On Efficient Bayesian Scene Interpretation

by

Ehsan Jahangiri

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

February, 2016

Abstract

Scene understanding, including object recognition, is perhaps the most challenging task in computer vision. Deep convolutional neural networks (CNNs) have received a flurry of interest in the past few years due to their superior performance. However, deep networks are computationally expensive and without efficient implementation on high performance computing systems not as practical as older methods. Furthermore, CNNs do not benefit from the human's visual selective attention and top-down contextual feedback connections. The human visual system makes extensive use of contextual information to facilitate and refine object detections; object detection and recognition based only on intrinsic features of target objects is not usually sufficient for reliable inference. In this thesis, we use a model-based approach to incorporate top-down contextual information, and analyze scenes in a coarse-to-fine fashion inspired by the visual selective attention property.

In addition to disambiguating object detection, the space of objects and their poses can be searched more efficiently by taking advantage of the contextual relations between different scene entities. We present a new approach to efficiently search the space of objects and their poses using a Bayesian method called "Entropy Pursuit", where contextual relations between object instances and other scene entities are incorporated via a prior model. Using the entropy pursuit approach we collect bits of information about the scene sequentially by greedily selecting patches whose analysis provide the most informative in an information-theoretic sense. As proof of concept we use the entropy pursuit method for multi-category object recognition in table-setting scenes. We have investigated the possibility of generating a scene interpretation by processing only a fraction of patches from an input image. Our results confirm the hypothesis that we can identify an accurate interpretation by processing only a fraction of patches if the right patches are selected in the right order. We can save computation time by processing only a fraction of patches.

Primary Reader: Donald Geman

Secondary Reader: Laurent Younes

Dedication

To my lovely mom and dad: Masoumeh and Eisa, for their unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scene understanding, including object recognition, is perhaps the grandest challenge in computer vision. Many of the current works on scene interpretation attempt to recognize objects from images of scenes by feeding local image features to classifiers trained for different object categories. Richer systems use intermediate representations based on part-based models to capture contextual relationships among those parts, *e.g.,* see [22], while other systems try to capture contextual relationships among objects, *i.e.,* the spatial layout of the scene. Object detection and recognition in scenes based only on intrinsic features of target objects is not usually sufficient for reliable inference. Contextual relations can disambiguate among competing detections. In some cases such as poor viewing quality where intrinsic features are degraded, contextual information can actually carry more information compared to local image features. The human visual system (HVS) makes extensive use of contextual information to facilitate and refine object detections which might, in fact, be one of the reasons that humans can easily detect and recognize objects and actions even under adverse conditions [65].

In the past few decades, several attempts have been made to exploit contextual information to improve performance of object recognition systems. In early vision systems, semantic context was expressed as pre-defined rules to facilitate object recognition task. For example, the popular VISIONS system by Hanson and Riseman [34] used hand-coded rules to incorporate semantic context, and in [23] Fischler used spatial context stored in the form of rules and graph-like structures to define spatial interactions between segments. Use of contextual information in these early works was based on common expert knowledge via pre-defined rules which per se constrains the system to a limited domain. To overcome this problem, recent work in object recognition uses a statistical approach that can generalize and exploit semantic context [12, 20, 25, 46, 47, 64, 71, 72, 93, 98]. Most of the recent work incorporates contextual information via pairwise terms between labeled regions of pixels obtained from segmentation or image patches using a Conditional Random Field (CRF) or Markov Random Field (MRF) model [20, 64, 72, 93]. While most of the work aimed at discriminative learning and reasoning in 2D [12, 20, 22, 39, 71, 72, 93], several attempts have been made recently at designing models that reason about surfaces of 3D scenes and the interaction between objects and their supporting surfaces [4, 40, 54, 58, 84, 90]. It has been shown that reasoning about the underlying 3D layout of the scene provides useful information to not only list the objects in the scene but to infer their interactions with other objects and surfaces [4, 41]. A 3D model can also help to detect partially occluded objects as it encourages occurrence of objects at their rightful place in the world independent of camera's vantage point. However, most of the current 3D models do not encode contextual relations among objects on supporting surfaces beyond their coplanarity.

There are two different theories on the human visual perception of scenes and objects: the first theory advocates for a view-specific representation that represents the shape of an object or scene by a single or multiple canonical view-specific 2D templates, whereas the second theory advocates for a 3D viewpoint-independent representation and processing of scenes and objects [41]. There is evidence to support both. For example, the response latency of humans in detecting objects at previously unseen viewpoints is usually considered supportive of the view-specific 2D templates theory [13, 32, 97]. However, this requires storing a extremely large number of 2D templates (representing inter and intra-class variability objects) to effectively capture viewpoint. Hence, while there are some justifications for the view-specific 2D representation theory it cannot describe the whole story due to some of its shortcomings. We seem to have an abstract representation of the 3D geometry of objects and scenes that we use in addition to their visual appearance to recognize them. Therefore, a combination of 3D representation of objects and scenes together with their 2D visual appearance seems to be a reasonable representation. In this dissertation, we use a 3D prior model coupled with a data model to incorporate contextual relations. The 2D projections of 3D scene samples drawn from a generative 3D model can be obtained using perspective projection given the camera's intrinsic and extrinsic parameters.

In addition to disambiguating object detection, contextual information can be used to search the space of objects and their poses more efficiently due to geometric and semantic contextual constraints. For example, assume that instances of two object categories, c_1 and c_2, almost always co-occur in close proximity of each other; in this case, it may not be very informative to apply a c_1-detector at a location where we are almost sure that an instance from the c_1 category exists due to confident detection of an instance from the c_2 category. In this dissertation, we study how to efficiently search the space of objects and their poses using a Bayesian approach where contextual relations between object instances and other scene entities are incorporated via a prior model. The goal is to show that we can make confident detections by processing only a small fraction of patches from a given test image if we appropriately define and investigate the "most informative" patches. We collect bits of information about the scene sequentially using our efficient search strategy, called "Entropy Pursuit", that greedily selects patches whose analysis will provide the most amount of informative in an information-theoretic sense [27]. In addition to the time saved by processing only a small fraction of patches, it is interesting to study the sequential analysis of scenes and the evolution of detection accuracy as more evidence is collected. As a proof of concept we will evaluate this scene parsing approach for table-settings. Again, the goal is to show that we do not need to process all patches to come up with the full scene interpretation and by processing only a small fraction of patches, which are the most informative ones, we can achieve decent performance. Our results have confirmed this hypothesis.

1.1 Thesis Outline and Main Contributions

In chapter 2, we introduce the entropy pursuit (EP) search strategy for scene interpretation. Entropy pursuit relies on a suitable prior model that incorporates contextual relations between various scene entities. We design two prior models both based on the high level representations of objects and their relations that acknowledge the underlying 3D physical scene for an image. In chapter 3, we present the first model called a “generative attributed graph” (GAG). We propose an algorithm to learn this model from annotated images. We built an image dataset of about 3000 fully annotated images collected from internet and annotated on the “LabelMe” online annotating website [82].

The second model (described in chapter 4), which is a Markov Random Field (MRF), is learned from the statistics calculated from GAG model samples. We propose a stochastic optimization algorithm with an accelerating step size and use it to learn the MRF model. We also provide a convergence proof for the proposed optimization algorithm. The MRF model is coupled with a data model (see chapter 5) to “regularize” the output of image descriptors in a conventional Bayesian framework. At the semantic level, the descriptors are discriminatively trained classifiers for detecting and localizing object instances. The MRF model integrates the output of classifiers with expected combinatorial and geometric patterns among objects.

The reason for designing two models was due to complementary advantages; each offered something not offered by the other model. We could suitably learn the GAG model from the limited number of annotated table-setting scenes due to the GAG model’s

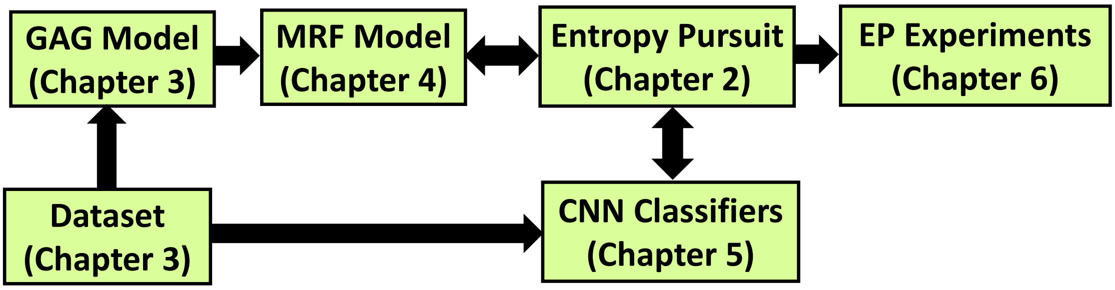


Figure 1.1: A block diagram showing the relation of different parts of the entropy pursuit project and the chapters discussing them.

Bayesian modularity; this was not the case for the MRF model. On the other hand, conditional inference can be performed much faster on the MRF model than the GAG model. Fast inference is important as it has to be performed online in each step of the EP algorithm.

The generative attributed graph model is one of the main contributions of this thesis. Note that a core challenge in computer vision is to develop generative models of the world that capture rich contextual relationships among scene entities. Generative models have recently received quite some attention for concept learning and scene perception *e.g.*, see [50, 51]. Such models are broadly applicable in scene understanding, including serving as prior models in a Bayesian framework as well as constructing model-based visual Turing tests (see [26]). The “Turing Test” evaluates a machine’s ability to exhibit intelligent behavior indistinguishable from a human. As the performance of computer vision systems improve, the need for suitable evaluation of these systems becomes more evident. According to Tomaso Poggio (2012): “We may need to understand visual cortex and perhaps the brain to achieve scene understanding at human level, and thereby develop systems that pass a full Turing test” [68]. A generative model can be used to generate sequences of unpredictable queries for testing computer vision systems; such a visual Turing test is described in [26], where the likelihoods of answers were estimated empirically from labeled data. Having an appropriate model would allow one to identify longer streams and more accurate estimates of unpredictability by sampling from the scene model conditional on oracle answers (in effect perfect classifiers). The proposed GAG model is a generative model for the viewpoint-independent 3D arrangement of multiple objects lying on a supporting plane. Many man-made scenes are composed of multiple parallel supporting surfaces upon which instances from different object categories are placed [4], often with considerable structure. The proposed model can be easily generalized to multiple supporting surfaces. The GAG model is a distribution over random attributed graphs that encodes favored layouts while accounting for variations in the number and relative poses of objects. Each graph node corresponds to an object instance that is labeled with a category and a 3D pose in the world coordinate system; the edges between nodes reflect the generative process. The proposed 3D prior model is consistent with Biederman's [8] description of well-organized scenes characterized by five relational constraints including support, interposition, probability of appearance, position, and size. Such generative 3D models can serve applications other than scene understanding, including: (1) robotics where building a physically consistent 3D map of the environment can be critical for effective interaction and prediction in physical space such as simultaneous localization and mapping, path planning, grasping and manipulating objects (see [33, 83]), and (2) computer graphics where a generative 3D model can be used for synthetic content creation; synthetic data can be used to train or evaluate computer vision systems specially when access to annotated real data is limited *e.g.,* see [44] where synthetic data from a text generation engine is used to train a deep neural network for natural scene text recognition. We illustrate how to learn the GAG model parameters from annotated images for table settings scenes which contain plates, bottles, glasses, utensils *etc*., placed on a table. We also proposed an algorithm to conditionally sample the GAG model.

We used state-of-the-art convolutional neural networks (CNN) to collect bits of information about scenes during our entropy pursuit approach. The design, training, and evaluation of CNN classifiers are discussed in chapter 5. The input image patches to the CNN classifiers have constant dimensionality resulting in the same computational cost to process each patch. This makes CNN classifiers theoretically ideal for the entropy pursuit framework that relies on unit cost tests.

The CNNs were inspired by the mechanism of the visual system. Our understanding of the visual system seems to be roughly consistent with convolutional networks [87], without benefiting from the human's visual selective attention and top-down contextual feedback connections [5]. We analyze scenes in a coarse-to-fine fashion inspired by the visual selective attention property, and use a model-based approach to incorporate top-down contextual information.

We learned a data model based on the Dirichlet distribution to integrate the collected evidence from the CNN classifiers into our Bayesian EP framework by updating the likelihoods. The data model is also described in chapter 5.

Finally, we ran experiments to evaluate performance and reported the results in chapter 6. Our results confirm that processing a small subset of patches from a given test image, if selected appropriately, can be sufficient for reliable detection.

Figure 1.1 shows a block diagram of how different parts of the EP project and the chapters discussing them are related. The main contributions of this thesis are summarized below.

1.1.1 Summary of Main Contributions

We propose a fundamentally new approach to multi-category object recognition called "Entropy Pursuit". This is a sequential search strategy, inspired by selective attention and divide-and-conquer algorithm that attempts to process the minimum number of patches from an input image necessary to acquire an accurate interpretation. This includes:

- A novel generative model, called "generative attributed graph" (GAG) model, on the 3D arrangement of multiple objects lying on a supporting plane basically a distribution on attributed graphs with random structure that encode favored layouts while accounting for variations in the number and relative poses of objects.
- A fully annotated dataset of about 3000 table-setting scenes called "JHU Table-Setting Dataset".
- An algorithm to learn the GAG model from annotated scenes.
- A method based on the Metropolis-Hastings algorithm for conditional sampling of the GAG model.
- A new stochastic optimization algorithm, with a proof of convergence, that we used to learn an MRF model from the GAG model statistics.
- A new data model based on the Dirichlet distribution for the output of CNN classifiers conditional on ground truth.

Chapter 2

Entropy Pursuit

Computer vision field and in particular object detection has observed significant progress in the past decade due to the development of better descriptors including SIFT [59, 60] and more discriminative deep features [49]. Figure 2.1 shows two perfect "plate" detection examples based on a convolutional neural network (CNN) classifier deploying the VGG-16 network [91] for both category and scale detection. The number in the brackets indicates the detection's confidence-rank and the number in the parentheses is the estimated ratio of the detected object's scale over the bounding box size. However, object detection based on only local image features (*i.e.,* classifiers only) is not always as clean as figure 2.1. Figure 2.2 shows two "plate" detection examples where almost all of the detections are false positive. A closer look at detections in Figure 2.2 reveals the reason for the false plate positives; all detections include an elliptical shape, *e.g.,* the base or top of a glass object instance, which based on only the local image features look like plate. It is interesting to note that the glasses whose base and top are detected as plate instances were recognized correctly by the CNN-based glass detector that observes a not-too-closed-up window as shown in Figure 2.3. Object detection based on only local image features can be hard even for humans and in particular for machines, if not impossible. Usually, there exist multiple object candidates with similar texture that are distinguishable only given the semantic or geometric context information. Using contextual information can disambiguate detections and make them more consistent. Consider the second, sixth, and seventh most confident plate detections in Figure 2.4(top) where the scale ratio of second and sixth most confident detections with respect to their bounding box is 0.91 whereas the scale ratio of the seventh most confident detection is 0.89 with respect to a bounding box whose size is half the size of the second and sixth bounding boxes; this is contextually (geometric context) inconsistent since plates have a typical size in real world and it is unlikely to vary too much in size as suggested by the three detections. Also consider the seventh detection in Figure 2.4(bottom) where an piece of cloth with elliptical shape is detected as plate; this detection is contextually (spacial context) inconsistent since we expect the detected plates to be on the table. Contextual information can play an even more important role in object detection based on shallow (versus deep) image features or in the presence of degradation (*e.g.,* blur). We usually make assumptions about object identities based on its size and location with respect to other objects present in the scene. Consider Figure 2.5, where the same black blob with the same identical pixels can be interpreted as a plate, bottle, cell phone, car, pedestrian or shoe, depending on the context (picture taken from [98]).

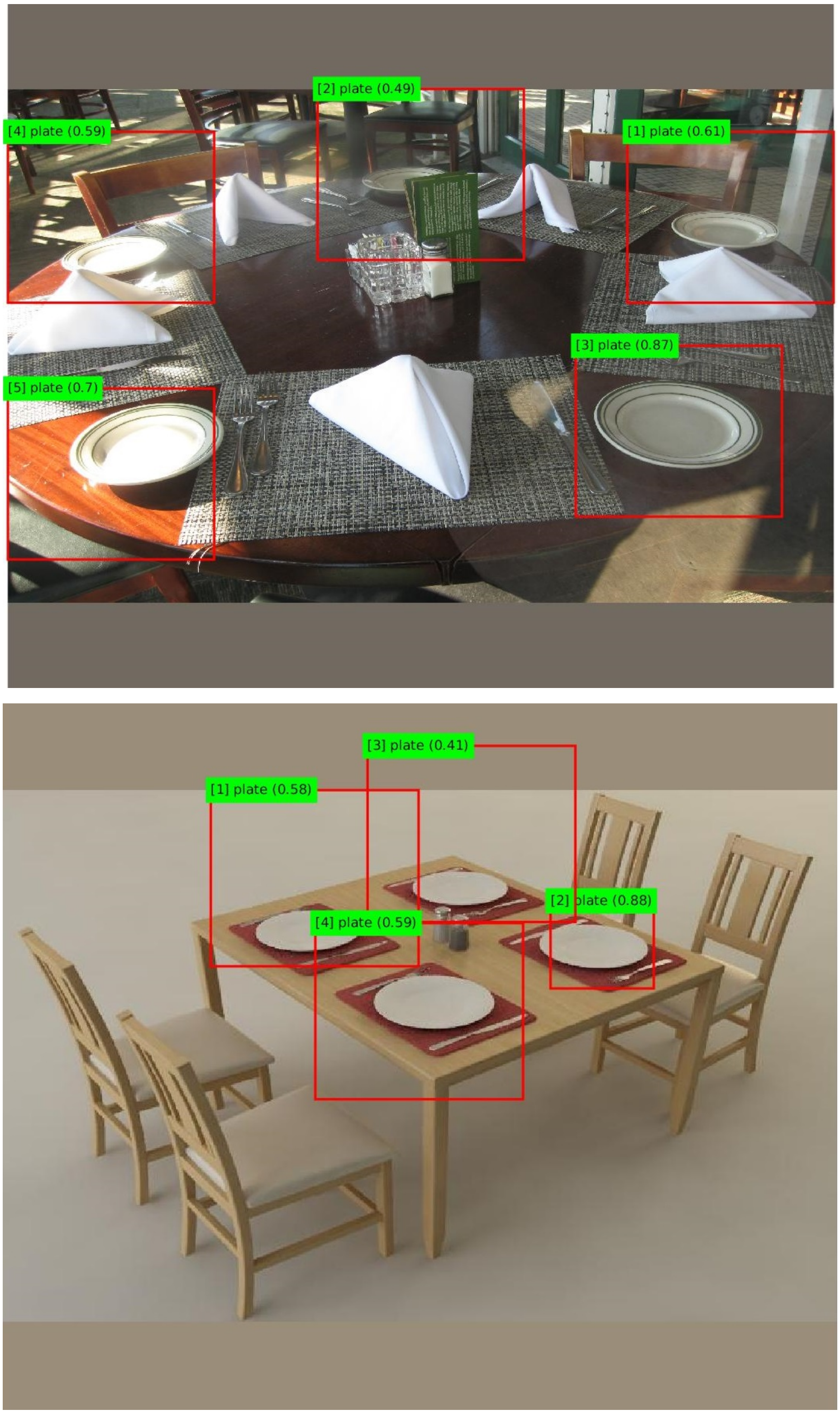


Figure 2.1: Perfect “plate” detection examples based on the CNN classifier.

