high probabilities by both word vectors will have high probability, and the other words will have low probability." This means that the sum of two word vectors results in a vector which shares the contexts (and therefore the meaning) of both input words. The principle also works in the other direction: the vector corresponding to a given word will be approximately equal to the sum of the vectors of words which define it. Because the vectors are sufficiently high dimensional, sums rarely cause unwanted collisions. The vector from France to Paris ($Paris - France$) can be considered as encoding the relation between the two words, of capturing the meaning "is the capital city of." When added to the name of

[1] or short phrases, but for simplicity both will be referred to throughout this paper as words

another country, the resultant vector was observed to be close to the vector for the capital city of that country.

This semantic arithmetic, $(b + c - a)$, has proved surprisingly powerful, and there are many relations captured pretty well by simply subtracting two vectors which share such a relation. However, there are some limitations to what kind of relations work well with such a system. These limitations fall into two types. Some relations, such as $object : color$, do not often show up in text corpora. Information that everyone already knows is not worth mentioning, and so is not captured well by any technique used to generate word vectors from a text corpus. This work is not particularly concerned with failures of this type. The second type of failure is a structural one: it is impossible for the method described above to accurately encode relations whose first term doesnt map to their second term in a functional (one-to-one and onto) way. Consider the relation *hyponym*. Three examples of this relation are $utensil : spoon$, $dog : chihuahua$ and $dog : terrier$. The analogy $utensil : spoon :: dog :?$ Is asking us to find a word that has the *hyponym* relation to *dog*, as *spoon* has the *hyponym* relation to *utensil*. However, both *chihuahua* and *terrier* are equally acceptable examples of the *hyponym* relation to *dog*. There is no reason to prefer one to the other. The relation *hyponym* is not one-to-one and onto: it is not functional. Instead it is one-to-many. Other relations may be many-to-one (e.g. the hypernym relation) or many-to-many. Another example: the relation *writer* has one meaning when referring to books, and a rather different meaning when referring to movies, so the relation is probably bimodal in a semantic vector space.

These non-functional relations may still be easy to represent in the distributional vector space, but, as observed by [114] , [107] and [143], the arithmetic of $(b + c - a)$ is just too simple to capture all the types of relation. A neural network trained to recognize whether four terms form a sound analogy, however, would have the flexibility to represent all kinds of different relations, including the non-functional ones.

The purpose in improving analogy-forming is not simply to get higher SAT test scores. [132]

showed how distributional vectors can be used together with knowledge bases to answer all kinds of questions about the world in a non-brittle way. As Douglas Hofstadter is fond of pointing out, analogy is the core of cognition. To the extent that our systems can discover sound analogies, they can extend their knowledge of the world from known cases to similar but unknown cases. This is a key tool for robust and flexible intelligence.

6.2 Distributional vectors solve a hard problem

Before the ability of distributional vectors to solve analogies was demonstrated, it seemed impossible to capture the many subtle relations a concept has with other concepts in our minds. In the 1996 book *Fluid Concepts and Creative Analogies* [63] wrote "What possible set of apriori criteria would allow a computer to reply, perfectly self-confidently, that the country of Monaco is "the Atlantic City of France?" The `word2vec` vocabulary used in this work doesn't include the term $Atlantic\_City$, but for the analogy $Nevada : Las\_Vegas :: France : \_?\_$ It returned the answers

1. $Paris$
2. $Nice$
3. $EuroDisney$
4. $Monte\_Carlo$

The 15th result was $Monaco$. The first one has the relation $largest\_city$, which is a true relation between Las_Vegas and $Nevada$. $EuroDisney$ isn't even a city, but it's relationship with Las_Vegas is unmistakeable. Overall, these seem like humanlike answers. Monaco isn't the first result, but the diversity and appropriateness of the answers is surprising. Hofstadter's point that any hand-designed system would rigidly only consider the names of cities clearly does not apply here.

Another example: In the paper "Relational priming is to analogy-making as one-ball juggling is to seven-ball juggling," [46] Robert M. French gives the following analogy as an example of something that could never be handled by relational priming:

"Ty Cobb, one of baseballs greatest players ever, did not hit over-the-wall home runs like his archrival, Babe Ruth. However, in May 1925, after being egged on by reporters, he said, "You want me to hit home runs like Babe Ruth? No problem." So, in the next game he hit three over-the-wall home runs and two more in the following game, after which he said, "I told you I could hit home runs. Now Im going back to playing baseball the way it was meant to be played." It occurred to me that this is analogous to a professor who publishes only books and who is criticized for never publishing in journals. One day, in response to his critics, he says, "I could publish in journals if I wanted to; I simply chose not to," and to prove his point, he rapidly racks up a number of publications in the top journals in his field. Thereafter, he returns to writing books. Here we have two situations in which the objects have no features in common (base hits and books; home runs and journal articles) and where the relations are semantically miles apart (hitting and writing). And yet, the analogy works perfectly. This is the heart of analogy-making, and it is not in the least clear how relational priming, as implemented by the authors, could begin to deal with this problem."

And yet, `word2vec` (which performs essentially relational priming, if you consider the vector between two terms to be a kind of weighted sum of all their relations) when given the analogy $BabeRuth : homerun :: \_?\_ : scientific\_study$ correctly returns $researchers$ and $scientists$ as the top two results.

This ability, while remarkable, is still limited, however. Recent work has shown how and why certain relations are more difficult to be solved.