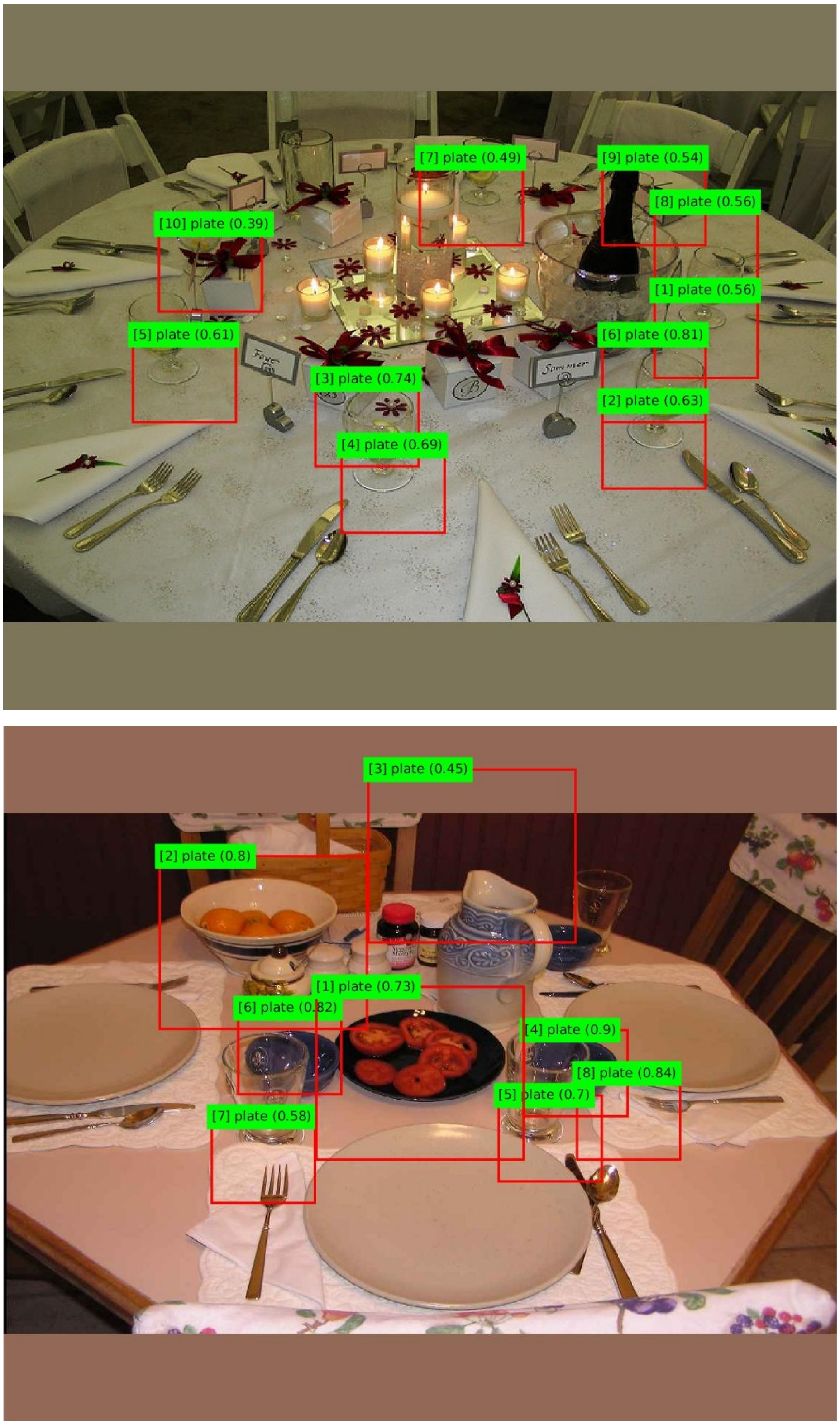


Figure 2.2: Poor "plate" detection examples based on the CNN classifier.

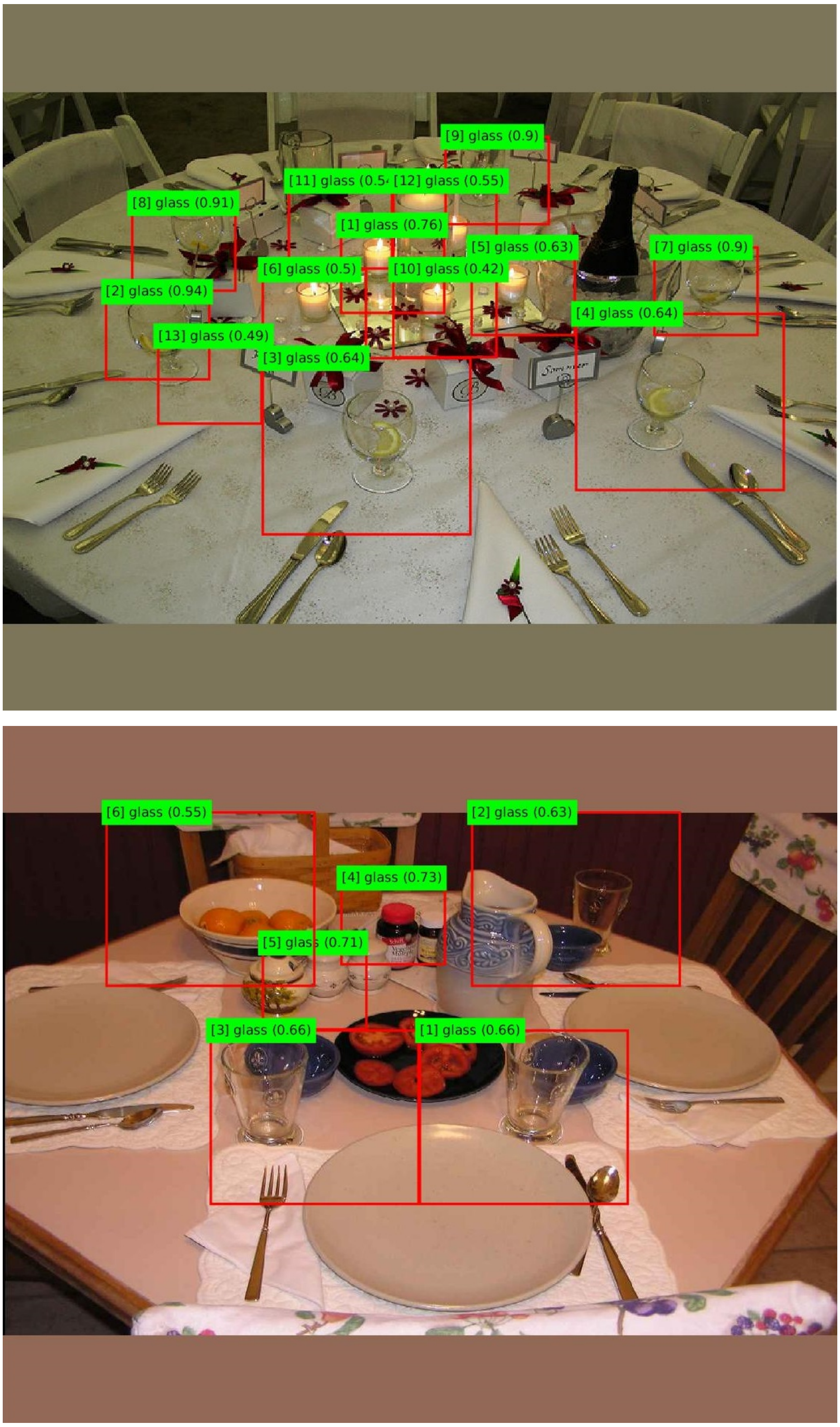


Figure 2.3: “Glass” detection examples based on the CNN classifier.

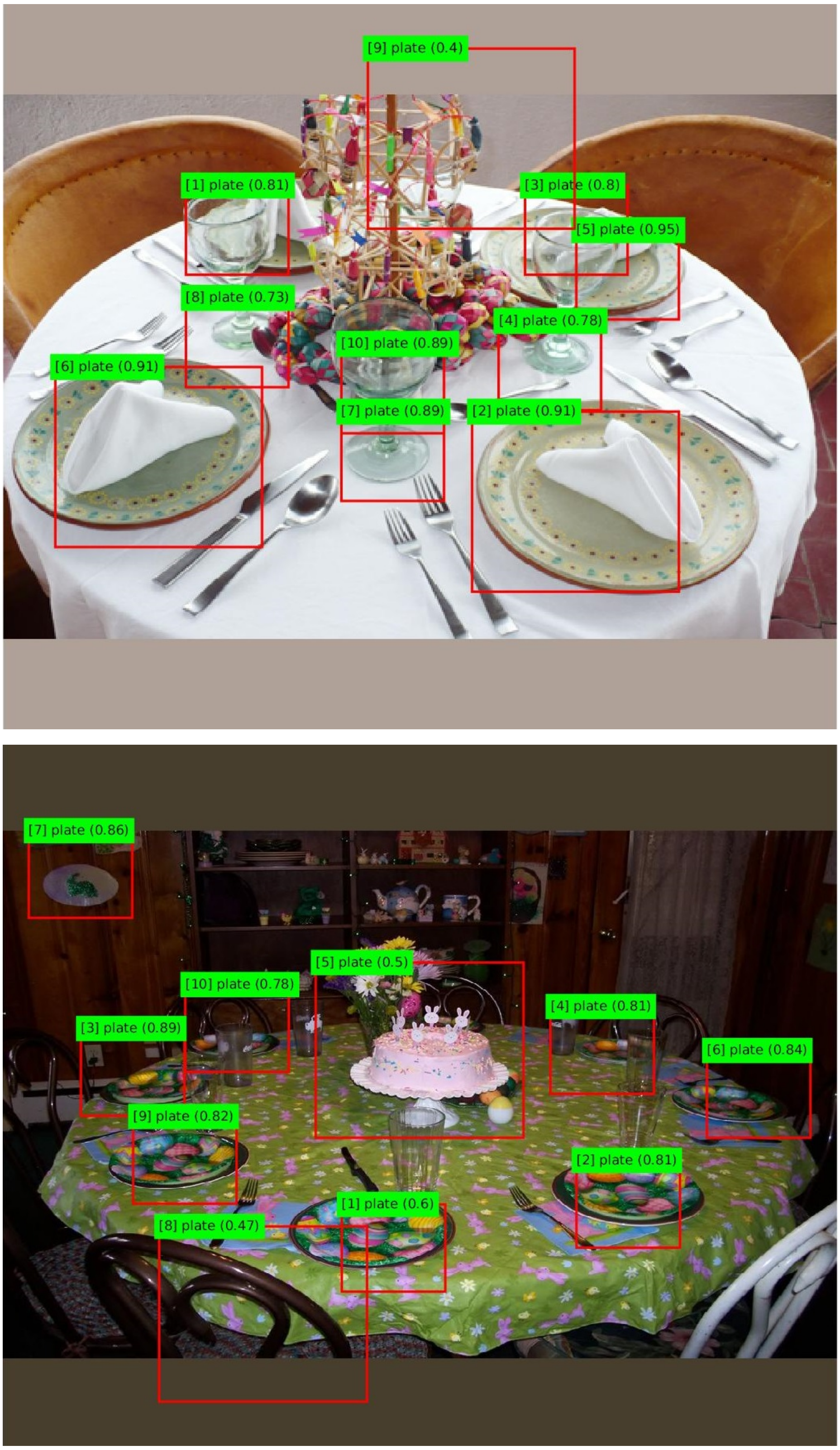


Figure 2.4: Contextually inconsistent detection example.

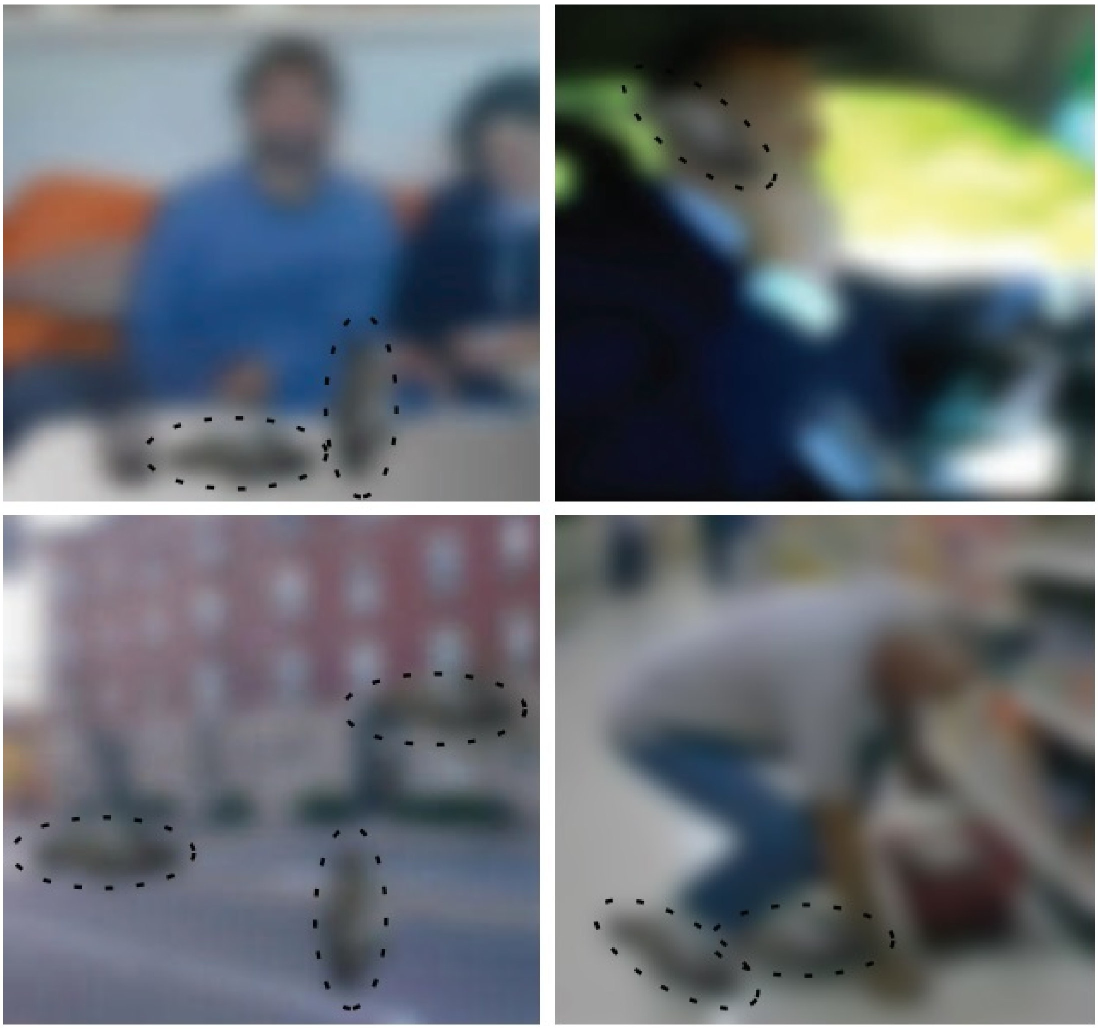


Figure 2.5: Object recognition by humans is strongly influenced by contextual information. The same black blob can be interpreted as a plate, bottle, cell phone, car, pedestrian or shoe, depending on the context (figure taken from [98]).

We propose a fundamentally new approach to automated scene interpretation that efficiently explores the contextual relations among various scene entities. This approach is inspired by attributes of natural vision which seem to be missing in most vision systems. The humans visual system selects potential targets in a scene and ignores other items through the acts of selective attention [74, 85]. We switch from a zoomed-out mode, where we monitor the scene holistically, to a zoomed-in mode for local scrutiny and back again depending on the currently collected evidence. We analyze scenes at different levels of resolution which is often coarse-to-fine. Hence, in order to simulate this property of the human visual system, we acquire evidence from multiple scales and locations at different levels of resolution and integrate them coherently by updating likelihoods. Our strategy is also motivated by the "divide-and-conquer" search strategy by humans. Using this strategy we excel in parlor and board games, such as "Twenty Questions", by making fast and accurate decisions due to inquiring the right questions in the right order. The "object detection" problem, as a part of more inclusive "scene interpretation" problem, can be formulated as a search problem in the space of instances from different object categories and their poses including location and scale. Motivated by the divide-and-conquer search strategy we analyze scenes sequentially by greedily identifying and applying the most informative classifier (in an information-theoretic sense) at each step given the accumulated history of already applied classifiers. One can think of classifiers at different locations and resolutions as inquiries in a "Twenty Questions" game.

Entropy Pursuit (EP) is what we call an adaptive stepwise strategy that we use to search the space of objects and their poses in scene interpretation. The method operates over a predefined collection of hidden partial interpretation units denoted by $\mathbf{Y} = \{Y_i\}_{i=1}^{K}$, where each interpretation unit Y_i is an unobserved random variable of one of the following two types:

■ *Annotation bit* (*Annobit* in short): an annobit is a binary high-level question about the scene that is basically the indicator of a certain event in the scene such as presence of an object from a specific category inside a particular image patch *e.g.,* presence of a plate instance in the upper left quadrant of the image. Note that we consider an object instance to be present in an image patch only if the instance is fully inside the patch. We define an annobit for every object category of interest $c \in \mathcal{C}$ and every patch from an "annocell hierarchy". The annocell hierarchy, which will be discussed later in this chapter, is basically a coarse-to-fine partitioning of the input image I into square patches with varying size at different locations.

■ *Annotation integer* (*Annoint* in short): an annoint is a composite interpretation unit determined by grouping the set of annobits corresponding to the same image patch but associated with different categories *i.e.,* $(Y_{i_1}, Y_{i_2}, ..., Y_{i_{|\mathcal{C}|}})$ for $(i_1, i_2, ..., i_{|\mathcal{C}|})$ being those annobits' indices. Note that an annoint can be equivalently represented by a non-negative integer from 0 to $(2^{|\mathcal{C}|} - 1)$ representing the subset of object categories from $\mathcal{C}$ present in the patch; the value 0 indicates that there is no instance from the object categories of interest present in the patch (all of the $|\mathcal{C}|$ corresponding annobits being zero), and $(2^{|\mathcal{C}|}-1)$ indicates that there is at least one instance from every object category of interest present in the patch (all of the $|\mathcal{C}|$ corresponding annobits being one).

Our objective is to recover the ground-truth interpretation units (annobits or annoints). To do so we design a set of predictors $\mathbf{X} = \{X_i\}_{i=1}^{K}$ that target interpretation units in one-to-one correspondence. These predictors are basically classifiers based on image pixels or low level features. The classifiers may target multiple object categories at once for a given test patch which, for instance, is the case for classifier based on deep Convolutional Neural Networks (CNN) with Multinomial Logistic Regression (SoftMax) output layer [49]. To compute good deep discriminative features the input patch undergoes extensive computation by passing through multiple convolutional and fully connected layers to finally get fed into a relatively cheap output layer for classification. Hence, it makes sense to classify all categories together once the features are computed. In the case of classifiers targeting all categories together we set up the EP framework based on the composite representation of interpretation units *i.e.,* annoints. For notational convenience, and since annoints are determined by annobits, we will represent interpretation units by $\mathbf{Y}$ in the rest of this dissertation unless we want to emphasize on the composite representation where we denote annoints by $\mathbf{Y^c} = \{Y_i^c\}_{i=1}^{K'}$, where $Y_i^c = (Y_{i_1}, Y_{i_2}, ..., Y_{i_{|\mathcal{C}|}})$ and $K' = \frac{K}{|\mathcal{C}|}$ where K denotes the number of annobits.

In the general case, we can also have "derived annobits" which are given as functions of "basic annobits" introduced initially. Basic annobits are the primary or atomic interpretation units defined as presence indicators of at least one instance of a particular category from $\mathcal{C}$ in the corresponding patch. The derived annobits can be introduced to account for more complex and comprehensive events in the scene including (i) union of events such as super-categories of similar object categories and (ii) compositions of different object categories that encode, for instance, specific contextual and spatial configurations of multiple objects *etc*. For example we can consider a "utensil" super-category which combines the basic categories of "fork", "knife" and "spoon". In the same context, a composition would refer to co-existence of instances from more than one object category in the corresponding patches *e.g.,* "plate-fork" or "bowl-spoon" compositions.

The entropy pursuit approach relies on a suitable joint prior defined over all the interpretation units $p(\mathbf{Y})$, and it attempts to recover the overall interpretation of an unlabeled test image by predicting its individual interpretation units step by step in a way that efficiently reduces their remaining uncertainty given the accumulated evidence. We define the joint distribution of annobits (or annoints) via a model which captures relations among multiple variables such as the number of different objects in the scene, their relative poses, and the scene geometry.

The bottom-up image features, encoded by the predictors $\mathbf{X}$, and top-down contextual information, encoded using the prior model $p(\mathbf{Y})$, can be combined in a Bayesian framework as below:

$$p(\mathbf{Y}|\mathbf{X}) = \frac{p(\mathbf{Y}) \times p(\mathbf{X}|\mathbf{Y})}{p(\mathbf{X})} \propto p(\mathbf{Y}) \times p(\mathbf{X}|\mathbf{Y}), \tag{2.1}$$

where, $p(\mathbf{Y}|\mathbf{X})$ is the posterior distribution on interpretation units $\mathbf{Y}$ given the result of our image scrutiny based on $\mathbf{X}$, and $p(\mathbf{X}|\mathbf{Y})$ is the data model term. The Bayesian approach to object recognition has been advocated and used in quite a few works in the past (*e.g.,* see [47, 55–57, 98, 104]). However, the critical question that has remained unaddressed is how much image scrutiny suffice, namely how many patches from the image need to be processed, to come up with scene interpretation and how should the image patches get selected to maximize the information gain which is the question we attempted to address in this dissertation. Estimating the ground truth interpretation units with minimum level of image scrutiny is particularly beneficial as classifiers become more computationally demanding (in the case of CNNs as they become deeper).

Entropy Pursuit was first introduced by Geman and Jedynak in [27] for road tracking in satellite images. This approach was later used in some other applications *e.g.,* see [95] where it was used for face detection and localization, and [96] where it was used for instrument tracking during retinal microsurgery. However, it has never been used for problems such as 3D scene interpretation with a much bigger search space and many more variables. In this dissertation we apply the EP strategy to sequential scene analysis. There is a key advantage in using the EP approach in scene analysis algorithms that use deep classifiers, namely attempting to recover the ground-truth interpretation units based on the fewest number of classifiers applied to the test image. State-of-the-art classifiers based on Convolutional Neural Networks (CNN) offer much better detection rate but yet they require heavy computation *e.g.,* on an end-of-the-line Intel i7-4790K CPU (with 8M Cache and up to 4.40 GHz clock rate) it takes about 12 seconds to process one single patch deploying the VGG network [91] based on the caffe framework's optimized C++ code [45]. Assume we had to process 1000 patches from a given test image and processing each patch would take 12 seconds, this would lead to hours of processing time for one single image on CPU. This is the reason for using GPUs for deep neural networks; the processing time of one single patch on a Tesla K40 GPU is about 0.2 seconds deploying the VGG network. The reason why GPUs are much faster than CPUs with deep neural networks is out of the scope of this dissertation. In this dissertation we investigate if we really need to process all patches or perhaps processing a relatively small fraction of all the patches would suffice to recover the ground-truth interpretation units. In the next section, we review the mathematical foundation of entropy pursuit.

2.1 Mathematical Foundation

In the entropy pursuit approach, we parse (interpret) an input scene image "I" by collecting bits of evidence about the scene encoded by annobits in an optimal order guided by the principle of uncertainty reduction.

For the prediction task, we consider a set of tests $\mathbf{X} = \{X_i\}_{i=1}^{K}$ (boolean tests for annobits and integer tests for annoints) that target the interpretation units $\mathbf{Y}$ in one-to-one correspondence. For example, if Y_i encodes presence of at least one bottle in a given patch then X_i is a bottle detector for that patch; or if Y_i^c encodes presence of at least one bottle and one glass instance in a patch, then X_i^c serves as the predictor of this event. The tests are assumed to have unit cost for our analysis to make sense. We will discuss entropy pursuit's question (*i.e.,* classifier) selection criteria for both binary and composite representations in the next two subsections.

2.1.1 Binary Representation

Assume we have a joint prior distribution on annobits $P(\mathbf{Y} = \mathbf{y})$. The temporal ordering of the tests is encoded via another set of random index variables $\{q_k\}_{k=1}^K$, where $q_k \in \{1, ..., K\}$ is the index of the test applied to image I at step k of entropy pursuit analysis. In other words, q_k is the k^{th} query and $x_{q_k} = X_{q_k}(I)$ is the corresponding (imperfect) answer to this query. Determining which annobit q_k to predict at step k depends on the total evidence obtained from I up to step $k-1$. To put the decision process more formally let $\mathbf{q}_{k-1} = (q_1, ..., q_{k-1})$ denote the list of queries up to step k and $\mathbf{a}_{k-1} = (X_{q_1}(I), ..., X_{q_{k-1}}(I))$ denote the corresponding list of answers, which are the classifiers' responses evaluated on I. Finally, the history of accumulated evidence at step k is denoted by $\mathbf{e}_{k-1} = (\mathbf{q}_{k-1}, \mathbf{a}_{k-1})$. The query q_k selected at the k-th step is a function of accumulated history up to step k determined by:

$$q_k(\mathbf{e}_{k-1}) = \underset{1 \le i \le K}{\operatorname{argmin}} H(\mathbf{Y}|\mathbf{e}_{k-1}, X_i), \tag{2.2}$$

where $H(.)$ stands for entropy. The above selection criterion translates as follows: "Let the k^{th} query q_k be the index i whose corresponding answer X_i minimizes the remaining uncertainty on the overall interpretation $\mathbf{Y}$ given evidence $\mathbf{e}_{k-1}$". Based on this intuitive principle, it is clearly not efficient to search the space of annobits by asking questions whose answers are highly predictable. For example, one would not ask "Is it an urban scene?" after already having got a positive response to "Is there a skyscraper?" nor would one ask if there is an object instance from category c in patch "A" if we already know it is highly likely that there is an object instance from category c in patch "B" that is a subset of "A" ($B \subseteq A$). We can interpret the question selection criterion of (2.2) slightly differently as follows:

$$\begin{aligned} q_k(\mathbf{e}_{k-1}) &= \underset{1\le i\le K}{\operatorname{argmin}}\, H(\mathbf{Y}|\mathbf{e}_{k-1}, X_i) = \underset{1\le i\le K}{\operatorname{argmin}}\big(H(\mathbf{Y}, X_i|\mathbf{e}_{k-1}) - H(X_i|\mathbf{e}_{k-1})\big) \\ &= \underset{1\le i\le K}{\operatorname{argmin}}\big(H(\mathbf{Y}|\mathbf{e}_{k-1}) + \underbrace{H(X_i|\mathbf{Y}, \mathbf{e}_{k-1}) - H(X_i|\mathbf{e}_{k-1})}_{-\mathcal{I}(X_i,\mathbf{Y}|\mathbf{e}_{k-1})}\big) \\ &= \underset{1\le i\le K}{\operatorname{argmax}}\, \mathcal{I}(X_i, \mathbf{Y}|\mathbf{e}_{k-1}), \qquad (2.3) \end{aligned}$$

where $\mathcal{I}(.,.)$ denotes mutual information. According to (2.3), in each step of EP we basically select the most informative query about the partial interpretation units $\mathbf{Y}$.

In the ideal case where the tests $\mathbf{X}$ are good predictors of annobits $\mathbf{Y}$, or more generally if $\mathbf{Y}$ determines X_i *i.e.,* $X_i = f(\mathbf{Y})$, equation (2.2) will be equivalent to:

$$q_k(\mathbf{e}_{k-1}) = \underset{1\leq i\leq K}{\operatorname{argmax}} H(X_i|\mathbf{e}_{k-1}). \tag{2.4}$$

The above is valid because in the ideal case where $X_i = Y_i$, or $X_i = f(\mathbf{Y})$, we have $H(X_i|\mathbf{Y}, \mathbf{e}_{k-1}) = 0$; hence, (2.3) leads to the question selection criterion in (2.4). Considering that tests tend to mimic their corresponding annobits, in the sense that X_i is a "good" predictor of Y_i, we may expect similar orderings among the entropies $\{H(X_i|\mathbf{e}_{k-1})\}_{i=1}^K$ and the entropies $\{H(Y_i|\mathbf{e}_{k-1})\}_{i=1}^K$. In that case, we can approximate (2.4) for $X_i = Y_i$ with:

$$q_k(\mathbf{e}_{k-1}) = \arg\max_{1\leq i\leq K} H(Y_i|\mathbf{e}_{k-1}) = \arg\min_{1\leq i\leq K} |P(Y_i = 1|\mathbf{e}_{k-1}) - \frac{1}{2}|. \tag{2.5}$$

The query selection criterion in (2.5) is basically what we use in practice. As in other Bayesian settings, we also represent the conditional distribution of variables in $\mathbf{X}$ conditional to $\mathbf{Y}$ *i.e.,* $p(\mathbf{X}|\mathbf{Y})$. But no matter how "convenient" this data model, the interactions among the variables in $\mathbf{Y}$ that encode a scene usually cannot be organized in a tree, thereby making the computation of the marginal $P(\mathbf{X})$ and in particular $P(X_i|\mathbf{e}_{k-1})$ hard. Moreover, we attempt to solve the entropy optimization of (2.4) online and many times; this is the reason for using the approximate criterion in (2.5). Note that it may be more practical to ask questions in batches because the posterior distribution most likely will not be affected significantly by asking only one question.

2.1.2 Composite Representation

The EP question selection criterion for composite representation is very similar to the binary representation discussed in the previous subsection. Assuming that the designed tests are good predictors of interpretation units leads to $H(X_i^c|\mathbf{Y^c}, \mathbf{e}_{k-1}^c) = 0$, and hence:

$$q_k^c(\mathbf{e}_{k-1}^c) = \operatorname*{argmin}_{1\leq i\leq K'} H(\mathbf{Y^c}|\mathbf{e}_{k-1}^c, X_i^c) = \operatorname*{argmax}_{1\leq i\leq K'} H(X_i^c|\mathbf{e}_{k-1}^c), \tag{2.6}$$

where $q_k^c \in \{1, ..., K'\}$ and $\mathbf{a}_{k-1}^c = (X_{q_1}^c(I), ..., X_{q_{k-1}}^c(I))$ in $\mathbf{e}_{k-1}^c = (\mathbf{q}_{k-1}^c, \mathbf{a}_{k-1}^c)$. This means that the most informative query is the one whose answer's distribution over $\{0, 1, ..., 2^{|\mathcal{C}|} - 1\}$ is closest to uniform. The approximate criterion for the composite representation is:

$$q_k^c(\mathbf{e}_{k-1}^c) = \operatorname*{argmax}_{1\leq i\leq K'} H(Y_i^c|\mathbf{e}_{k-1}^c). \tag{2.7}$$

The composite representation is what we will use throughout this dissertation since the classifiers we will use are based on state-of-the-art convolutional neural networks with Multinomial Logistic Regression (softmax) output that target multiple object categories at once *i.e.,* annoints. The CNN classifier designed for $|\mathcal{C}|$ different number of object categories has $(|\mathcal{C}| + 1)$ outputs where each output value estimates the existence proportion ratio (relative probability) of the corresponding category (presence of at least one instance of this category) in the input patch except the first output that estimates the proportion ratio of the complement category set *i.e.,* relative probability of existing no object instance from the category set $\mathcal{C}$.

A more extensive description of classifiers will be given later in chapter 5.

2.2 Annocell Hierarchy

We construct a coarse-to-fine hierarchy of "annotation cells", or in short "annocells", where each annocell is a square patch. The size of patches get smaller from top to the bottom of the annocell hierarchy. We first pad and center the input image I to a square whose size is the maximum of the number of rows and columns of the input image denoted by D_{max} leading to a $D_{\text{max}} \times D_{\text{max}}$ pixels padded image. We then partition the padded image using sliding windows at different sizes to build the annocell hierarchy (see Figure 2.6). We build a 4-level hierarchy whose patch size at the l-th level is $D_{\text{max}}/2^l$ where $l \in \{0, 1, 2, 3\}$. The row and column shift ratio for every level is 25% which leads to 75% overlap between nearest windows at each level. This leads to 1, 25, 169, and 841 number of patches for, respectively, level-0, level-1, level-2, and level-3 (a total of 1036 annocells). Figure 2.6 shows some annocells chosen from all of the four levels of the hierarchy. Associated with each of these canonical annocells in the hierarchy there are $|\mathcal{C}|$ basic annobits. Hence, the total number of basic annobits is $K = 1036 * |\mathcal{C}|$ and the total number of annoints is $K' = 1036$. Note that each annobit serves as the presence indicator of at least one instance from the corresponding category; the instance should be fully inside the corresponding annocell to turn on the annobit. The formal definition of an annobit is as below:

$$Y_i := \mathbf{1}\{\exists \text{ at least one object instace from category } c_i \text{ in } A_i\}, \tag{2.8}$$

where A_i denotes the corresponding annocell. The ordering of annobits obviously does not matter and we can choose any ordering; we may choose an ordering where annobits associated with the first category are indexed $1 \leq i \leq 1036$ (each corresponding to a different annocell out of the 1036 canonical annocells), annobits associated with the second category are indexed $1037 \leq i \leq 2072$, and so on.

In the next chapter, we will describe our prior model design which is a key component in entropy pursuit scene analysis.

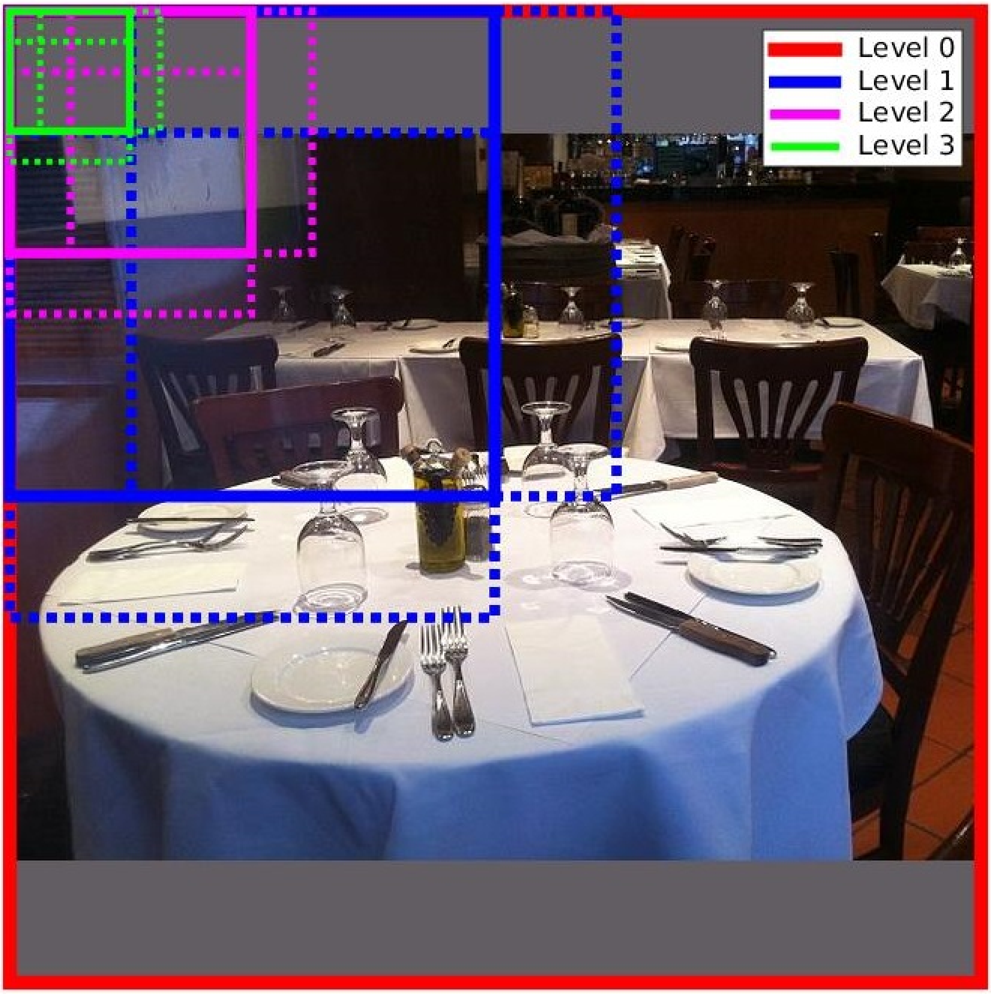


Figure 2.6: Some selected cells from different levels of the annocell hierarchy. Rectangles with dashed lines are the nearest neighbor patches to the rectangles with solid lines from the same color.

Chapter 3

Prior Model

Entropy pursuit scene interpretation relies on a suitable prior model on the interpretation units $p(\mathbf{Y})$. According to the EP criterion ((2.5) and (2.7)) we choose an interpretation unit at step k whose marginal posterior $p(Y_i|\mathrm{e}_{k-1})$ has maximal entropy. Using the Bayes rule we have:

$$p(\mathbf{Y} \mid \mathbf{e}_{k-1}) = \frac{p(\mathbf{Y}, \mathbf{e}_{k-1})}{p(\mathbf{e}_{k-1})} \propto p(\mathbf{Y}) \times p(\mathbf{e}_{k-1} \mid \mathbf{Y}), \tag{3.1}$$

where, $p(\mathbf{e}_{k-1} \mid \mathbf{Y})$ is determined by the data model $p(X \mid Y)$ and $p(\mathbf{Y})$ is the prior model on the interpretation units defined in the image coordinate system. However, it is hard to directly construct and learn a prior model defined on the interpretation units to estimate the marginal posterior distributions $p(Y_i|\mathrm{e}_{k-1})$. This is mainly due to the fact that the interpretation units are defined in the image's coordinate system and can represent very different geometrical properties and relations in the world coordinate system for different camera position, orientation, and calibration parameters. On the other hand, a proper model should consider the 3D geometry of the scene otherwise the incorporated contextual relations will not in essence go beyond simple 2D relations *e.g.,* sky is expected to appear at the top of the image and floor at the bottom. Hence, instead of designing the prior model directly in the image coordinate system we design a generative viewpoint-independent prior model in the world coordinate system. This also leads to interpretable model parameters since the model parameters have physical meaning. Each sample from this model is basically a 3D scene description. We project samples from the model to the image coordinate system via perspective projection assuming a pinhole camera model and evaluate interpretation units. The marginal posteriors $\{p(Y_i|\mathrm{e}_{k-1})\}_{i=1}^{K}$ are then estimated by aggregating the $\mathbf{Y}$ samples.

Motivated by our application to table-settings scenes we assume that the scene contains a dominant world plane (the table plane) where different objects lie (spoons, plates, cups, *etc.*). Our 3D probabilistic model is at the level of objects rather than low-level image features and allows for encoding expected properties and multi-object relationships among a distinguished family of objects, the number of objects in the scene and their relative poses. The underlying graph in the model is a forest of directed trees which captures a natural generative process in which objects are placed down on the surface in stages; the number of root nodes (*e.g.,* a place setting instance) refers to the conditional placement of an object instance relative to the size and geometry of the scene; an edge from a parent (*e.g.,* a plate instance) to a child (*e.g.,* a utensil instance) refers to the conditional placement of the child relative to the parent and so forth (see Figure 3.1). Moreover, edges are only allowed between certain types of object categories and these restrictions are imposed by a “Master Graph”. Designing 3D models which encode favored relationships but still accommodate real-world variability is not straightforward. In particular, given purely object-annotated image data, learning is complicated because the homography between the world and the camera plane, and the graph structure encoding object relations are hidden.