6.3 Recent work on capturing relations in distributional vectors

Mikolov [100] introduced `word2vec` and the Google semantic/syntactic dataset (hereafter "the Google dataset"). This consists mainly of one-to-one relations. In [102] their phrase dataset includes the relations of newspapers, NHL teams, and NBA teams from particular cities, airlines of particular countries, and company executives for particular companies. These are also all one-to-one or few-to-one relations.

Research in how distributional semantics is able to capture particular relations using alternatives to the $(b + ca)$ method has focused mainly on linguistic relations such as hyponyms, hypernyms, and synonyms. [114] attempts to find hyponyms. [107] is an attempt to find hypernyms. [143] used an entropy measure to establish that "hyponyms and hypernyms are distributionally similar, but hyponyms tend to occur in more informative contexts than hypernyms." [85] is a skeptical take on whether hypernymy is really captured in distributional vectors. [59] examines the relative frequency and distribution of hypernyms, hyponyms, meronyms, holonyms, and synonyms from Wordnet and finds they have unique distributions.

There has also been recent work on studying the extent to which various properties of concepts are captured by distributional vectors. These can also be considered as relations. [55] looks at to what extent quantitative features of countries and cities are learned by distributional methods and finds modest improvements over baselines, suggesting that some of this information is being learned. [118] finds that while taxonomic properties are captured well by distributional vectors, certain attributive properties (such as shape and color information) are not learned nearly as well.

6.4 Finding the region containing solutions to an analogy

In the attempt to improve results in solving analogies, some authors e.g. [84] have hand-designed new formulas for finding the solution to $a : b :: c : d$. While this has been somewhat successful, it can be difficult to find such formulas because of the counterintuitive behavior of mathematical operations in high-dimensional space. For example, let a and b be two words (that is, labeled points) in normalized distributional vectors of length 1 in space of dimension d. The nearest neighbors to the following operations all pick out nearly the same set of words:

- element-wise $max(a, b)$
- element-wise $min(a, b)$
- $mean(a, b)$
- $sum(a, b)$
- $n*$ any of the above where $n > .00001$ and has no upper limit

These facts follow from the distribution of values in the elements of the vectors and the properties of high-dimensional space. For each element m of a and b, $c_m = max(a_m, b_m)$ will either equal a_m or b_m, with either answer equally likely for each element. Approximately half of the elements will be identical between c and a, and half will be identical between c and b. If we consider points of distance < 1 to be near to this $d/2-$ dimensional subspace, we find that the region it picks out is $2^{d/2}/2^d$, much smaller than the full space at distance 1 (for $d = 300$ this is $7 * 10^{-46}$.) Since both a and b lie in this region, the region will consist only of those points semantically very near to a, b, or both. The case for minimum is similar. The average of a and b is the point directly between a and b on the line joining them, so it also picks out the small set of points near a, b, or both a and b. The function $(b * c)/(a + \epsilon)$

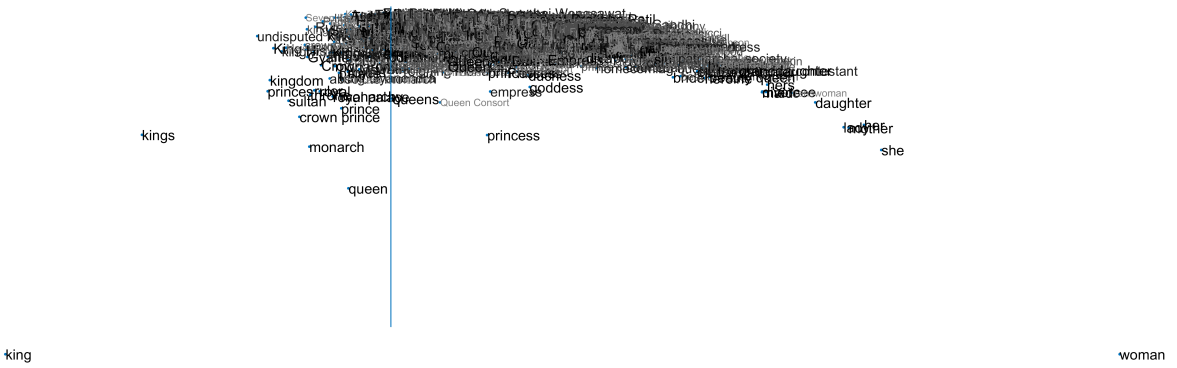


Figure 6.1: the hundred terms nearest to the word *queen*. Every vector in this image has had the vector for *man* subtracted from it. The horizontal axis is the line segment connecting the word *king* to the word *woman*. The vertical axis shows how far away each word is from this line segment (collapsing the remaining 299 dimensions). The word *queen* is the closest to the point $king - man + woman$. However, words like *kings*, *she*, and *daughter* are also ranked highly, despite not sharing in the meaning of both *king* and *woman*, simply because they happen to be nearby *king* or *woman* respectively. A better method would concentrate more on the central area of the graph.

to solve analogies in distributional vector spaces was introduced in [84]. This multiplication method also picks out a similar region.