Many human-created scenes are composed of parallel supporting surfaces where different objects lie [4]. Hence, the proposed generative model can be extended to a much wider range of scene types beyond the table-settings scenes. However, we present the model by examples from table-setting scenes. In that case, we consider $\mathcal{C} = \{\text{plate}, \text{bottle}, \text{glass}, \text{utensil}\}$ which are amongst the most annotated categories in our table-setting dataset. Instances from $\mathcal{C}$ are placed on a table whose geometric properties are denoted by T. In the simplified case that the table is rectangular we have $T = (T_L, T_W)$ where T_L and T_W, respectively, represent the length and width of the table. We model table-setting scenes in the world coordinate system given table geometry using a generative attributed graph model described later in the next section. We consider a world coordinate system whose origin is located at the center of the table's surface, whose z axis is orthogonal to the table's surface, and assuming a rectangular table, the x and y axes are parallel to the edges of the table as illustrated in Figure 3.2. We also define a coordinate systems attached to the camera as shown in Figure 3.2. Note that we may sometimes refer to the world coordinate system as the table coordinate system and to the camera coordinate system as the image coordinate system.

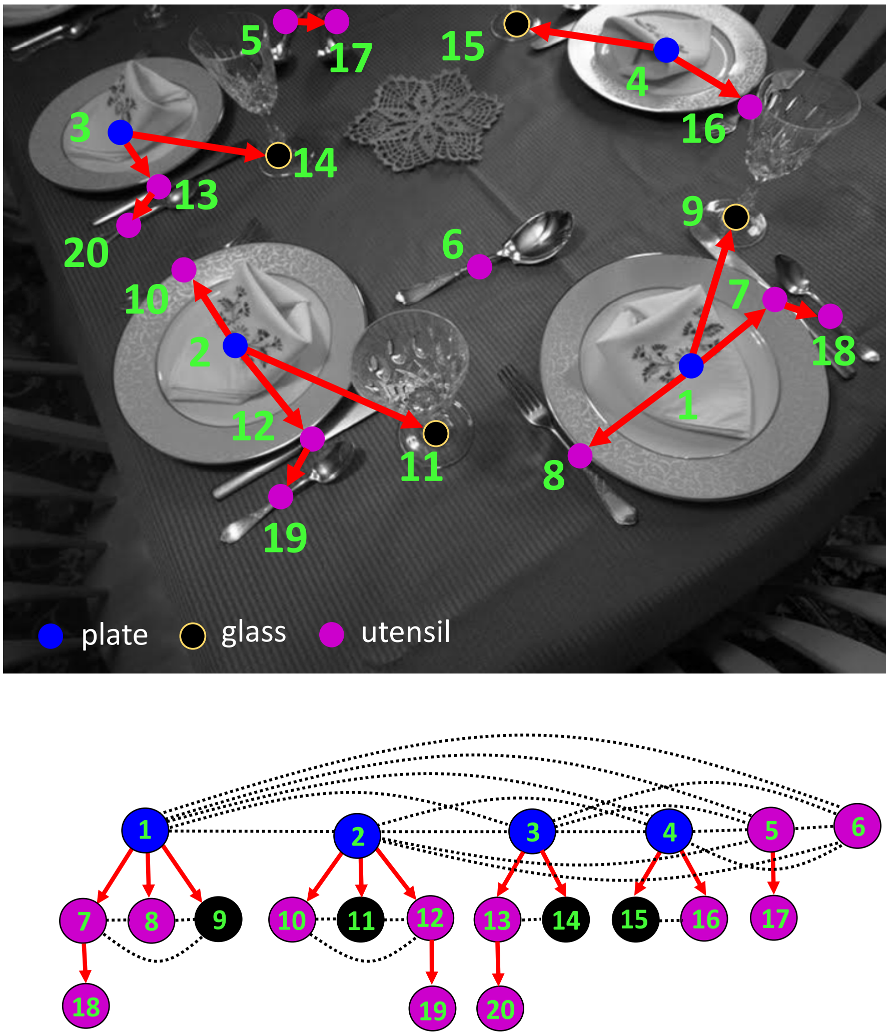


Figure 3.1: A table-setting scene (top) and its corresponding category-labeled base graph (bottom) where the categories (plate, bottle, glass, and utensil) are color-coded in the graph. Root nodes V_0 initialize the generative process; here there are six. The terminal nodes for this instance are $V_T = \{6, 8, 9, 10, 11, 14, 15, 16, 17, 18, 19, 20\}$. According to the base graph $n_{(0,\text{plate})} = 4$, $n_{(0,\text{bottle})} = 0$, $n_{(0,\text{glass})} = 0$, and $n_{(0,\text{utensil})} = 2$. Removing the undirected edges (dashed lines) leads to a Bayesian tree.

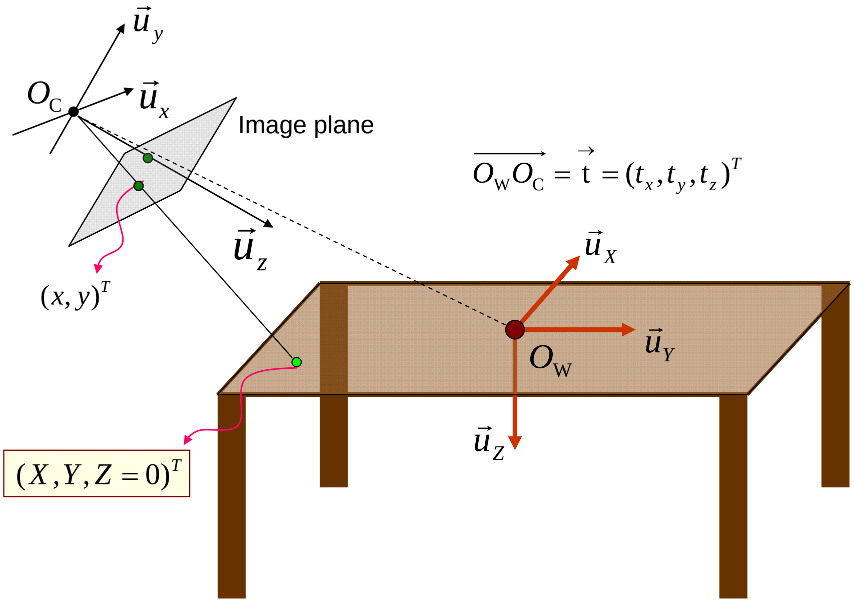


Figure 3.2: Table and camera coordinate systems.

Projection of a 3D point in the world coordinate system onto the image plane can be calculated using a perspective projection matrix (homography) that is a deterministic function of the camera's extrinsic (camera's pose in the world coordinate system) and intrinsic (calibration matrix) parameters (see Appendix A). We denote the set of all extrinsic and intrinsic parameters by $\mathcal{W}$. Hence, given $\mathcal{W}$ (see Appendix B) we can calculate the pose of an object instance in the image plane, denoted by ξ, from its corresponding pose in the world coordinate system, denoted by θ. We can approximate the 3D shape of an object by an enclosing ellipsoid, which encodes the 3D object pose (location and orientation of the ellipsoid) and scale (lengths of the three major axes). Observe that the projection of an ellipsoid is an ellipse, and that for nearly planar objects, the ellipsoid will degenerate to an ellipse. Also, under the assumption that the height of objects is small relative to their distance from the camera we can assume that objects are planar. Moreover, if we parameterize an ellipse in the scene plane by a 3×3 matrix $\theta \in \mathbb{R}^{3\times3}$, then we can find the projection of this ellipse in the image plane via $\xi = H^{-\top}\theta H^{-1}$, where H is the homography from the scene to the image plane (see Appendix B for details about ellipse representation and projection).

The joint distribution over all variables is defined by:

$$p(g, T, \mathcal{W}) = p(g|T)\,p(\mathcal{W})\,p(T), \tag{3.2}$$

here, $p(g|T)$ is the distribution of the number and 3D poses of different objects via the attributed graph model conditioned on table geometry. In (3.2), we assumed that $\mathcal{W}$ (the set of camera's extrinsic and intrinsic parameters) is independent of the table geometry T, and since the attributed poses of the graph g are defined in the world coordinate system g is independent of $\mathcal{W}$.

Let ξ_V denote the set of 2D poses in the image coordinate system whose corresponding 3D poses θ_V are attributes of g; the set of vertices of g are denoted by V. With this notation, we may write:

$$p(\xi_V, g, T, \mathcal{W}) = p(\xi_V|g, \mathcal{W})\,p(g|T)\,p(\mathcal{W})\,p(T). \tag{3.3}$$

However, since the set of object poses in the image coordinate system ξ_V can be calculated deterministically from the corresponding poses in the world coordinate system θ_V

and $\mathcal{W}$, the leftmost distribution $p(\xi_V|g, \mathcal{W})$ will be singular, resulting in a degenerate joint distribution (3.3).

The joint distribution (3.2) can model table-setting scenes, and is fully specified by choosing appropriate distributions for the table geometry T, camera parameters $\mathcal{W}$, and the conditional model $p(g|T)$. We describe each of the specifying distributions in the following sections of this chapter.

3.1 Generative Attributed Graph Model

A scene is described as a collection of object instances from different categories at different poses. Each object instance is associated with a vertex $v \in V$ of a *base* graph $g_0 \in \mathcal{G}_0$ which captures contextual relationships among object instances. An attributed graph is a triple $g = (g_0, c_V, \theta_V)$, where $c_V = \{c_v, v \in V\}$ and $\theta_V = \{\theta_v, v \in V\}$ denote the set of category labels and 3D poses of objects, respectively. The categories are restricted to a set $\mathcal{C}$ of size $|\mathcal{C}|$. The root nodes are denoted by $V_0 \subset V$, the terminal nodes by $V_T \subset V$, the parent of node v by $pa(v)$ and the set of its children by $ch(v)$. Obviously, $pa(v) = \emptyset$ for a root node $v \in V_0$, and $ch(v) = \emptyset$ for a terminal node $v \in V_{\text{T}}$. Given $g \in \mathcal{G}$ (the space of attributed graphs), $v \in V$, and $c \in \mathcal{C}$ we define:

$$n_{(0,c)} = n_{(0,c)}(g) = \sum_{v \in V_0} I_{\{c_v = c\}}, \tag{3.4}$$

which is the number of root nodes of type c;

$$n_{(v,c)} = n_{(v,c)}(g) = \sum_{v' \in V} I_{\{pa(v')=v, c_{v'}=c\}}, \tag{3.5}$$

which is the number of children of v of type c; and

$$n_{(c)} = n_{(0,c)} + \sum_{v \in V} n_{(v,c)}, \tag{3.6}$$

indicates the total number of object instances of type c. Note that $\sum_c n_{(c)} = |V|$.

The model is a probability distribution $p(g|T),\ g \in \mathcal{G}$, on the space of (equivalence classes of) attributed graphs $\mathcal{G}$ conditioned on the environment's geometric properties T. An attributed graph sample specifies a 3D scene sample with existing objects' category labels and 3D poses together with the relations between them. The generative attributed graph (GAG) model is specified by the following four sets of distributions:

1. $p^{(0)}(n_{(0,1)}, \cdots, n_{(0,|\mathcal{C}|)}|T)$: this is the conditional joint distribution of the number of root nodes from various categories $\{n_{(0,c)}\}_{c=1}^{|\mathcal{C}|} \in \mathbb{N}_0^{|\mathcal{C}|}$ where $\mathbb{N}_0 = \{0, 1, 2, \dots\}$.

2. $\{p^{(c)}(n_{(v,1)}, \cdots, n_{(v,|\mathcal{C}|)}),\ c_v = c \in \mathcal{C}\}$: for a parent node v from category c, $p^{(c)}(.)$ is the joint distribution of the number of children $\{n_{(v,c)}\}_{c=1}^{|\mathcal{C}|}$ from each object category c. These distributions can be thought of as the basis distributions in a multi-type branching process [63]. To limit the complexity of the model, we will also restrict the set $\mathcal{G}$ to graphs with bounded depth (distance to the closest root), denoting the upper

limit by l_{max}. This is equivalent to setting $p^{(c_v)}(0, 0, ..., 0) = 1$ for node v located at the maximum allowed depth l_{max}. We assume that this distribution is independent of the environment's geometry T.

3. $p(\theta_{V_0}|c_{V_0}, T)$: this is the conditional joint pose distribution of the root nodes given their corresponding category labels and T.

4. $\{p(\theta_{ch(v)}|c_{ch(v)}, c_v, \theta_v, T), \, v \in V \backslash V_{\text{T}}\}$: this is the joint distribution of the poses of the children of v given their parent's pose and the corresponding category labels and T.

The category-labeled base graph (g_0, c_V) distribution can be specified based on the first and second distributions above as follows:

$$p(g_0, c_V|T) = \, p^{(0)}(n_{(0,1)}, \cdots, n_{(0,|\mathcal{C}|)}|T) \times \prod_{v \in V \backslash V_{\text{T}}} p^{(c_v)}(n_{(v,1)}, \cdots, n_{(v,|\mathcal{C}|)}), \tag{3.7}$$

where, again $n_{(v,c)}$ denotes the number of children of vertex v from the c-th category. The above distribution can be sampled in a top-down fashion by first sampling the number and category of root nodes, and then sampling the children and repeating this process until reaching the maximum allowed depth l_{max}. A base graph with a flexible and random skeleton according to (3.7) allows for a number of plausible 3D arrangements in content. The full distribution on attributed graphs $g \in \mathcal{G}$ is:

$$\begin{aligned} p(g|T) &= p(g_0, c_V|T) \times p(\theta_V|g_0, c_V, T) \\ &= p^{(0)}(n_{(0,1)}, \cdots, n_{(0,|\mathcal{C}|)}|T)\, p(\theta_{V_0}|c_{V_0}, T)\times \\ &\prod_{v\in V\setminus V_{\mathrm{T}}} p^{(c_v)}(n_{(v,1)}, \cdots, n_{(v,|\mathcal{C}|)}) \times p(\theta_{ch(v)}|c_{ch(v)}, c_v, \theta_v, T). \end{aligned} \tag{3.8}$$

Note that the previous expressions define probabilities of sets of variables without defining a specific labeling order for objects within the same category; they should be understood as probabilities over equivalence classes up to such relabeling.

Assuming the poses of children are conditionally independent given their corresponding category labels, the pose of their parent, and the geometric properties of the supporting surface, the graph reduces to the standard *Bayesian* forest *i.e.,* Directed Acyclic Graph (DAG). Otherwise, we will have a hybrid graph with both directed and undirected edges. Figure 3.1 illustrates an example scene and its corresponding base graph. Removing the dashed undirected edges in the hybrid graph (shown in the bottom figure) leads to the Bayesian forest overlaid on top of the table-setting scene (shown at the top). The undirected edges in the hybrid graph can be used to provide more global structure among the poses of objects corresponding to a tree or subtree from the base graph forest, the minimal use being to avoid impossible or downweight unlikely overlaps (*e.g.,* two glasses on top of each other). The proposed model can prevent such configurations to some extent by assigning a zero or small probability to “forbidden” pose combinations among the children of a

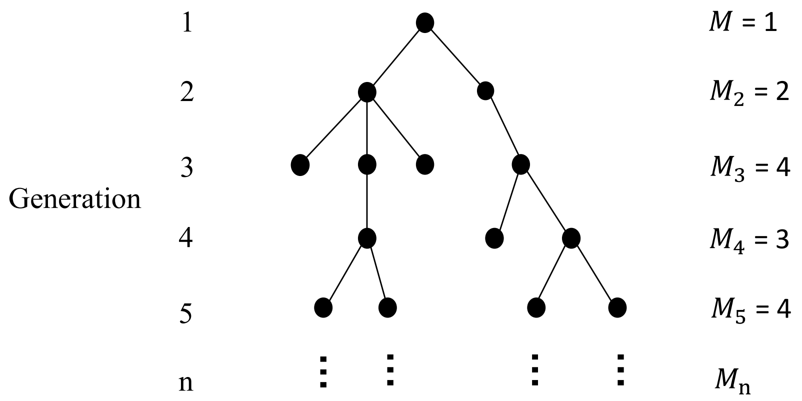


Figure 3.3: A branching process example. M_n indicates the total number of instances generated in the n^{th} generation.

given node (but cannot address possible object overlaps on the global level, which would require connecting all of the relevant nodes resulting in a significant increase in complexity, breaking the top-down nature of the model).

As was mentioned earlier, the probability distributions $p^{(c)}(.)$ can be thought of as the basis distributions in a multi-type branching process. A branching process is a Markov process that is usually used to model a population where each individual in the n-th generation generates some random number of individuals in the $(n+1)$-th generation. In the original Galton-Watson branching process individuals, which are also referred to as particles, are indistinguishable as shown in Figure 3.3 by the same color circles. But, in the multi-type extension distinguishable type of particles are allowed where each type has a different probabilistic behavior. In both cases, a particle generates offspring particles independent of other particles within the same generation. However, there are two fundamental differences between a typical branching process analysis and what we care about here.

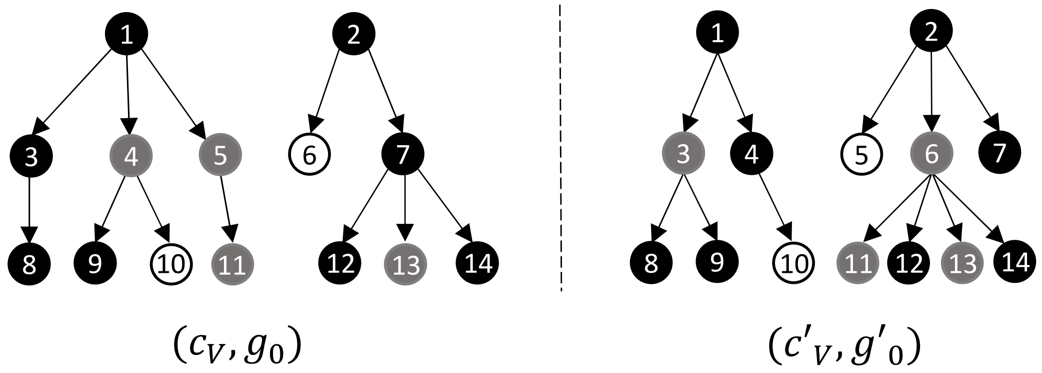


Figure 3.4: Two distinct category-labeled base graphs with exactly the same counts per category at each level. In the graphs, different node colors represent different categories.

First, by allowing at most l_{max} number of levels we are not anymore concerned with the extinction probability calculation whereas asymptotic analysis and calculating the extinction probability is of particular interest in the analysis of a branching process (see [1, 63] for more details). Secondly, branching process analysis is agnostic to the graph structure and is concerned with modeling the population size of each type at each generation whereas the generative process for (c_V, g_0) is only partially determined by the count statistics. That is, we can have two distinct category-labeled graphs (c_V, g_0) and (c'_V, g'_0) with exactly the same counts per category at each level but different structure and probabilities (Figure 3.4 shows an example of this case).

Obviously, some category pairs have stronger expected contextual relationships than others, and it is reasonable to assume $p^{(c_i)}(n_{(v,1)}, \ldots, n_{(v,|\mathcal{C}|)}) = 0$ for configurations with $n_{(v,j)} > 0$ for certain pairs c_i and c_j. To capture these preferred relationships, we define a directed "Master Graph", $G_M = (\mathcal{V}_M, \mathcal{E}_M)$, over the set of object categories $\mathcal{C}$, which

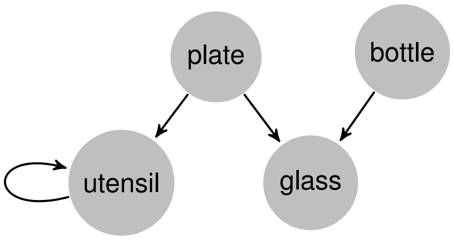


Figure 3.5: An example master graph

constrains the branching probabilities $p^{(c)}(\cdot)$. Every vertex in G_M corresponds to one object category in $\mathcal{C}$ and every edge $(c_i \to c_j) \in \mathcal{E}_M$ indicates existence of an allowed generative relationship between categories c_i and c_j; that is, a vertex of category c_i will not generate a vertex of category c_j if there is no directed edge from c_i to c_j in G_M. An appropriate master graph can be partially or fully user-determined by a combination of a priori contextual domain knowledge together with some learning. For example, an undirected version might be estimated from annotated data by assigning weights to the edges of a fully-connected context graph over the set of object categories based on frequencies of sufficiently local co-occurrence and then thresholding to prune edges with weak co-occurrences. Directions could then be assigned by ordering the object categories in some fashion, *e.g.,* larger to smaller, or incorporating knowledge about how the scenes of interest are created. Figure 3.5 shows an example master graph for table-setting scenes with four object categories.