The usual way of finding the fourth word in an analogy $a : b :: c : d$ in distributional vector space is to look for words near to $(b + c - a)$. Due to the properties explored in the previous paragraph, this will indeed pick out vectors near to both b and c, which are also distant from a. Unfortunately, this hyperspherical region surrounding $(b+c-a)$ also will naturally contain vectors b and c, and their near neighbors, because of those same properties. While these may be considered to be "semantically near" in some sense to $(b + c - a)$, correct answers must share in the meanings of **both** b and c. This is normally dealt with by explicitly excluding b and c from the results, but that fails to eliminate other terms semantically very close to

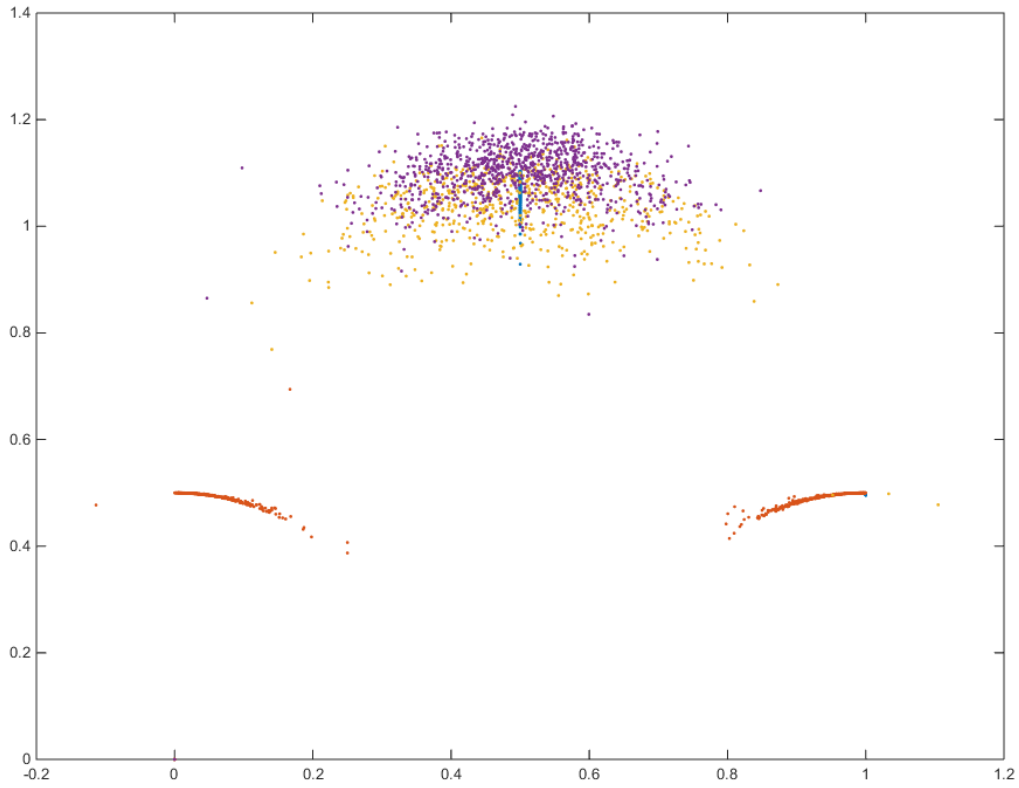


Figure 6.2: (best viewed in color) This plots the distribution of correct answers relative to other terms. The axes are the same as in figure 1, but with *king* and *woman* replaced by the second and third terms of the respective analogies in the SemEval 2012 dataset. The yellow dots are the true answers.The two orange arcs near the bottom are the top results using the Euclidean distance metric, excepting the search terms. The blue line marks the center. The purple dots in a cluster near the top are randomly chosen from among the 10000 closest terms for each analogy. The point at $(0.5, 0)$ is the solution to $(b + c - a)$. The best method would look for a solution where the probability of finding a correct answer is highest.

b without any significant similarity to c, and vice versa. What we want instead is to only consider those points semantically similar to both b and c. If we imagine a Venn diagram consisting of a circle of words near b and a circle of words near c, we only want those words which lie in the intersection. Instead of a hypersphere surrounding the point $(b + c - a)$, the region we really want is the lens-shaped intersection of the hyperspheres surrounding b and c. The ball of radius r surrounding b includes all words semantically near to b, for a definition of "near" that depends on r. To find whether a word e lies in the intersection of balls of radius r surrounding b and c, we can check whether $max(||e - b||, ||e - c||) < r$. This formula picks out words which lie in the intersection of their respective hyperspheres. Using this metric generally improves performance a few percentage points over simply choosing the word nearest $(b + c - a)$.

This raises the question, however, of whether this is actually the best function empirically. If we plot the distribution of correct answers in relation to $(b - a/2)$ and $(c - a/2)$ for many analogies, we can see that the distribution is not matched by either the usual hypersphere surrounding $(b + ca)$ or this improved region. The shape of the region may vary depending on the particular relation being considered, and there is no reason to believe it has to have a high degree of symmetry. Instead, a better approach seems to be learning what it ought to be from examples.

6.5 Non-functional relations

The above analysis assumes that there is one best answer to $a : b :: c : d$, and we simply are trying to maximize our chances of finding that answer. However, this is not the case for many relations. The $(b+c-a)$ method does not do a good job of capturing non-functional relations. For the hyponym relation in particular, words near c were found by [114] to be more likely to be hyponyms of c than words near $(b+c-a)$. The lesson is that the distributional vectors have done a good job of organizing terms semantically, but that different approaches may be

needed to learn what the best mapping is for each particular relation. Still, many relations are well served by the same method, so we want to be able to transfer what we have learned about one relation to similar relations. This is the motivation for training a neural network to learn to solve analogies using distributional vector representations as input features.