The distribution of the numbers of root nodes from each category in g obviously depends on the size of the objects relative to the available surface space. We assume that given T (and of course the scene type) the number of root nodes from different categories are independent:

$$p^{(0)}(n_{(0,1)},...,n_{(0,|\mathcal{C}|)}|T) = \prod_{i=1}^{|\mathcal{C}|} p_u^{(0)}(n_{(0,i)}|T), \tag{3.9}$$

where each of the univariate conditional distributions can be modeled using a *Poisson* distribution with an average rate proportional to the environment area T_A resulting in $|\mathcal{C}|$ parameters *i.e.,* one parameter per object category (proportionality ratio). We also decouple the offspring counts:

$$p^{(c_v)}(n_{(v,1)},...,n_{(v,|\mathcal{C}|)}) = \prod_{i=1}^{|\mathcal{C}|} p_u^{(c_v)}(n_{(v,i)}). \tag{3.10}$$

In the case of nonparametric distributions, this can reduce the number of model parameters from combinatorial to linear in the number of categories $|\mathcal{C}|$. Consider a nonparametric probability mass function with a finite support size for the univariate distribution $p_u^{(c_1)}(n_{(v,c_2)})$ such that it is non-zero if $0 \leq n_{c_2} < l_{(c_1 \to c_2)}$ and zero otherwise. Since the probabilities over the support must sum up to one, the number of independent parameters of this mass function is $(l_{(c_1 \to c_2)} - 1)$. Therefore, the total number of parameters to model all of the offspring generating distributions is $\sum_{e \in \mathcal{E}_M}(l_{(e)} - 1)$. Note that if (c_1, c_2) is an impossible parent-child pair according to the master graph then $l_{(c_1,c_2)} = 1$, because $p_u^{(c_1)}(n_{(v,c_2)} = 0) = 1$, and therefore the corresponding distribution has zero number of parameter.

In the attributed graph model for table-setting scenes we allow at most three generations (levels) *i.e.,* $l_{\max} = 3$. The root instances are generated according to four independent *Poisson* distributions (one per category). We choose the average rate of the Poisson distributions to be proportional to the area of table *i.e.,* $\lambda_c = \alpha_c T_A,\ c \in \mathcal{C}$. Hence, we need to learn four average rate factors $\alpha_c,\ c \in \mathcal{C}$. The second and third generation instances, denoted by V_1 and V_2, respectively, are generated according to independent non-parametric probability mass functions with finite support size. For the permissible category pairs "$c_1 \to c_2$" consistent with the master graph including "plate → utensil", "plate → glass", "bottle → glass", and "utensil → utensil" we choose the support size of $p_u^{(c_1)}(n_{(v,c_2)})$, namely $l_{(c_1,c_2)}$, to be 3, 3, 4, and 3, respectively. This means that for example we allow at most three utensils to be adopted by a plate instance. We choose $p(\theta_v|c_v, T)$ for all $v \in V_0$ to be two-step-pyramid-shaped distributions that place objects at least d_{c_v} meters away from the edge on the table with probability ρ_{c_v} and on a strip of width d_{c_v} around the table with probability $(1 - \rho_{c_v})$. The distribution is assumed uniform within each of the two areas except that vertical objects *e.g.,* glasses are prevented from overlapping. We assume that the orientation of a spontaneous utensil instance $v \in V_0$ follows a von Mises distribution whose mean is set to be 90 degrees from the orientation of the nearest table edge. The dispersion parameter of the von Mises distribution is set to zero if this instance is located farther than 40 centimeters from all sides of the table and greater than zero otherwise. Note that a von Mises distribution with zero dispersion parameter is basically a uniform distribution in $(0, 2\pi]$. Also, note that the 3D orientation of a bottle or glass instance is assumed to always be normal to the table surface whereas the orientation of their base is uniform in $(0, 2\pi]$. We also consider a uniform orientation distribution for the plate category because of its circular shape. In the general case, we can have a size distribution such as an inverse gamma distribution but since the sizes of table-setting object categories have small variations within categories we decided to keep the object sizes fixed and equal to their typical world sizes for simplicity *e.g.,* the diameter of a plate is typically 25 centimeters. We specify the pairwise pose distributions, $p(\theta_v | c_v, c_{pa(v)}, \theta_{pa(v)}, T)$, by a radial distribution and a conditional angular distribution in a polar system centered at the location of the parent object. We model the relative pose of a parent-child object pair assuming that their relative location is independent from their relative orientation. We chose a scaled beta distribution for the radial distance between pairs of objects and either a Von mises (single or mixture) or uniform distribution for the angular location of the child in the periphery of the parent object. Normally, we expect a c_1-category parent and a c_2-category child to be within some distance from each other in order to justify their local contextual relationship. Let $d_{(c_1,c_2)}$ denote this user-defined distance. The scale of the beta distribution used for the radial distance of a (c_1, c_2) object pair is set to $d_{(c_1,c_2)}$ and kept fixed throughout design and learning. The set of pose distribution parameters therefore includes the set of beta and von Mises distributions' parameters for different categories. Obviously, different probabilistic models may be adopted for different type of scenes and object categories thereof.

In the next section, we introduce a data set of table-setting scenes that we built and used for learning the model.

3.2 JHU Table-Setting Dataset

We collected and annotated about 3000 images of table setting scenes with more than 30 object categories. The images in this dataset were mostly collected using the “Extreme Picture Finder” software which downloads images from multiple sources such as Google, Flickr, Altavista, *etc*. The annotation task was carried out by three annotators over a period of about ten months using the “LabelMe” online annotating website [82]. The consistency of labeled images were then verified and synonymous labels were consolidated. The annotation of this dataset was done with careful supervision resulting in high quality annotations, better than what we normally get from “Amazon Mechanical Turk”. Figure 3.6 shows a snapshot of the “JHU Table-Setting Dataset”.

Figure 3.7 (top) shows the annotation histogram of the 30 most annotated categories, and the bottom figure shows the averaged histogram per image. The average number of annotations per image is about 17. Additionally, the *Homography* matrix for every image in the data set is manually estimated. To estimate the homography (up to scale) at least four pairs of corresponding points are needed according to the Direct Linear Transformation (DLT) algorithm [35, p. 88]. These four pairs of corresponding points were located by annotators’ the best visual judgment about four corners of a square in real world whose center coincides with the origin of the table (world) coordinate system. We are able to undo the projective distortion due to the perspective effect by back-projecting the table surface in the image coordinate system onto the world coordinate system [35]. The homography matrices are scaled appropriately (using object’s typical sizes in real world) such that after

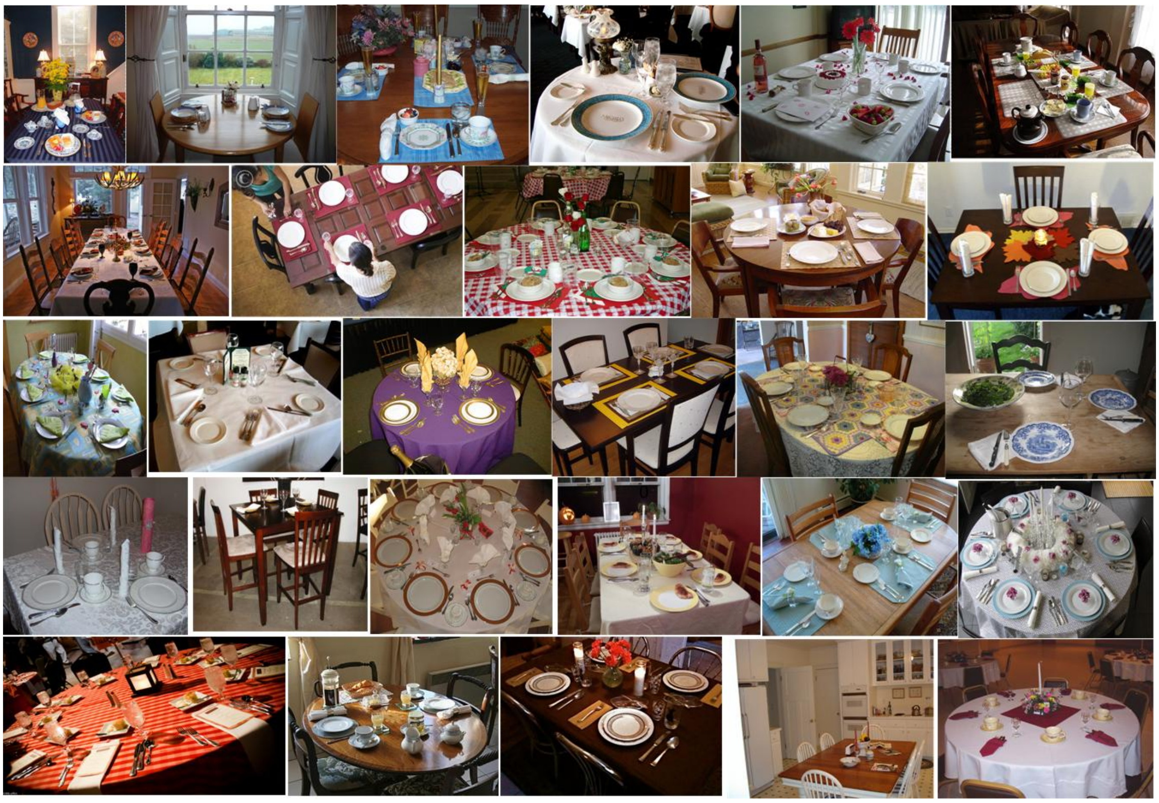
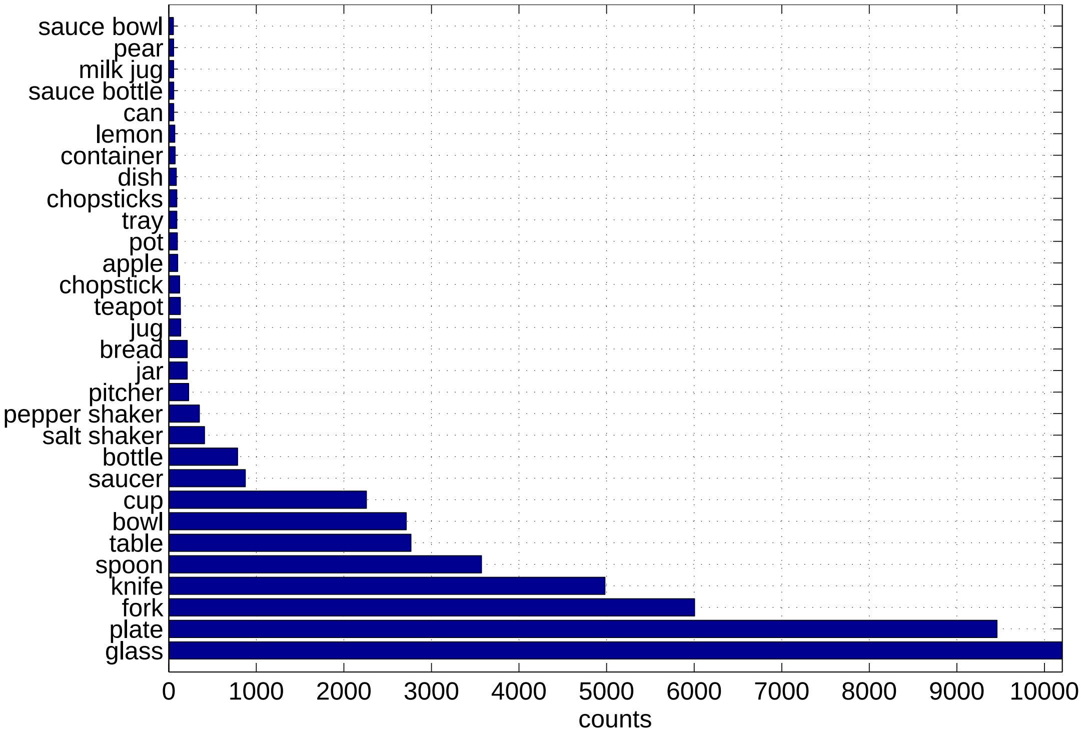


Figure 3.6: A snapshot of the dataset.

back-projection the distance of object instances in the world coordinate system (measured in meters) can be computed. Figure 3.8 shows two typical images from this dataset and their rectified versions.

We also utilized a synthetic table-setting scene renderer developed by Erdem Yörük for verification purposes. This synthetic image renderer inputs the camera's calibration parameters, six rotation and translation camera's extrinsic parameters, table length and width, and 3D object poses in the table's coordinate system and outputs the corresponding table setting scene. Figure 3.9 shows some synthetic images generated by this renderer.

Each object instance was annotated with an object category label plus an enclosing



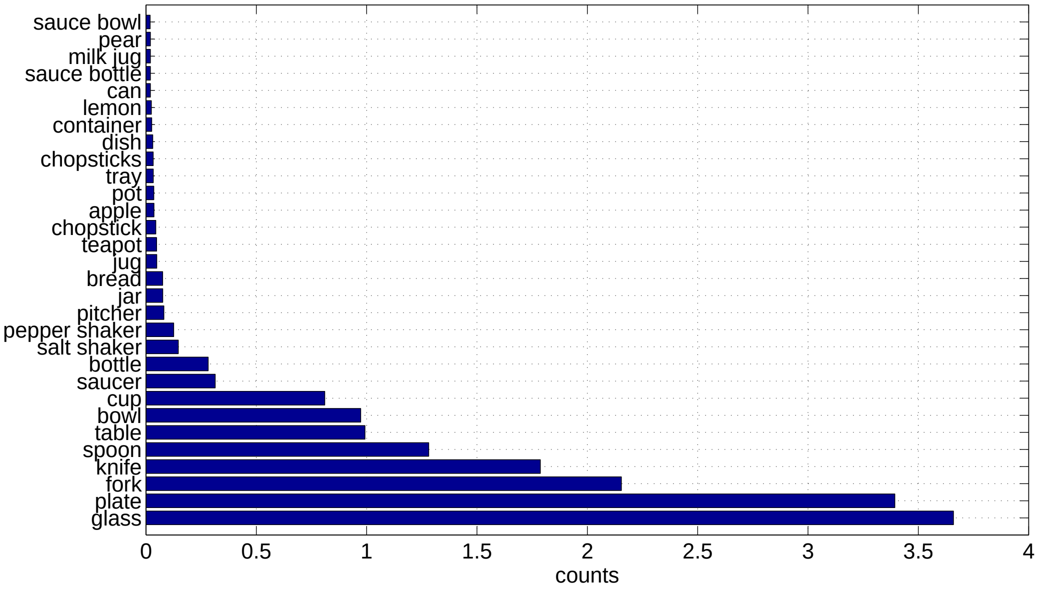


Figure 3.7: (top): The number of annotated instances of each object category in the whole dataset for the 30 top most annotated object categories. (bottom): The average number of annotated instances per object category per image.

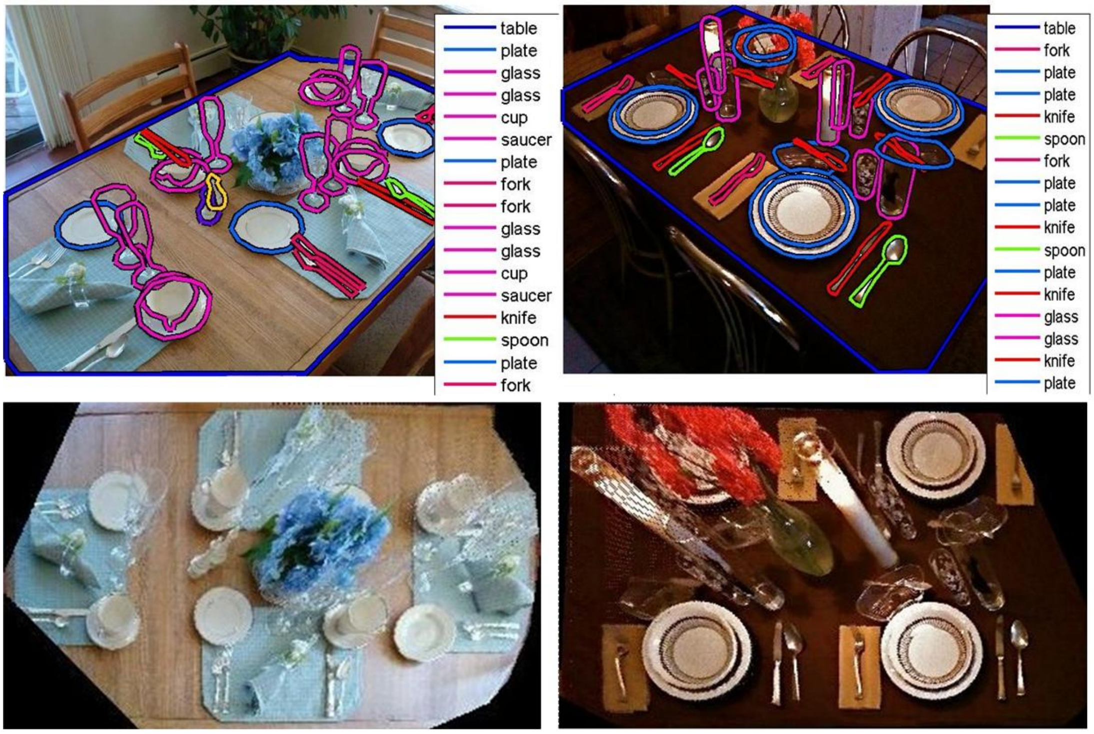


Figure 3.8: Rectification of table surfaces after back-projection.

polygon. Then, an ellipse was fit to the vertices of the polygon to estimate the object's shape and pose in the image plane. Figure 3.10 (left) shows an example annotated image; Figure 3.10 (middle) shows the corresponding back-projection of vertices of annotation polygons for plates (in red), glasses (in green), and utensils (in black). Note that non-planar objects (*e.g.,* glass) often get distorted after back projection (*e.g.,* elongated green ellipses) since the homography transformation is a perspective projection from points on the table surface to the camera's image plane. Hence, we estimated the base of vertical objects (shown by black circles in the middle figure) to estimate their location in the table (world) coordinate system since the center of fitting ellipse to the back-projection of such objects' annotation points is not a good estimate of their 3D location in the real world.