Thinking of analogies as discovering which relation exists between the first pair of terms and applying it to the second pair of terms is reasonable, but it can be overreductive and impoverishing for particular analogies. Returning to the example above, the relation country:Japan is also a hyponym relation, but most people would agree that $country : Japan :: dog : chihuahua$ is not as good an analogy as $country : Japan :: dog : Shiba$ (Shiba dogs were originally bred in Japan and are still a popular breed in Japan). The relation being captured by this analogy is actually a more subtle and nuanced one than simply a hyponym. The relation might be better expressed as $hyponym with a Japanese feel$. Distributional vectors are actually not bad at finding such subtle relations, if a way can be found to restrict the choices to only a list of hyponyms. But for the purposes of this work, we treat all examples of a relation as suitable, and simply try to discover whether two pairs of terms share the same (coarse) relation or not.

6.6 Dataset

In this work, we apply our approaches on three datasets: Never Ending Language Learner (NELL) data, the Google dataset, and the SemEval 2012 dataset. In all experiments we use 300-dimensional

`word2vec` vectors pretrained on the Google News corpus and made available for downloading. [2] The vectors have been normalized, so each has the same norm. Although there are 3,000,000 vectors in this set, we use only the 1,000,000 most common in our experiments because of RAM limitations.

[2] `https://code.google.com/archive/p/word2vec/Pre-trained_word_and_phrase_vectors`

The NELL data is the "Every belief in the KB" file from the Never Ending language Learner[3]. These (Entity, Relation, Value) triples are derived in a mostly unsupervised fashion from a collection of 500 million web pages[4]. They include a wide variety of relations, many of which are not functional. The (Entity, Value) pairs are extracted and grouped by relation.

The original `word2vec` paper by Mikolov et. al. [100] introduced the Google dataset[5]. The semantic relations included in this dataset are capital cities, currency, city-in-state, and man-woman. City-to-state is many-to-one relation, all the others are one-to-one. The syntax relations are adjective-to-adverb, opposite, comparative, superlative, present participle, nationality adjective, past tense, plural nouns, and plural verbs. Each of these relations is functional, or very nearly so.

The SemEval 2012 dataset[6] consists of 79 relations with a varying number of examples within each relation. Few if any of these relations are functional. The dataset was created to explore degrees of semantic similarity between relations. We are ignoring this purpose and simply using the dataset as another source of relations.

6.6.1 Data preparation

As discussed above, given two pairs of words as inputs, we would like to predict if they share the same semantic relation. As first step, we prepare the data for training and testing.

Every dataset consists of examples of multiple relations. Each of these relations is characterized by ordered pairs of terms. For example, one of the relations in the SemEval 2012 dataset is "Activity:Stage." Some of the ordered pairs that represent this relation are "farming:harvesting" and "cooking:peeling." We first take half of the pairs for each relation, and combine these to form four-term positive samples. In the example above, this would be

[3] `http://rtw.ml.cmu.edu/rtw/resources`

[4] `http://lemurproject.org/clueweb09/`

[5] `https://github.com/dav/word2vec/blob/master/data/questions-words.txt`

[6] `https://sites.google.com/site/semeval2012task2/download`

"farming:harvesting::cooking:peeling." This is a sound analogy (though not necessarily the absolute "best" analogy by human judgement) because the relation is the same between the first pair of terms and the second pair. Using the other half of the samples, we form unsound analogies, replacing the second pair with a random selection from other relations.

The vocabulary of the word vectors did not contain every word in the relation datasets. When this happened, the pair was discarded. The NELL data, in particular, had many pairs of this kind.

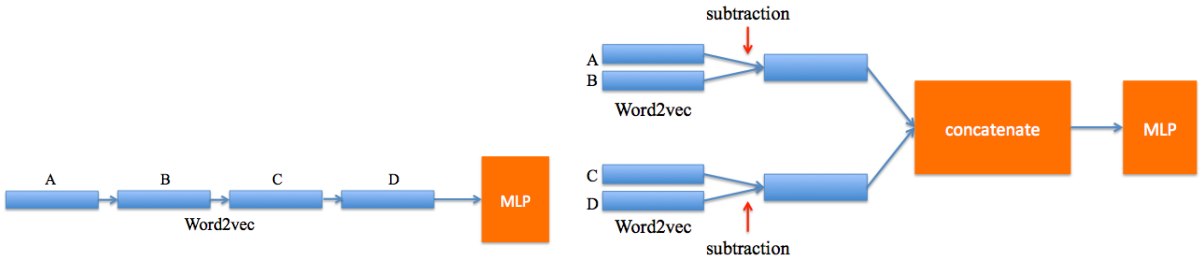
We hold out around 20% of data as the test set to report performance. To do that, we randomly select 20% of data from both positive and negative samples. The rest is used as training data to train our model.

6.7 Approach

We tested two models: concatenation and subtraction. As the figure 6.3 shows, each model takes 4 words as inputs and outputs 1 or -1, showing match or dismatch, where each input is encoded with `word2vec` as a 300-dimensional vector. We describe each of them below.

Concatenation In this model, we concatenate all the 4 inputs in `word2vec` feature to form 1200-dimensional vectors, which are then fed into multi-layered perception (MLP) layers. Each MLP layer has 300 nodes.

Subtraction Given with 2 pairs of words A, B and C, D that share the same relation, $A - B = C - D$ is usually expected to hold in `word2vec` space. Therefore, we take the difference of the `word2vec` vectors for each word pair, and the two vectors are then concatenated and fed into MLP layers.



(a) concatenation model (b) subtraction model

Figure 6.3: Models to predict relation matching

6.8 Results

We apply the above three models on all the three datasets, with our own data split. Note that there is much less training data in the emEval 2012 dataset. To train our models, we instead used Google training data. As a baseline, we verify if two pairs of words have a matched relation by evaluating the distance between them. If the distance is smaller than a threshold, then it is considered to be a matched pair, otherwise not. The threshold is obtained from the training data. On the training data, we first take the difference between each pair of word vectors, then we compute the euclidean distance between them, i.e. the distance between $A - B$ vs. $C - D$. The distance is then sorted in ascending order. We set one threshold for the entire set, and evaluate the performance by checking if the distance is smaller then the threshold or not. The threshold that achieves the highest accuracy is used to evaluate the test data.

While training our models, we hold out 20% of the training data as a validation set, where we tune the parameters of our neural networks. On the NELL and Google semantic/syntactic datasets, we set the learning rate as 0.01, and batch size as 256. All the models are trained for 50 epochs, with rmsprop as the optimizer. The results are shown in Table 6.1.

To the best of our knowledge, there has been no existing work on evaluating analogies formed from the relations in the NELL dataset. On the Google semantic/syntactic dataset,

Table 6.1: Comparisons of results between our models and baseline

	NELL	Google	SemEval 2012
word2vec baseline	80.35	76.52	50.29
concatenation	80.76	86.57	51.88
subtraction	**81.95**	**89.15**	**52.50**

existing works report performance on the whole dataset, whereas we need to use part of it for training. The SemEval 2012 data has been typically used for a somewhat different task of ranking word pairs on the quality of the analogy they form. Therefore, we performed no direct comparison between the result of our approaches and others' work.

6.9 Conclusion

We illuminated certain features of high-dimensional geometry that explain a common source of errors in finding analogies in distributional vector spaces. We also showed improvements over the baseline in the ability to recognize sound analogies on three diverse datasets. This shows that there are regularities in the distribution of analogical structures in distributional vectors that go beyond the simple arithmetic method. There were improvements for both functional and non-functional relations.

Chapter 7

Conclusion

AI systems have done incredible work and showed their great power in many applications. However, in most cases, we do not really understand how the systems work, since they usually work in a black-box fashion. As users, we can only see the inputs and outputs, without knowing the inner parts. This could be a real obstacle that prevents us from improving the AI systems.

In this dissertation, we aimed at making AI systems, especially computer vision systems, more interpretable. We followed three directions, the first one was, given the underlying system, we built a system ALERT, that was able to predict which input might be "problematic" and led to a failure of the system. Computer vision systems that try to solve the same task might be complementary to each other, so in the second direction, we would like to build a portfolio vision system, which could pick input-specific approach from candidates. Then at test time, given with each input, we were able to select the most suitable approach and made the overall best performance. Both systems are general, and can be applied to any computer vision system. Also, they have the benefit of not having to run the underlying vision systems.

The first two directions focused on the exterior of the systems, e..g when the system will

fail, when the input will fail. In the last one, we would like to focus on the inner module of the systems, trying to see if the systems *really* worked as we expect. We focused on two tasks, VQA and semantic matching. In VQA, we took the attention neural network and specifically made the system attend the correct image regions with given question. In the first stage, we worked on binary questions on abstract scenes. By summarizing each question into tuple form, we were able to specifically extract image features on regions of interest, which corresponded to the primary and second objects. Following that, we extended this to all kinds of questions and real images.

We took the successful attention neural network. Adding human attention as secondary supervision, we forced the system to learn correct attention mask, resulting in a better performance. In semantic matching, we built a network to learn the similarity between two pairs of words directly in semantic word vector space, so that the system focused on the same goal as we did.

While the community is paying more attention on build new models to achieve better performance over existing ones, we believe diagnose the failures of current models and understand their inner modules are even more important. Only know where the bottleneck is, we can avoid possible failures and efficiently make the system work better.

Bibliography

[1] Keras: Theano-based deep learning library. `https://github.com/fchollet/keras.git`, 2015.

[2] A. A Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[3] Omid Aghazadeh and Stefan Carlsson. Properties of datasets predict the performance of classiers. In *BMVC*, 2013.

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

[5] Stanislaw Antol, C. Lawrence Zitnick, and Devi Parikh. Zero-Shot Learning via Visual Abstraction. In *ECCV*, 2014.

[6] O. Mac Aodha, G. J. Brostow, and M. Pollefeys. Segmenting video into classes of algorithm suitability. In *CVPR*, 2010.

[7] Nguyen Bach, Fei Huang, and Yaser Al-Onaizan. Goodness: A method for measuring machine translation confidence. In *ACL*, 2011.

[8] Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom

Yeh. VizWiz: Nearly Real-time Answers to Visual Questions. In *User Interface Software and Technology*, 2010.

[9] Mustafa Bilgic and Lise Getoor. Reflect and correct: A misclassification prediction approach to active inference. *ACM KDD*, 2009.

[10] Arijit Biswas and Devi Parikh. Simultaneous active learning of classifiers & attributes via relative feedback. In *CVPR*, 2013.