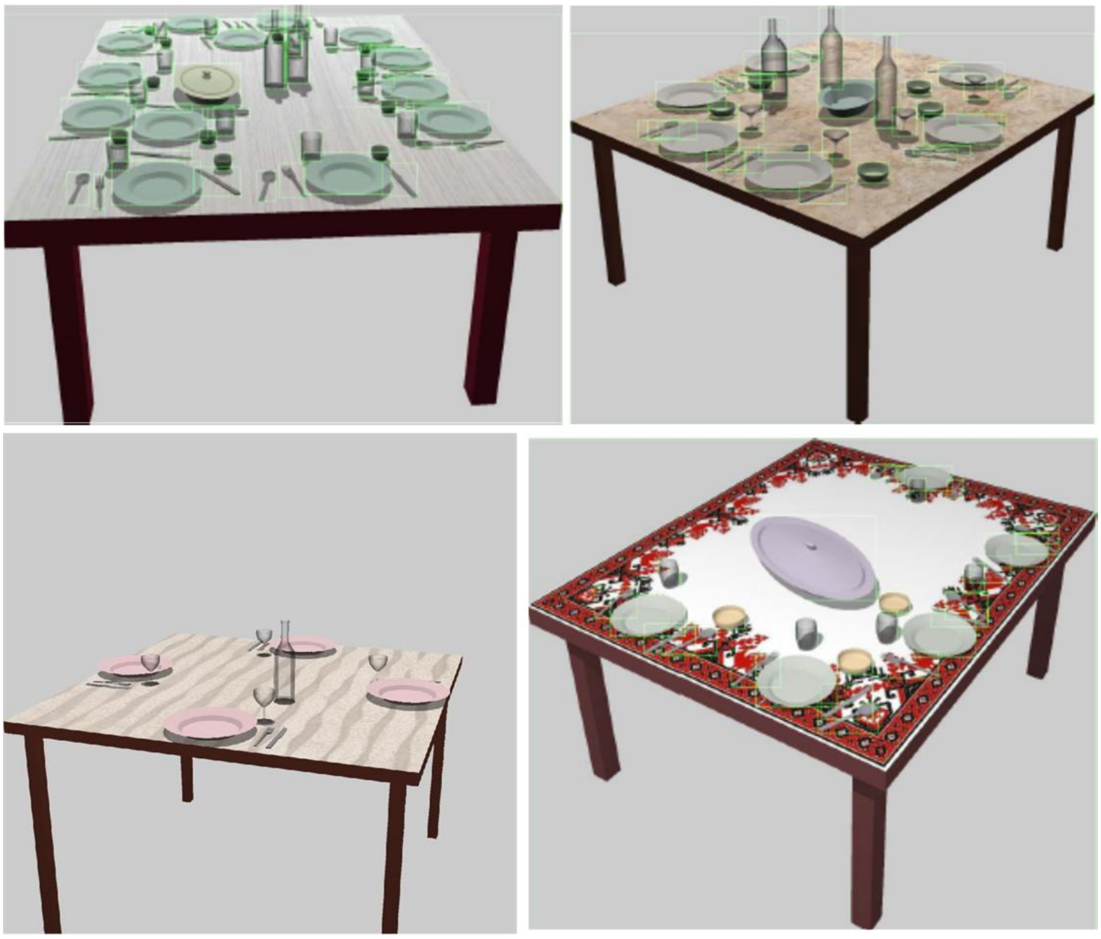


Figure 3.9: Synthetic table-setting scene samples.

Figure 3.10 (right) shows top-view visualization of the annotated scene in the left using top-view icons of the corresponding object instances for plates, glasses, and utensils (note that all utensil instances are shown by top-view knife icons).

In the next section, we propose a method to learn model parameters from annotated scenes.

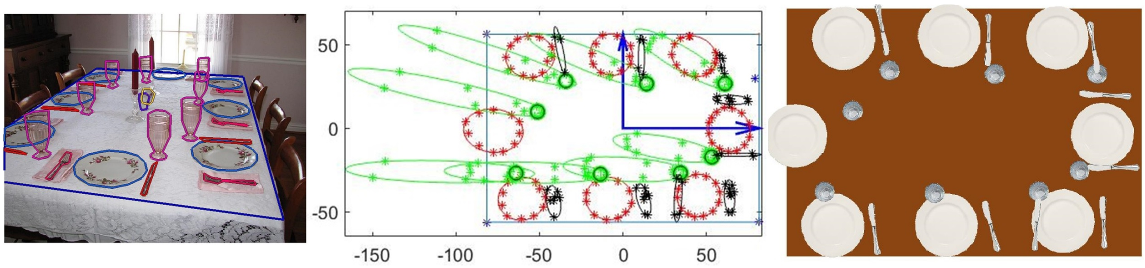


Figure 3.10: **Left**: an annotated image from the table-setting dataset. **Middle**: back-projection of table (in blue), plates (in red), glasses (in green), and utensils (in black). The unit of axes is centimeter. **Right**: top-view visualization of the table-setting (all utensil instances including fork, knife, and spoon are shown by knife icon).

3.3 Model Learning

We learn $p(g|T)$ from a subset of annotated images in the dataset whose table geometry roughly matches T in size. Note that since the circumference of the table surface is also annotated for every image we can compute the table geometry after back-projection. In an annotated image we do not directly observe the pose of objects in the world coordinate system but instead we observe (c_V, ξ_V). However, since we have manually estimated the homography matrix for each image in the dataset we can approximate (c_V, θ_V) per image.

Assume we have a dataset of J annotated scenes from which we obtain object attributes, namely we can get the category and 3D pose of each object instance *i.e.,* $\mathcal{D} = \{c_V[j], \theta_V[j]\}_{j=1}^{J}$. However, $\mathcal{D}$ is not a sufficient statistic for learning the model parameters since the set of corresponding base graphs $\mathcal{M} = \{g_0[j]\}_{j=1}^{J}$ is not observable. Therefore, we are facing a learning-from-incomplete-data problem where $\mathcal{D}$ is given and $\mathcal{M}$ is missing. The combination of missing and incomplete data constitutes the complete data composed of attributed graphs for each scene $\mathcal{D}^{+} = \langle \mathcal{D}, \mathcal{M} \rangle = \{g[j] = (g_0[j], c_V[j], \theta_V[j])\}_{j=1}^{J}$.

We propose a parameter learning method based on the Monte-Carlo version of the *Stochastic Expectation-Maximization* (SEM) algorithm [11, 17], usually referred to by MCEM in the literature [100]. According to the proposed learning strategy, we sample the conditional base graph distribution given the object attributes using Gibbs sampling to complete $\mathcal{D}$ (data imputation) and estimate the parameters by iteratively maximizing the complete data likelihood over the parameters. Let Φ denote the set of all the parameters in the model including the parameters of the four sets of distributions summarized in section 3.1 and $l(\Phi : \langle \mathcal{D}, \mathcal{M} \rangle)$ denote the log-likelihood of complete data with respect to Φ namely:

$$l(\Phi : \langle \mathcal{D}, \mathcal{M} \rangle) = \sum_{j=1}^{J} \log p(g[j] \mid \Phi). \tag{3.11}$$

Assuming that the base graphs for different images are independent given their corresponding object attributes we have:

$$p(\mathcal{M} \mid \mathcal{D}, \Phi^t) = \prod_{j=1}^{J} p(g_0[j] \mid c_V[j], \theta_V[j], \Phi^t). \tag{3.12}$$

We iteratively estimate the parameters until convergence according to:

$$\begin{aligned} \Phi^{t+1} &= \underset{\Phi}{\operatorname{argmax}} \ \ \boldsymbol{E}_{p(\mathcal{M} \mid \mathcal{D}, \Phi^t)}[l(\Phi : \langle \mathcal{D}, \mathcal{M} \rangle)] \\ &= \underset{\Phi}{\operatorname{argmax}} \ \ \sum_{j=1}^{J} \sum_{g_0[j]} p(g_0[j] \mid c_V[j], \theta_V[j], \Phi^t) \times \log p(g[j] \mid \Phi), \end{aligned} \tag{3.13}$$

where, the joint distribution $p(g[j] \mid \Phi)$ is based on (3.8) for parameters Φ. Note that we have dropped T in this section for notational convenience. Let $g_0^{(l)}[j]$ denote a base graph sample for the j-th annotated scene taken from $p(g_0[j] \mid c_V[j], \theta_V[j], \Phi^t)$. If N is a sufficiently large number of such samples, then using Monte-Carlo integration we have:

$$\Phi^{t+1} \approx \underset{\Phi}{\operatorname{argmax}} \sum_{j=1}^{J} \sum_{l=1}^{N} \log p(g^{(l)}[j] = (g_0^{(l)}[j], c_V[j], \theta_V[j]) \mid \Phi). \tag{3.14}$$

We now describe how to sample from the conditional base graph distribution. Since, for an annotated scene, (c_V, θ_V) is given:

$$p(g_0[j] \mid c_V[j], \theta_V[j], \Phi^t) = \frac{1}{Z(c_V[j], \theta_V[j], \Phi^t)} \times p(g_0[j], c_V[j], \theta_V[j] \mid \Phi^t), \tag{3.15}$$

where Z is a normalizing factor. The unnormalized probability in the right-hand-side term in (3.15) is the attributed graph model distribution and is easy to evaluate. We avoid computing the normalizing factor in every iteration by Gibbs sampling [29] the conditional base graph distribution as follows. We drop image index j for notational convenience hereafter. For a scene with $|V|$ annotated objects we consider a $|V|$-dimensional variable vector $\mathbf{z} = \big(z_1 = pa(v_1), z_2 = pa(v_2), ..., z_{|V|} = pa(v_{|V|})\big)$ that encodes any possible base graph g_0 for the scene in a one-to-one correspondence. Let $f(.)$ denote this one-to-one transformation *i.e.,* $g_0 = f(\mathbf{z})$. Gibbs sampling the conditional base graph distribution is performed according to the following steps:

Step-1. Begin with initial configuration $\mathbf{z}^{(0)} = (z_1^{(0)} = \emptyset, ..., z_{|V|}^{(0)} = \emptyset)$, $i \leftarrow 1$, and $l \leftarrow 1$.

Step-2. Sweep the $|V|$-dimensional variable vector $\mathbf{z}$ by sampling one element at the time conditioned on all the other most recent variables of $\mathbf{z}$ given the annotated scene attributes (c_V, θ_V). The i-th element (corresponding to the i-th vertex) is sampled in the l-th sample drawing according to the following probabilities

$$p(z_i) = \frac{p(g_0^{(l)}(z_i), c_V, \theta_V)}{\sum_{z_i \in S_{pa(v_i)}} p(g_0^{(l)}(z_i), c_V, \theta_V)}, \; z_i \in S_{pa(v_i)},$$

where $g_0^{(l)}(z_i) = f(z_1^{(l)}, ..., z_{i-1}^{(l)}, z_i, z_{i+1}^{(l-1)}, ..., z_{|V|}^{(l-1)})$ and $S_{pa(v_i)}$ denotes the set of permissible parents for v_i. Let $z_i^{(l)}$ denote the sampled i-th element.

Step-3. Generate the corresponding base graph sample $g_0^{(l)} = f(\mathbf{z}^{(l)})$, where:

$$\mathbf{z}^{(l)} = (z_1^{(l)}, ..., z_{i-1}^{(l)}, z_i^{(l)}, z_{i+1}^{(l-1)}, ..., z_{|V|}^{(l-1)}).$$

Set $l \leftarrow l + 1$ and $i \leftarrow (l \mod |V|)$ and go to Step-2 to draw the next sample.

The parent set $S_{pa(v_i)}$ is:

$$S_{pa(v_i)} = \emptyset \cup \{v_k \in V,\; k \text{ s.t. } d(\theta_{v_k}, \theta_{v_i}) \leq d_{(c_{v_k}, c_{v_i})} \text{ and } (c_{v_k} \to c_{v_i}) \in \mathcal{E}_M\},$$

which is the union of the empty set and a set of objects from permissible categories (according to the master graph) that are located within the threshold radius $d_{(c_{v_k}, c_{v_i})}$. In the case that there is no permissible parent for an instance *i.e.,* $S_{pa(v_i)} = \emptyset$, we will skip sampling the corresponding element in $\mathbf{z}$ *i.e.,* z_i, since we always have $z_i = \emptyset$.

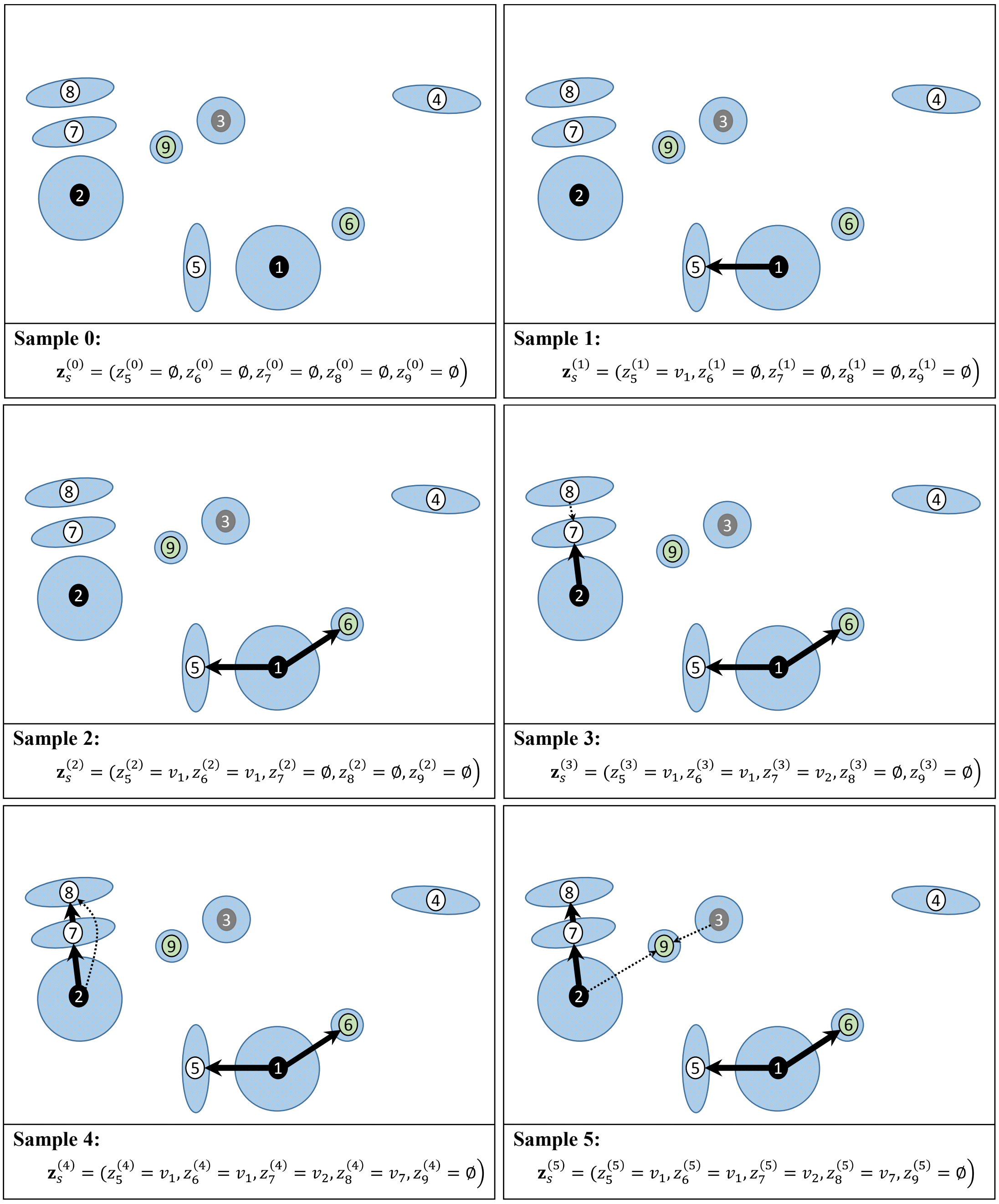
Let $\mathbf{z}_s$ denote a vector including elements of $\mathbf{z}$ whose parent space is not empty *i.e.,* $\mathbf{z}_s = \{z_i, \text{ s.t. } S_{pa(v_i)} \neq \emptyset\}$. Figure 3.11 shows a full sweep Gibbs sampling of $\mathbf{z}_s$ for an example annotated scene. In Figure 3.11 the blue ellipses represent the pose of objects and the colored circles located at the centers of these ellipses indicate the corresponding vertices in the attributed graph where different colors indicate different object categories. Let's assume that the sets of permissible parents for vertices in the graph are as below:

$$
\begin{aligned}
&S_{pa(v_1)} = S_{pa(v_2)} = S_{pa(v_3)} = S_{pa(v_4)} = \emptyset, \\
&S_{pa(v_5)} = S_{pa(v_6)} = \{\emptyset, v_1\}, \\
&S_{pa(v_7)} = \{\emptyset, v_2, v_8\}, \\
&S_{pa(v_8)} = \{\emptyset, v_2, v_7\}, \\
&S_{pa(v_9)} = \{\emptyset, v_2, v_3\}.
\end{aligned}
$$

Hence, $\mathbf{z}_s = (z_5, z_6, z_7, z_8, z_9)$. The Gibbs sampling starts with an initial configuration in "Sample 0" where all object instances are orphan (top left figure). In the next sample, the fifth instance is adopted by the first instance. In "Sample 2" the sixth instance is also adopted by the first instance. The seventh instance in "Sample 3" could be adopted by either one of v_2 or v_8, or remain orphan, however it got randomly adopted by the second instance; the dashed narrower line between v_8 and v_7 indicates that this relationship could be selected but was not selected. The next samples are taken accordingly as shown in Figure 3.11. Note that the solid thicker lines represent the selected parent-child relations whereas the dashed narrower lines represent the possible parent-child relations that were not selected during Gibbs sampling.

Using the above Gibbs sampling approach we are able to sample the conditional base graph distribution for every annotated scene. Assuming N base graph samples per image in each iteration we will have $N \times J$ complete samples from which we estimate the model parameters for the next iteration. We continue the iterations until convergence of parameters. The parameters of various distributions are learned from annotated table-setting images whose area is within 0.5 square meters difference. For example, the learned average rate of plate Poisson distribution is 3.8 which means on average we expect to observe 3.8 plates per square meter of table area. Figure 3.12 shows the SEM learning trajectory of two concentration parameters κ from the von Mises distributions on utensil orientation, where the blue line is for a utensil orientation whose parent is a plate instance and the black like is for a utensil orientation whose parent is another utensil instance. The SEM algorithm converged in about 10 iterations.

Figure 3.13 shows top-view visualization of some annotated images in the dataset that roughly match in size to a $1.5 \times 1.5 m^2$ table that are generated similar to Figure 3.10 (from left to right) and some samples drawn from the generative attributed graph model for a square table of size $1.5 \times 1.5 m^2$ learned from matching annotated images. Visual similarity of the samples taken from the generative attributed graph model to natural scene samples confirm suitability of this model for table setting scenes although the proposed model is quite general and can be used to model different types of scenes. A projected 2D



Sample 0:

$\mathbf{z}_s^{(0)} = \left(z_5^{(0)} = \emptyset, z_6^{(0)} = \emptyset, z_7^{(0)} = \emptyset, z_8^{(0)} = \emptyset, z_9^{(0)} = \emptyset\right)$

Sample 1:

$\mathbf{z}_s^{(1)} = \left(z_5^{(1)} = v_1, z_6^{(1)} = \emptyset, z_7^{(1)} = \emptyset, z_8^{(1)} = \emptyset, z_9^{(1)} = \emptyset\right)$

Sample 2:

$\mathbf{z}_s^{(2)} = \left(z_5^{(2)} = v_1, z_6^{(2)} = v_1, z_7^{(2)} = \emptyset, z_8^{(2)} = \emptyset, z_9^{(2)} = \emptyset\right)$

Sample 3:

$\mathbf{z}_s^{(3)} = \left(z_5^{(3)} = v_1, z_6^{(3)} = v_1, z_7^{(3)} = v_2, z_8^{(3)} = \emptyset, z_9^{(3)} = \emptyset\right)$

Sample 4:

$\mathbf{z}_s^{(4)} = \left(z_5^{(4)} = v_1, z_6^{(4)} = v_1, z_7^{(4)} = v_2, z_8^{(4)} = v_7, z_9^{(4)} = \emptyset\right)$

Sample 5:

$\mathbf{z}_s^{(5)} = \left(z_5^{(5)} = v_1, z_6^{(5)} = v_1, z_7^{(5)} = v_2, z_8^{(5)} = v_7, z_9^{(5)} = \emptyset\right)$

Figure 3.11: Example base graph samples for a synthetic annotated scene.