[11] Michael Boshra and Bir Bhanu. Predicting performance of object recognition. *PAMI*, 2000.

[12] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *ICCV*, 2009.

[13] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation, release 1. http://sminchisescu.ins.uni-bonn.de/code/cpmc/.

[14] Joao Carreira and Cristian Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, 2010.

[15] ImageNet challenge leader board.

[16] PASCAL challenge leader board.

[17] D. Chen and C. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP '14)*, 2014.

[18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *arXiv:1606.00915*, 2017.

[19] Xinlei Chen and C. Lawrence Zitnick. Mind's Eye: A Recurrent Visual Representation for Image Caption Generation. In *CVPR*, 2015.

[20] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, and Shi-Min Hu. Global contrast based salient region detection. *PAMI*, 37(3):569–582, 2015.

[21] Stephen Choularton. Early stage detection of speech recognition errors, 2009.

[22] M. Cogswell and D. Batra. Semantic segmentation with deep learning. In *Scene Understanding Workshop (SUNw) on CVPR*, 2014.

[23] D. Dai, H. Riemenschneider, and L. Gool. The synthesiza- bility of texture examples. In *CVPR*, 2014.

[24] A. Das, H. Agrawal, C. L. Zitnick, D. Parikh, and Batra D. Human attention in visual question answering: Do humans and deep networks look at the same regions? In *EMNLP*, 2016.

[25] Google DeepMind.

[26] Sarah Jane Delany, Pdraig Cunningham, and Dnal Doyle. Generating estimates of classification confidence for a case-based spam filter. In *International Conference on Case-based Reasoning*, 2005.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[28] Jia Deng, Jonathan Krause, Alex Berg, and Li Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *CVPR*, 2012.

[29] U R Devarakota, Bruno Mirbach, and Bjrn Ottersten. Confidence estimation in classification decision: A method for detecting unseen patterns. In *ICAPR*, 2007.

[30] Jacob Devlin, Saurabh Gupta, Ross B. Girshick, Margaret Mitchell, and C. Lawrence Zitnick. Exploring nearest neighbor approaches for image captioning. *CoRR*, abs/1505.04467, 2015.

[31] J. Donahue and K. Grauman. Annotator rationales for visual recognition. In *ICCV*, 2011.

[32] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *CVPR*, 2015.

[33] Mark Dredze and Koby Crammer. Confidence-weighted linear classification. In *ICML*, 2008.

[34] L. Duan, C. Wu, J. Miao, L. Qing, and Y. Fu. Visual saliency detection by spatially weighted dissimilarity. In *CVPR*, 2011.

[35] Robert P. W. Duin and David M. J. Tax. Classifier Conditional Posterior Probabilities. In *Joint IAPR International Workshops on Advances in Pattern Recognition*, 1998.

[36] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[37] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[38] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[39] Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From Captions to Visual Concepts and Back. In *CVPR*, 2015.

[40] A. Farhadi, I. Endres, D. Hoiem, and D.A. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

[41] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision, CVPR*, 2004.

[42] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. In *PAMI*, 2010.

[43] Edward Filisko and Stephanie Seneff. Error detection and recovery in spoken dialogue systems. In *NAACL Workshop: Spoken Language Understanding for Conversational Systems*, 2004.

[44] J. Flavell and H. Wellman. Metamemory. *Perspectives on the Development of Memory and Cognition*, 1988.

[45] David F. Fouhey and C.L. Zitnick. Predicting object dynamics in scenes. In *CVPR*, 2014.

[46] Robert M French. Relational priming is to analogy-making as one-ball juggling is to seven-ball juggling. *Behavioral and Brain Sciences*, 31(04):386–387, 2008.

[47] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[48] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *EMNLP*, 2016.

[49] Chuang Gan, Yandong Li, Haoxiang Li, Chen Sun, and Boqing Gong. Vqs: Linking segmentations to questions and answers for supervised attention in vqa and question-focused semantic segmentation. In *ICCV*, 2017.

[50] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, and Alan Yuille. Are you talking to a machine? dataset and methods for multilingual image question answering. In *NIPS*, 2015.

[51] Tianshi Gao and Daphne Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011.

[52] Donald Geman, Stuart Geman, Neil Hallonquist, and Laurent Younes. A Visual Turing Test for Computer Vision Systems. In *PNAS*, 2014.

[53] Luis Gilberto Mateos Ortiz, Clemens Wolff, and Mirella Lapata. Learning to interpret and describe abstract scenes. In *NAACL: Human Language Technologies*, 2015.

[54] Patrick Grother and Elham Tabassi. Performance of biometric quality measures. *PAMI*, 2007.

[55] Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. Distributional vectors encode referential attributes. In *Proceedingsof the Conference on Empirical Methods in Natural Language Processing (EMNLP2015), Lisboa, Portugal*, 2015.

[56] Ralf Haeusler and Daniel Kondermann Rahul Nair. Ensemble learning for confidence measures in stereo vision. In *CVPR*, 2013.

[57] P Halcsy, A. Kornai, and C. Oravecz. Hunpos - an open source trigram tagger. In *ACL*, 2007.

[58] P.L. Hammer, P. Hansen, and B. Simeone. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Math. Programming*, 1984.

[59] Abram Handler. *An empirical study of semantic similarity in WordNet and Word2Vec.* PhD thesis, Columbia University, 2014.