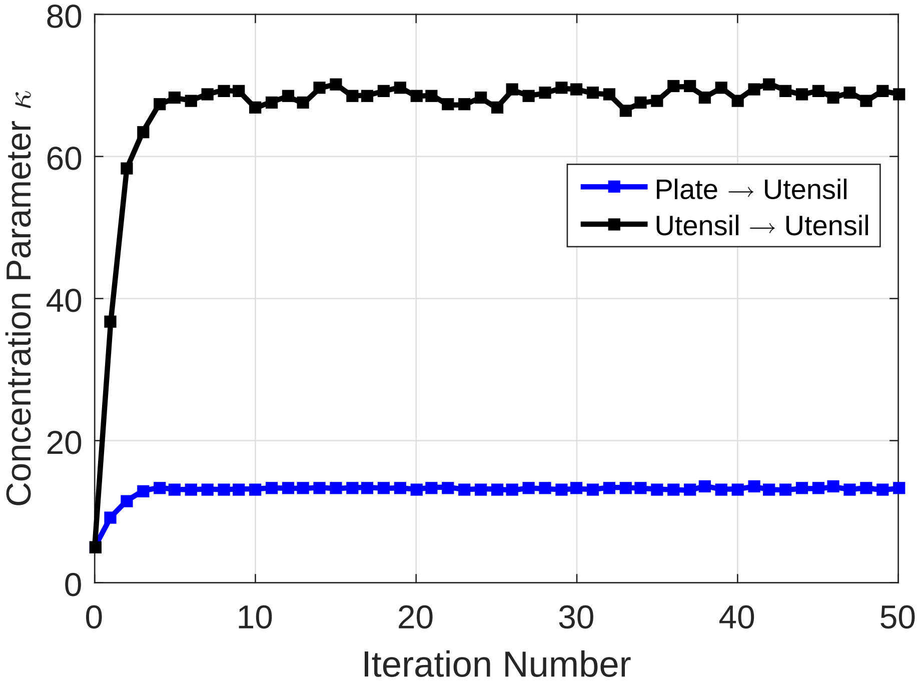


Figure 3.12: The trajectory of two concentration parameters from the von Mises distributions on utensil orientation learned iteratively using the SEM algorithm. The blue line is for a utensil orientation whose parent is a plate instance and the black like is for a utensil orientation whose parent is another utensil; the mean of these two von Mises distributions are, respectively, along a line perpendicular to the line connecting center of the parent plate instance to the center of the child utensil, and the line along the orientation of the parent utensil instance.

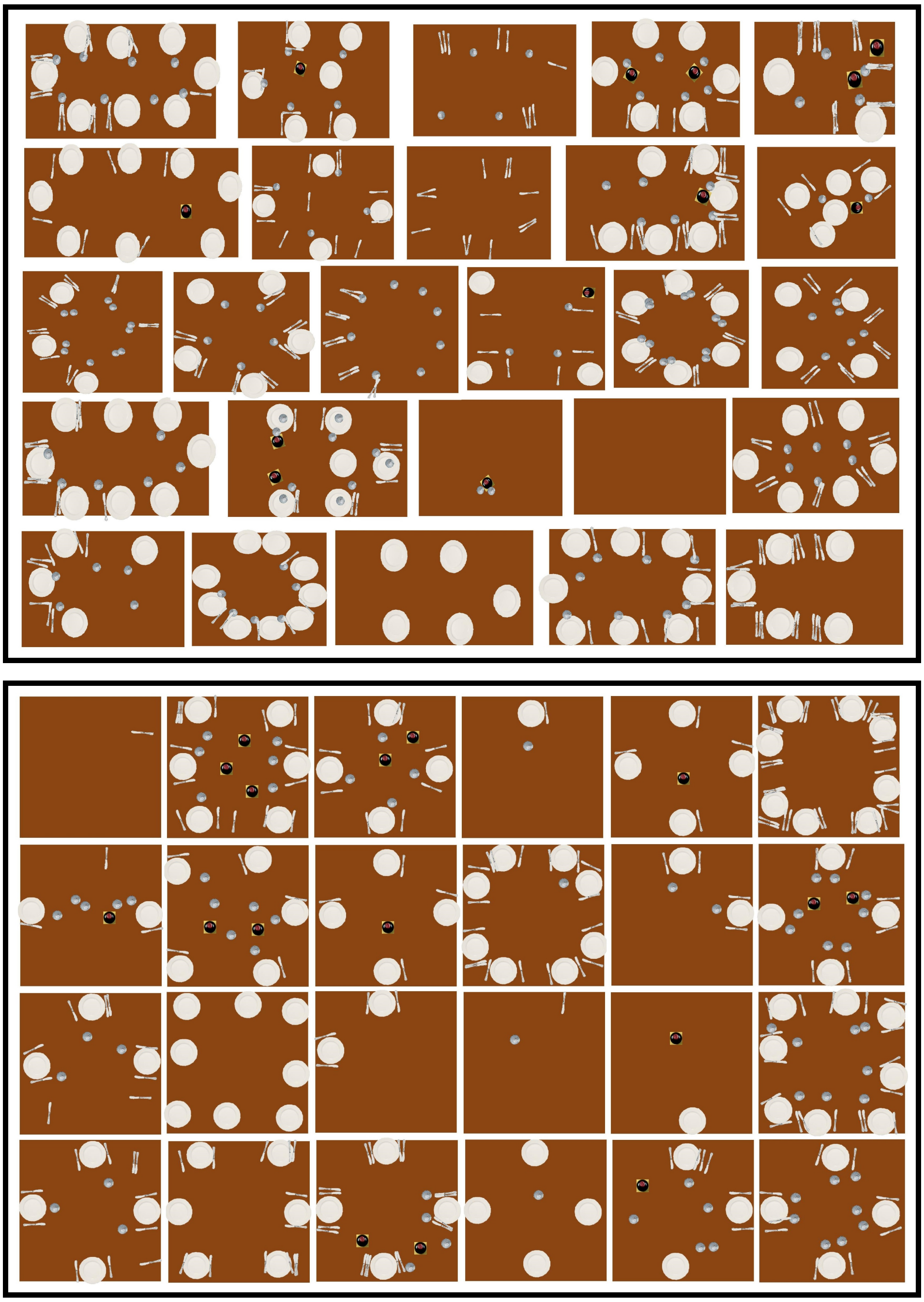


Figure 3.13: Top-view icon visualization of table-settings considering only plate, bottle, glass, utensil categories. **Top box**: visualization of some annotated images in the dataset that roughly match in size to a $1.5 \times 1.5\,m^2$ table (generated similar to figure 3.10 from left to right). **Bottom box**: samples from the generative attributed graph model for a square table of size $1.5 \times 1.5\,m^2$.

model in the image coordinate system similar to (3.2) is also applicable to different scene types.

This completes our proposed parameter learning method. We propose methods for unconditional and conditional sampling of the prior model in the next section.

3.4 Sampling the Generative Attributed Model

We discuss both unconditional and conditional sampling of the prior distribution. We start with the unconditional sampling of the attributed graph distribution $p(g|T)$ in a top-down fashion and then conditional sampling of $p(g|T, X)$ where X denotes an event based on g. The top-down unconditional samples taken from $p(g|T)$ will be used to generate statistics required to learn the Markov Random Field (MRF) Model in the next chapter. Even though direct conditional sampling from $p(g|T, X)$ is not a necessary building block in the process of our application to scene understanding it is still presented in this chapter since conditional sampling from a distribution on graphs with random structure is an interesting yet difficult problem and it could be applied to other problems *e.g.,* a model-based Visual Turing Test [26]. We drop table geometry T in the rest of this chapter for notational convenience.

3.4.1 Unconditional Sampling

To sample $p(g)$ we first sample the category-labeled base graph (g_0, c_V). There is an underlying hierarchical process to generate category-labeled base graphs for both Bayesian and hybrid graphs that can be summarized as below:

1. First sample the number and types of vertices in V_0 by generating a sample from $p^{(0)}$. The size of V_0 is $|V_0| = \sum_c n_{(0,c)}$. Note that V_i denotes the set of vertices in the i-th level of g_0.

2. For $i = 1$ to l_{max}:

   (a) Visit each vertex $v \in V_{i-1}$ in *any order* and *independently* generate a sample from $p^{(c_v)}$. This generates a set of category-labeled vertices in the next level V_i. The size of the next set is $|V_i| = \sum_c \sum_{v \in V_{i-1}} n_{(v,c)}$.

   (b) Draw a directed edge from node $v \in V_{i-1}$ to each of its children $v' \in ch(v)$. In addition, connect all of the children via undirected edges to build a hybrid model as illustrated in figure 3.1(bottom). In the special case of the Bayesian forest model we skip connecting children by undirected edges.

Given a category-labeled base graph (g_0, c_V) we sample the pose of roots and children instances according to the corresponding pose distributions in a top-down fashion leading to a complete prior sample (g_0, c_V, θ_V).

3.4.2 Conditional Sampling

One way of conditional sampling is to first sample the model in a top-down fashion (as was explained in the previous subsection) and accept the sample if it satisfies the condition X and reject it otherwise. However, this approach will result in too many rejections if the condition X is strong (not easily satisfied) and consequently causing long periods of freeze-ups. We propose an alternative sampling method. Obviously, conditional sampling will not be as straightforward as the top-down unconditional sampling. According to the Bayes' rule we can write:

$$p(g|X) = \frac{p(g)p(X|g)}{p(X)} \propto p(g)p(X|g). \tag{3.16}$$

Based on (3.16), given X the posterior distribution is proportional to the prior distribution times the probability of X given the attributed graph g. The inverse conditional term $p(X|g)$ takes binary values if X is a deterministic function of g. In this case, $p(X|g) = 1$ if $X = f_X(g)$ and $p(X|g) = 0$ if $X \neq f_X(g)$; in another words, the inverse conditional term is equal to one if g and X are consistent and zero otherwise. The inverse conditional term can take any value between zero and one if there is a stochastic relationship between g and X *e.g.,* if X denoted the output of some classifier then $p(X|g)$ can be estimated from the error statistics of the classifier. In the rest of this subsection we will assume that the value of $p(X|g)$ is known for all $g \in \mathcal{G}$.

We propose a generic method based on the Metropolis-Hastings algorithm [36, 61] to sample the posterior distribution $p(g|X)$ which works as follows. Let $g^{(t)}$ denote the current configuration of the attributed graph at step t of the sampling algorithm. The next configuration is generated in two steps: first propose a change $g^{(t)}_{try}$, according to some transition probability $Q(g^{(t)}, g^{(t)}_{try})$ and accept it (setting $g^{(t+1)} = g^{(t)}_{try}$) with probability:

$$\pi^{(t)} = \min\left(1, \frac{p(g^{(t)}_{try}|X)Q(g^{(t)}_{try}, g^{(t)})}{p(g^{(t)}|X)Q(g^{(t)}, g^{(t)}_{try})}\right), \tag{3.17}$$

and set $g^{(t+1)} = g^{(t)}$ otherwise. Note that we choose the initial configuration $g^{(0)}$ to be consistent with X. Using (3.16) we get:

$$\pi^{(t)} = \min\left(1, \frac{p(g^{(t)}_{try})p(X|g^{(t)}_{try})Q(g^{(t)}_{try}, g^{(t)})}{p(g^{(t)})p(X|g^{(t)})Q(g^{(t)}, g^{(t)}_{try})}\right), \tag{3.18}$$

where the term $p(X)$ is cancelled out from the numerator by the same term in the denominator. Note that, for the transition to be well defined, we need to ensure that Q satisfies the weak symmetry condition:

$$Q(g, g') > 0 \Leftrightarrow Q(g', g) > 0.$$

Because large changes $g^{(t)} \rightarrow g^{(t)}_{try}$ typically result in very small values for the acceptance probability $\pi^{(t)}$ leading to long periods of freeze time, one typically designs Q so that it makes small variations on the current configuration. An efficient proposal $Q(.,.)$ depends on X such that the suggested configurations are completely or at least partially consistent with X in order to avoid a high number of proposal rejections. Hence, we cannot offer a generic recipe to design a proposal distribution that is efficient for every X.

We review a series of possible elementary changes and their associated transition probabilities for conditional sampling. These moves may be chosen at random or based on another proposal distribution.

***Birth and Death*:** Removing a node is only allowed if this node has no child $v \in V_{\text{T}}$. Let g denote the current configuration. A transition can be formulated as below:

Death-1. Pick a node v at random among the childless ones $v \in V_{\text{T}}$.

Death-2. Form the new configuration g' by deleting node v from g and also the directed edges issued from its parent and any undirected edges connected to v if applicable.

This defines the following transition probability:

$$Q_{\text{death}}(g, g') = \frac{1}{|V_{\text{T}}(g)|},$$

where $|V_{\text{T}}(g)|$ is the number of terminal nodes in g.

To compute the acceptance probability for this move, we first need to define an inverse move which allows a transition from g' to g. The corresponding *birth* move can work as follows, letting $\tilde{g}$ denote the current configuration.

Birth-1. Randomly select a category $\tilde{c} \in \mathcal{C}$.

Birth-2. Add node $\tilde{v}$ from category $\tilde{c}$. We may add $\tilde{v}$ as either a root node or the child of another node that can accept one more child from category $\tilde{c}$. In the later case, let $pa(\tilde{v})$ denote the parent node that is chosen to adopt $\tilde{v}$.

Birth-3. Sample a pose $\tilde{\theta}_{\tilde{v}}$ for the new object according to the relevant pose distribution (conditioned on parent or not).

Birth-4. Add this object, required edges and pose to $\tilde{g}$, yielding a new configuration $\tilde{g}'$.

The corresponding transition probability therefore is:

$$Q_{\text{birth}}(\tilde{g}, \tilde{g}') = \frac{1}{|\mathcal{C}|\big(1 + n_{\tilde{c}-\text{open}}(\tilde{g})\big)} p(\tilde{\theta}_{\tilde{v}}|\tilde{c}, c_{pa(\tilde{v})}, \tilde{\theta}_{pa(\tilde{v})})$$

where $\tilde{\theta}_{pa(\tilde{v})}$ is the pose of the selected parent (if $\tilde{v}$ is not a root node) and $n_{c-\text{open}}(\tilde{g})$ is the number of nodes than can adopt a child from category c at step Birth-2.

We can now define the transition ratio (for the transition $g \to g'$) as below:

$$\begin{aligned} \rho_{\text{death}}(g, g') &= \frac{p(g')p(X|g')Q_{\text{birth}}(g', g)}{p(g)p(X|g)Q_{\text{death}}(g, g')} \\ &= \frac{|V_{\text{T}}(g)|}{|\mathcal{C}|\big(1 + n_{c(v)-\text{open}}(g')\big)} \times \frac{p(g_0', c_{V'})p(X|g')}{p(g_0, c_V)p(X|g)}, \end{aligned}$$

where v is the deleted object. Note that $p(\theta_{V'}|g_0', c_{V'})/p(\theta_V|g_0, c_V)$ is equal to $1/p(\theta_v|c, c_{pa(v)}, \theta_{pa(v)})$ and cancels with the corresponding term in $Q_{\text{birth}}(g', g)$. The ratio $p(g_0', c_{V'})/p(g_0, c_V)$ is given by:

$$\frac{p(g_0', c_{V'})}{p(g_0, c_V)} = \frac{p^{(c_{pa(v)})}(n_{(pa(v),1)}, ..., n_{(pa(v),c_v)} - 1, ..., n_{(pa(v),K)})}{p^{(c_{pa(v)})}(n_{(pa(v),1)}, ..., n_{(pa(v),c_v)}, ..., n_{(pa(v),K)})},$$

if v is not a root node, and:

$$\frac{p(g_0', c_{V'})}{p(g_0, c_V)} = \frac{p^{(0)}(n_{(0,1)}, ..., n_{(0,c_v)} - 1, ..., n_{(0,K)})}{p^{(0)}(n_{(0,1)}, ..., n_{(0,c_v)}, ..., n_{(0,K)})},$$

if v is a root node. If we assume (3.9) and (3.10) are true then we have:

$$\frac{p(g_0', c_{V'})}{p(g_0, c_V)} = \frac{p_u^{(c_{pa(v)})}(n_{(pa(v),c_v)} - 1)}{p_u^{(c_{pa(v)})}(n_{(pa(v),c_v)})},$$

and

$$\frac{p(g_0', c_{V'})}{p(g_0, c_V)} = \frac{p_u^{(0)}(n_{(0,c_v)} - 1)}{p_u^{(0)}(n_{(0,c_v)})},$$

for v being a non-root and a root node, respectively. The ratio for the birth transition $\tilde{g} \to \tilde{g}'$ is given by:

$$\begin{aligned} \rho_{\text{birth}}(\tilde{g}, \tilde{g}') &= \frac{p(\tilde{g}')p(X|\tilde{g}')Q_{\text{death}}(\tilde{g}', \tilde{g})}{p(\tilde{g})p(X|\tilde{g})Q_{\text{birth}}(\tilde{g}, \tilde{g}')} \\ &= \frac{|\mathcal{C}|\big(1 + n_{\tilde{c}-open}(\tilde{g})\big)}{|V_{\text{T}}(\tilde{g}')|} \times \frac{p(\tilde{g}_0', c_{\tilde{V}'})p(X|\tilde{g}')}{p(\tilde{g}_0, c_{\tilde{V}})p(X|\tilde{g})}. \end{aligned}$$

For example, an efficient proposal distribution will not try the birth and death move from a particular category if the condition X deterministically enforces that a certain number of objects from that category have to exist in the scene and we already started with an initial attributed graph which had that many category instances.

***Edge Deletion/Addition*:** These transitions remove a parent-child link or create one. The deletion can be implemented with the following sequence of operations:

Del-1. Select a directed edge $(\text{pa}(v), v)$ from the current configuration g and delete it.

Del-2. Let g' denote the new configuration.

The transition probability is $Q_{\text{del}}(g, g') = 1/n_{\text{edge}(g)}$. The reverse operation can be done according to the following steps, starting with a configuration $\tilde{g}$.

Add-1. Select a parentless node $\tilde{v} \in \tilde{V}_0$ such that there exist nodes from permissible categories (according to the master graph) that can adopt $\tilde{v}$ as child. Let $S_{pa(\tilde{v})}$ denote the set of possible parents for $\tilde{v}$, and $n_{\text{add}(\tilde{g})}$ denote the number of parentless nodes like $\tilde{v}$ that may be adopted by another node.

Add-2. Select a node $\tilde{v}' \in S_{pa(\tilde{v})}$ and add the edge $\tilde{v}' \to \tilde{v}$ to $\tilde{g}$.

Add-3. Let $\tilde{g}'$ be the new configuration.