[60] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[61] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[62] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[63] Douglas R Hofstadter. *Fluid concepts and creative analogies: Computer models of the fundamental mechanisms of thought*. Basic books, 1996.

[64] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *ECCV*, 2012.

[65] Andrew G. Howard. Some improvements on deep convolutional neural network based image classification. *CoRR*, abs/1312.5402, 2013.

[66] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size. In *arXiv:1602.07360*, 2016.

[67] P. Isola, J. Xiao, A. Torralba, and A. Oliva. What makes an image memorable? In *CVPR*, 2011.

[68] N. Jammalamadaka, A. Zisserman, M. Eichner, V. Ferrari, and C. V. Jawahar. Has my algorithm succeeded? an evaluator for human pose estimators. In *ECCV*, 2012.

[69] H. Jiang. Confidence measures for speech recognition: A survey. *Speech Communication*, 2005.

[70] H. Jiang, J. Wang, Z. Yuan, T. Liu, N. Zheng, and S. Li. Automatic salient object segmentation based on context and shape prior. In *BMVC*, 2011.

[71] Tilke Judd, Krista Ehinger, Frdo Durand, and Antonio Torralba. Learning to predict where humans look. In *CVPR*, 2009.

[72] A. Karpathy, A. Joulin, and Fei-Fei Li. Identifying relations for open information extraction. In *NIPS*, 2014.

[73] Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *CVPR*, 2015.

[74] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *ECCV*, 2012.

[75] Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. In *ICLR*, 2017.

[76] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *TACL*, 2015.

[77] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. Eye tracking for everyone. In *CVPR*, 2016.

[78] Matjaz Kukar. Estimating confidence values of individual predictions by their typicalness and reliability. In *ECAI*, 2004.

[79] N. Kumar, A. Berg, P. Belhumeur, and S.K. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, 2009.

[80] C.H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

[81] Nathan J. Leap. *A confidence paradigm for classification systems.* Air Force Institute of technology, Biblioscholar publisher, 2012.

[82] David Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, 2009.

[83] Yong Jae Lee and Kristen Grauman. Object-graphs for context-aware category discovery. In *CVPR*, 2010.

[84] Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180, 2014.

[85] Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. Do supervised distributional methods really learn lexical inference relations. *Proceedings of NAACL, Denver, CO*, 2015.

[86] F. Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categorie. In *Generative-Model Based Vision workshop on CVPR*, 2004.

[87] Lihong Li, Michael L. Littman, and Thomas J. Walsh. Knows what it knows: a framework for self-aware learning. In *ICML*, 2008.

[88] Z. Li, E. Gavves, K. Sande, C. Snoek, and A. Smeulders. Codemaps segment, classify and search objects locally. In *ICCV*, 2013.

[89] Zicheng Liao, Ali Farhadi, Yang Wang, Ian Endres, and David A. Forsyth. Building a dictionary of image fragments. In *CVPR*, 2012.

[90] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

[91] X. Lin, M. Cogswell, D. Parikh, and D. Batra. Propose and re-rank semantic segmentation via deep image classification. In *Big Vision Workshop on CVPR*, 2014.

[92] Xiao Lin and Devi Parikh. Don't just listen, use your imagination: Leveraging visual common sense for non-visual tasks. In *CVPR*, 2015.

[93] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[94] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, 2017.

[95] Oisin Mac Aodha, Gabriel J. Brostow, and Marc Pollefeys. Segmenting video into classes of algorithm-suitability. In *CVPR*, 2010.

[96] L. Mai and F. Liu. Comparing salient object detection results without ground truth. In *ECCV*, 2014.

[97] Mateusz Malinowski and Mario Fritz. A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input. In *NIPS*, 2014.

[98] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *ICCV*, 2015.

[99] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. Explain Images with Multimodal Recurrent Neural Networks. *CoRR*, abs/1410.1090, 2014.

[100] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[101] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, 2013.

[102] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[103] George A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38:39–41, 1995.

[104] Michael Muhlbaier, Apostolos Topalis, and Robi Polikar. Ensemble confidence estimates posterior probability. In *International Conference on Multiple Classifier Systems*, 2005.

[105] Daniel Munoz, J. Andrew (Drew) Bagnell, and Martial Hebert. Stacked hierarchical labeling. In *ECCV*, 2010.

[106] L. Navarro-Serment, A. Suppe, D. Munoz, D. Bagnell, and M. Hebert. An architecture for online semantic labeling on ugvs. In *Proc. SPIE Unmanned Systems Technology XV*, 2013.

[107] Neha Nayak. Learning hypernymy over word embeddings. 2015.

[108] A. Parkash and D. Parikh. Attributes for classifier feedback. In *ECCV*, 2012.

[109] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *CVPR*, 2012.

[110] Chao Peng, Xiangyu Zhang, Gang Yu, and Jian Sun Guiming Luo. Large kernel matters – improve semantic segmentation by global convolutional network. In *arXiv:1703.02719*, 2017.

[111] P. Jonathon Phillips and J. Ross Beveridge. An introduction to biometric-completeness: the equivalence of matching and quality. In *IEEE international conference on Biometrics: Theory, applications and systems*, 2009.

[112] E. Rahtu, J. Kannala, M. Salo, and J. Heikkila. Segmenting salient objects from images and videos. In *ECCV*, 2010.

[113] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[114] Marek Rei and Ted Briscoe. Looking for hyponyms in vector space. In *CoNLL*, pages 68–77. Citeseer, 2014.

[115] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In *NIPS*, 2015.

[116] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. Grounding of textual phrases in images by reconstruction. In *ECCV*, 2016.

[117] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary mrfs via extended roof duality. Technical report, CVPR, 2007.

[118] Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. How well do distributional models capture different types of semantic knowledge? In *Proceedings of ACL*, volume 2, pages 726–730, 2015.

[119] Fereshteh Sadeghi, Santosh K. Kumar Divvala, and Ali Farhadi. Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In *CVPR*, 2015.

[120] Arup Sarma and David D. Palmer. Context-based speech recognition error detection and correction. In *NAACL (Short papers)*, 2004.

[121] Scott Satkin, Jason Lin, and Martial Hebert. Data-driven scene understanding from 3D models. In *BMVC*, 2012.

[122] W. J. Scheirer, A. Rocha, R. J. Micheals, and T. E. Boult. Meta-recognition: The theory and practice of recognition score analysis. In *PAMI*, 2011.

[123] Walter Scheirer, Neeraj Kumar, Peter Belhumeur, and Terrance Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *CVPR*, 2012.

[124] Walter Scheirer, Anderson Rocha, Ross Michaels, and Terrance Boult. Robust fusion: Extreme value theory for recognition score normalization. In *ECCV*, 2012.

[125] Walter J. Scheirer, Anderson Rocha, and Terrance E. Boult. Learning for meta-recognition. *IEEE T.IFS*, 2012.

[126] Walter J. Scheirer, Anderson Rocha, Ross J. Micheals, and Terrance E. Boult. Meta-recognition: The theory and practice of recognition score analysis. *PAMI*, 2011.

[127] K. Shih, S. Singh, and D. Hoiem. Where to look: Focus regions for visual question answering. In *CVPR*, 2016.

[128] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

[129] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. Institute of Electrical and Electronics Engineers, Inc., August 2003.

[130] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[131] David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*. MIT Press, 2011.

[132] Douglas Summers-Stay, Clare Voss, and Taylor Cassidy. Using a distributional semantic vector space with a knowledge base for reasoning in uncertain conditions. *Biologically Inspired Cognitive Architectures*, 16:34–44, 2016.

[133] E. Tabassi, C. Wilson, and C. Watson. Fingerprint image quality. *NIST Internal Report 7151*, 2004.

[134] Torch. https://github.com/torch/torch7.git. 2015.

[135] A. Torralba and A. Efros. Unbiased look at the bias. In *CVPR*, 2011.

[136] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR'*, 2011.

[137] Kewei Tu, Meng Meng, Mun Wai Lee, Tae Eun Choe, and Song Chun Zhu. Joint Video and Text Parsing for Understanding Events and Answering Queries. *IEEE MultiMedia*, 2014.

[138] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[139] Ramakrishna Vedantam, Xiao Lin, Tanmay Batra, C. Lawrence Zitnick, and Devi Parikh. Learning common sense through visual abstraction. In *ICCV*, 2015.

[140] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. In *CVPR*, 2015.

[141] Peng Wang, Qiang Ji, and James L. Wayman. Modeling and predicting face recognition system performance based on analysis of similarity scores. *PAMI*, 2007.

[142] Rong Wang and Bir Bhanu. Learning models for predicting recognition performance. In *ICCV*, 2005.

[143] Julie Weeds, Daoud Clarke, Jeremy Reffin, David J Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *COLING*, pages 2249–2259, 2014.

[144] Peter Welinder, Max Welling, and Pietro Perona. A lazy man's approach to benchmarking: Semisupervised classifier evaluation and recalibration. In *CVPR*, 2013.

[145] John J. Westerkamp, David C. Gross, and Adrian Palomino. Automatic target recognition of time critical moving targets using 1d high range resolution (hrr) radar. *IEEE Aerospace and Electronic Systems Magazine*, 2000.

[146] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*. IEEE, 2010.

[147] H. Xu and K. Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *arXiv:1511.05234*, 2016.

[148] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[149] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. Stacked attention networks for image question answering. *CoRR*, abs/1511.02274, 2015.

[150] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *ICCV*, 2017.

[151] O. Zaidan, J. Eisner, and C. Piatko. Using annotator rationales to improve machine learning for text categorization. In *NAACL - HLT*, 2007.

[152] Hao Zhang, Danfeng (Daphne) Yao, and Naren Ramakrishnan. Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 39–50. ACM, 2014.

[153] Hao Zhang, Danfeng (Daphne) Yao, and Naren Ramakrishnan. Causality-based sense-making of network traffic for Android application security. In *Proceedings of the 9th ACM Workshop on Artificial Intelligence and Security (AISec'16)*, pages 47–58, 2016.

[154] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh. Predicting failures of vision systems. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[155] Weiyu Zhang, Stella X. Yu, and Shang-Hua Teng. Power svm: Generalization with exemplar classification uncertainty. In *CVPR*, 2012.

[156] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *arXiv:1612.01105*, 2016.

[157] C. Lawrence Zitnick and Devi Parikh. Bringing Semantics Into Focus Using Visual Abstraction. In *CVPR*, 2013.

[158] C. Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. Learning the Visual Interpretation of Sentences. In *ICCV*, 2013.

[159] C. Lawrence Zitnick, Ramakrishna Vedantam, and Devi Parikh. Adopting Abstract Images for Semantic Scene Understanding. *PAMI*, 2015.