The transition probability is:

$$Q_{\text{add}}(\tilde{g}, \tilde{g}') = \frac{1}{|S_{pa(\tilde{v})}| n_{\text{add}(\tilde{g})}}.$$

The corresponding acceptance ratio for edge deletion is:

$$\begin{aligned}
\rho_{\text{del}}(g, g') &= \frac{p(g')p(X|g')Q_{\text{add}}(g', g)}{p(g)p(X|g)Q_{\text{del}}(g, g')} \\
&= \frac{p_u^{(0)}(n_{(0,c_v)}(g')|T)}{p_u^{(0)}(n_{(0,c_v)}(g)|T)} \frac{p_u^{(c_{pa(v)})}\big(n_{(pa(v),c_v)}(g')\big)}{p_u^{(c_{pa(v)})}\big(n_{(pa(v),c_v)}(g)\big)} \\
&\times \frac{p(\theta_v|c_v)}{p(\theta_v|c_v, c_{pa(v)}, \theta_{pa(v)})} \frac{p(X|g')}{p(X|g)} \frac{n_{\text{edge}(g)}}{|S_{pa(v)}(g')| n_{\text{add}(g')}},
\end{aligned}$$

for deletion of edge $pa(v) \to v$. Note that $n_{(0,c_v)}(g') = n_{(0,c_v)}(g) + 1$ and $n_{(pa(v),c_v)}(g') = n_{(pa(v),c_v)}(g) - 1$. The corresponding factor for edge addition is:

$$\begin{aligned}
\rho_{\text{add}}(\tilde{g}, \tilde{g}') &= \frac{p(\tilde{g}')p(X|\tilde{g}')Q_{\text{del}}(\tilde{g}', \tilde{g})}{p(\tilde{g})p(X|\tilde{g})Q_{\text{add}}(\tilde{g}, \tilde{g}')} \\
&= \frac{p_u^{(0)}(n_{(0,c_{\tilde{v}})}(\tilde{g}')|U)}{p_u^{(0)}(n_{(0,c_{\tilde{v}})}(\tilde{g})|U)} \frac{p_u^{(c_{pa(\tilde{v})})}\big(n_{(pa(\tilde{v}),c_{\tilde{v}})}(\tilde{g}')\big)}{p_u^{(c_{pa(\tilde{v})})}\big(n_{(pa(\tilde{v}),c_{\tilde{v}})}(\tilde{g})\big)} \\
&\times \frac{p(\theta_{\tilde{v}}|c_{\tilde{v}}, c_{pa(\tilde{v})}, \theta_{pa(\tilde{v})})}{p(\theta_{\tilde{v}}|c_{\tilde{v}})} \frac{p(X|\tilde{g}')}{p(X|\tilde{g})} \frac{|S_{pa(\tilde{v})}(\tilde{g})| n_{\text{add}(\tilde{g})}}{n_{\text{edge}(\tilde{g}')}},
\end{aligned}$$

for addition of edge $pa(\tilde{v}) \to \tilde{v}$. Note that in above $n_{(0,c_{\tilde{v}})}(\tilde{g}') = n_{(0,c_{\tilde{v}})}(\tilde{g}) - 1$ and $n_{(pa(\tilde{v}),c_{\tilde{v}})}(\tilde{g}') = n_{(pa(\tilde{v}),c_{\tilde{v}})}(\tilde{g}) + 1$.

***Pose Change*:** We make use of normal and von Mises distributions as proposal distributions for location and orientation of a randomly selected object instance in the following steps:

Pose-1. Select a node $v \in V$ at random and either change its location by sampling a bivariate normal distribution centered at the current location of the corresponding instance or change its orientation by sampling a von Mises distribution centered at the current orientation of the instance.

Pose-2. Update the attributed graph with the new pose and denote the new configuration by g'.

Since both normal and von Mises distributions are symmetric they will cancelled out from the numerator and denominator of the acceptance ratio to leave only the ratio of posteriors as in below:

$$\rho_{\text{pose}}(g, g') = \frac{p(g')p(X|g')}{p(g)p(X|g)} \tag{3.19}$$

$$\tag{3.20}$$

The ratio $p(g')/p(g)$ may be further simplified since changing the pose of an instance only affects the pose distribution of itself and its neighbors (connected via an edge).

3.5 Camera Parameters Prior

Assuming that the camera's extrinsic and intrinsic variables are independent, we have:

$$\begin{aligned} p(\mathcal{W}) &= p(\psi_x, \psi_y, \psi_z, t_x, t_y, t_z, f, s_x, s_y, \dot{x}_0, \dot{y}_0) \\ &= p(\psi_x, \psi_y, \psi_z, t_x, t_y, t_z) \times p(f, s_x, s_y, \dot{x}_0, \dot{y}_0) \\ &= p(\psi_x, \psi_y, \psi_z \mid t_x, t_y, t_z) \times p(t_x, t_y, t_z) \times p(f, s_x, s_y, \dot{x}_0, \dot{y}_0), \end{aligned} \tag{3.21}$$

where, extrinsic parameters include: $\boldsymbol{t} = (t_x, t_y, t_z)$ denoting camera's location in the world coordinate system, and ψ_x, ψ_y, and ψ_z denoting, respectively, counter-clockwise rotation of camera's coordinate system about the x-axis, y-axis, and z-axis of the table coordinate system[1]; and intrinsic parameters include: camera's focal length f, the size of pixels in the x and y directions denoted respectively by s_x and s_y, and $(\dot{x}_0, \dot{y}_0)$ denoting the point on the image plane (in pixels) where the camera's principal axis meets the image plane [35]. It makes sense to assume that the height of camera, t_z, is independent of its horizontal location, (t_x, t_y):

$$p(t_x, t_y, t_z) = p(t_x, t_y) \times p(t_z). \tag{3.22}$$

We choose $p(t_z)$ to be a shifted and scaled *Beta* distribution with camera height ranging from 0.3 to 3 meters (respect to the table surface) *i.e.,* $t_z \in [-3, -0.3]$ (see Figure 3.2). The

[1] The counter-clockwise rotation about an axis means the counter-clockwise rotation occurs when the axis points toward the observer.

horizontal location of the camera, (t_x, t_y), is represented in a polar coordinate system centered at the center of the world coordinate system (center of the table): $t_x = r \times \cos(\alpha)$ and $t_y = r \times \sin(\alpha)$. We assume the variables r and α are independent *i.e.,* $p(r, \alpha) = p(\alpha)\, p(r)$, where $p(\alpha) = \frac{1}{2\pi}$, $\alpha \in [0, 2\pi)$, and $p(r)$ is a scaled *Beta* distribution in the range $r \in [0, 4]$ meters. The distributions $p(t_x, t_y)$ and $p(r, \alpha)$ are related via the absolute determinant value of the Jacobian of the new variables, (r, α), with respect to the old variables, (t_x, t_y), as below:

$$p(t_x, t_y) = p(r, \alpha) |\text{det}(\frac{\partial(r, \alpha)}{\partial(t_x, t_y)})| = p(r, \alpha) \frac{1}{r}. \tag{3.23}$$

The conditional distribution of the rotation variables given camera's location is constructed assuming that on average the camera's z-axis, $\vec{u}_z$, points toward the table's center, namely $\mathbb{E}\{\vec{u}_z\} = -\frac{\boldsymbol{t}}{\|\boldsymbol{t}\|}$, where $\mathbb{E}\{.\}$ stands for expectation, and the camera's x-axis, $\vec{u}_x$, is parallel to the table plane, hence, we choose its average to be the cross product of the average normal to the camera's image plane, $\mathbb{E}\{\vec{u}_z\}$, and the table's normal, $\vec{u}_Z$, namely $\mathbb{E}\{\vec{u}_x\} = \mathbb{E}\{\vec{u}_z\} \times \vec{u}_Z$, and finally, we set $\mathbb{E}\{\vec{u}_y\} = \mathbb{E}\{\vec{u}_z\} \times \mathbb{E}\{\vec{u}_x\}$. We also assume that the rotation angles are conditionally independent given the camera's location:

$$p(\psi_x, \psi_y, \psi_z \mid t_x, t_y, t_z) = p(\psi_x \mid t_x, t_y, t_z) \times p(\psi_y \mid t_x, t_y, t_z) \times p(\psi_z \mid t_x, t_y, t_z), \tag{3.24}$$

where, each of the conditional distributions on the right-hand side of (3.24) is chosen to be a *von Mises* distribution:

$$p(\psi \mid \mu, \kappa) = \frac{\exp(\kappa \cos(\psi - \mu))}{2\pi I_0(\kappa)}, \tag{3.25}$$

where, μ represents the mean of distribution, κ_0 is a measure of dispersion, and $I_0(.)$ is the modified *Bessel* function of order 0. We calculate the mean values of the *von Mises* distributions, μ_x, μ_y, and μ_z, from the mean axes of the camera's coordinate system $(\mathbb{E}\{\vec{u}_x\}, \mathbb{E}\{\vec{u}_y\}, \mathbb{E}\{\vec{u}_z\})$ and the axes of the world coordinate system $(\vec{u}_X, \vec{u}_Y, \vec{u}_Z)$. Comparing (A.4) and (A.5), we obtain

$$\begin{aligned}
\mu_y &= \sin^{-1}(-\mathbb{E}\{\vec{u}_x\}.\vec{u}_Z), \\
\mu_x &= \measuredangle \left(\frac{\mathbb{E}\{\vec{u}_z\}.\vec{u}_Z}{\cos(\mu_y)}, \frac{\mathbb{E}\{\vec{u}_y\}.\vec{u}_Z}{\cos(\mu_y)} \right), \\
\mu_z &= \measuredangle \left(\frac{\mathbb{E}\{\vec{u}_x\}.\vec{u}_X}{\cos(\mu_y)}, \frac{\mathbb{E}\{\vec{u}_x\}.\vec{u}_Y}{\cos(\mu_y)} \right),
\end{aligned}$$

which specifies the conditional rotation distribution in (3.24).

We assume that the pixels are square (symmetric horizontally and vertically) and therefore $s_y = s_x$ and the rest of the intrinsic parameters are mutually independent:

$$p(f, s_x, s_y, \dot{x}_0, \dot{y}_0) = p(f) \times p(s_x, s_y) \times p(\dot{x}_0) \times p(\dot{y}_0). \tag{3.26}$$

We choose a uniform density for the focal length ranging from 10 to 40 millimeters:

$$p(f) = \frac{1}{0.03}, \;\; f \in [0.01, 0.04].$$

For a given test image I with width of W_p pixels and camera image plane width of W_m meters, we have $s_x = \frac{W_m}{W_p}$. Since for a given test image the size W_p is known, the only free parameter would be camera's image plane width W_m. We choose a uniform distribution for W_m ranging from 1 cm to 1.2 cm, hence:

$$p(s_x, s_y) = p(s_y|s_x) \times p(s_x), \tag{3.27}$$

where:

$$p(s_x) = \frac{W_p}{0.002}, \;\; s_x \in [\frac{0.010}{W_p}, \frac{0.012}{W_p}], \tag{3.28}$$

and since we assumed $s_y = s_x$, therefore $p(s_y|s_x)$ would simply be a deterministic function.

If the given test image I is not cropped, the point where the camera's principal axis $\vec{u}_z$ meets the image plane would be $(\dot{x}_0 = \frac{W_p}{2}, \dot{y}_0 = \frac{H_p}{2})$, for W_p and H_p respectively denoting the image's width and height in pixels. However, in order to not limit ourselves to non-cropped images we choose a uniform distribution for $\dot{x}_0$ and $\dot{y}_0$ centered at $\frac{W_p}{2}$ and $\frac{H_p}{2}$, respectively, as below:

$$p(\dot{x}_0) = \frac{2}{W_p}, \;\; \dot{x}_0 \in [\frac{W_p}{4}, \frac{3W_p}{4}], \tag{3.29}$$

and:

$$p(\dot{y}_0) = \frac{2}{H_p}, \;\; \dot{y}_0 \in [\frac{H_p}{4}, \frac{3H_p}{4}]. \tag{3.30}$$

We set all of the parameters of $p(\mathcal{W})$ manually and calibrate them such that the synthetic scene samples generated by the synthetic image renderer look reasonable.

3.6 Table Geometry Prior

We make the simplifying assumption that the table is rectangular with length T_L and width T_W meters *i.e.,* $T = (T_L, T_W)$, and the length and width of table are independently distributed:

$$p(T) = p(T_L, T_W) = p(T_L) \times p(T_W), \tag{3.31}$$

The length and width of table are identically distributed according to a shifted and scaled *Beta* distribution ranging from 0.5 to 3 meters.

3.7 Summary

In this chapter, we proposed a generative stochastic model for 3D scenes consisting of configurations of objects from a fixed library on a planar surface. Modeling is at the level of objects in that the variables account for the numbers and poses of object instances, and encodes preferred combinatorial and geometric configurations. These configurations are represented as attributed graphs with a flexible random skeleton that still allows for considerable variation in patterns among scene entities. A generative 3D scene model in the world coordinate system whose samples mimic natural 3D scenes can be applied to scene understanding, robotics and synthetic content creation. The model is easily extended to 3D layouts with multiple support planes, which covers a wide range of human-created scenes, *e.g.,* consider the aerial image in Figure 3.14 where each building has a random number of associated parking lots, from 0 to 4 lots, and each lot has a random number of cars. The high-level attributed graph together with its design in the world coordinate system render the model parameters interpretable and consequently easy to verify and set by the user. We also proposed Monte-Carlo methods to sample from the model on attributed graphs with random structure in both unconditional and conditional settings.

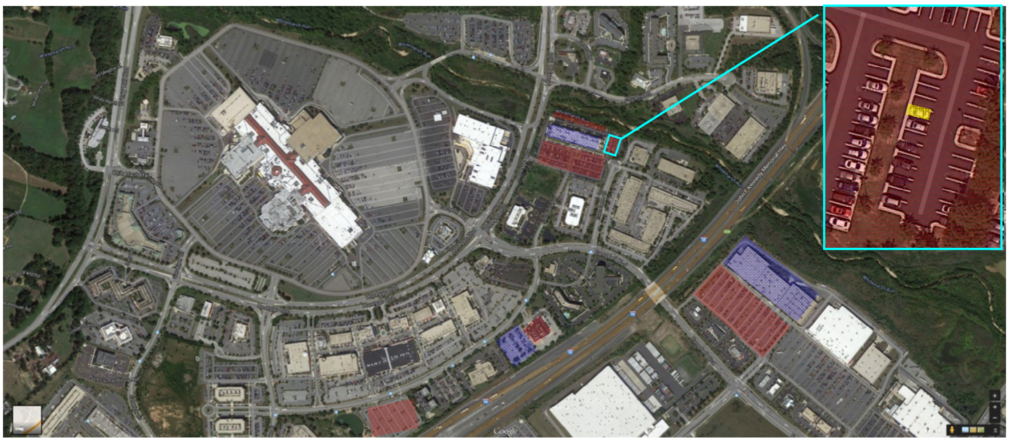


Figure 3.14: Google satellite view from the White Marsh mall in Baltimore county, Maryland.

Chapter 4

Markov Random Field Model

In chapter 3 we proposed a prior model based on the "Generative Attributed Graph" (GAG). We also proposed a posterior sampling algorithm based on Metropolis-Hastings [36, 61]. However, the Metropolis-Hastings algorithm has low acceptance rate (normally $<$ 25%) [3, 76] and depends heavily on the starting graph configuration and its consistency with the sampling condition. To increase the acceptance rate, smaller and more conservative changes need to be proposed by the proposal distribution which leads to sluggish convergence behavior. Starting with a graph configuration that does not comply with the sampling condition can further increase the burn-in period by generating a long sequence of inconsistent samples at the beginning. Hence, the previously proposed conditional sampling can result in a lengthy and computationally expensive inference. This issue is even more pronounced because the inference has to be done online at each step of the entropy pursuit algorithm. To circumvent this lengthy inference we model table-settings using a set of Gibbs distributions for various table geometries T to replace the GAG model (replacing $p(g|T)$ in (3.2) with a Gibbs distribution). However, the GAG model is a necessary component and cannot be removed from the story since we will learn the Gibbs distribution parameters from empirical statistics of the GAG model samples. The Gibbs model parameters could be learned directly from the dataset's annotated samples if we had sufficiently larger number of samples for each given T, but this is not the case. On the other hand, we can efficiently learn the GAG model from limited dataset's annotated samples roughly matching in table geometry T. Hence, since we can generate as many samples as desired from the GAG model for any given T, we first learn the GAG model (with orders of magnitude fewer number of parameters compared to the Gibbs model) from the dataset's annotated images and then learn the Gibbs model from the GAG model samples. According to the "Bias-Variance Tradeoff (Dilemma)" [28] complicated models with many parameters require more training samples if they are to avoid high variance. The GAG model has relatively few parameters because it is a Bayesian network and hence more modular. We learn the Gibbs distribution for every T from statistics of the GAG model samples drawn given the same T.

The random variables of the Gibbs distribution are binary and serve as presence indicators of the center of object instances from different categories in relatively small cells (5cm $\times$ 5cm) on the table and in the world coordinate system. The conditional inference is carried out using Gibbs sampling [29]. This leads to considerably faster inference firstly because Gibbs sampling has 100% acceptance rate and secondly because of the table area discretization result in faster exploration of the support space. For convenience we hereafter drop the table geometry T from notation. A Gibbs distribution is from the exponential family and is of the following form:

$$p_\lambda(\omega) = \frac{1}{Z(\lambda)} \exp\bigl(\lambda^T.\mathbf{f}(\omega)\bigr), \tag{4.1}$$

where ω denotes the model variables, $\mathbf{f}(\omega) = [f_1(\omega), f_2(\omega), \cdots, f_M(\omega)]^\top$ denotes the collection of features or sufficient statistics, $\lambda = [\lambda_1, \lambda_2, \cdots, \lambda_M]^\top$ are the corresponding feature weights or model parameters, and $Z(\lambda) = \sum_\omega \exp\bigl(\lambda^\top.\mathbf{f}(\omega)\bigr)$ is the normalizing factor (partition function) ensuring that the probabilities sum up to one. Figure 4.1 shows a table and its fitted mesh where each of the cells is a 5cm × 5cm square called "table cell". There are $|\mathcal{C}|$ binary variables in ω associated with each table cell which are presence indicators of at least one instance from the corresponding category centered anywhere in the 5cm × 5cm table cell. Hence, the number of model variables depends on the table size, the number of object categories of interest, and the size of table cells. We choose binary feature functions $\{f_i(\omega) \in \{0,1\},\ 1 \le i \le M\}$ that can be of either *singleton* or *conjunction* type. Singleton features depend only on one object category whereas the conjunction features depend on two different object categories. The singleton feature functions are incorporated to preserve the overall empirical statistics on the existence of an object category at a particular location on the table whereas the conjunction feature functions are aimed to incorporate the contextual relations between different object categories.

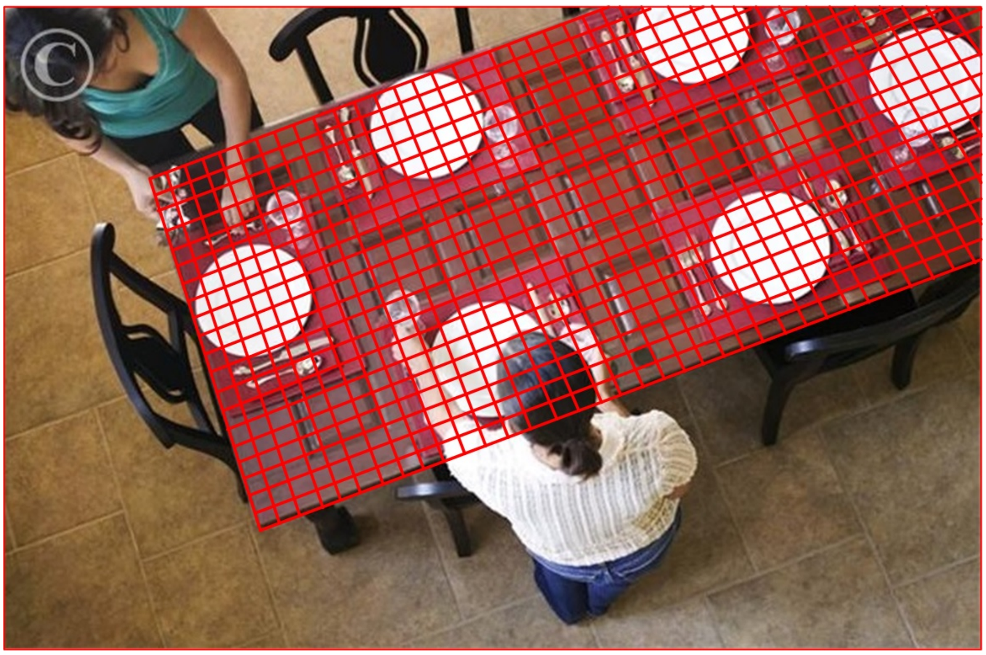


Figure 4.1: Table fitting mesh.

We include singleton feature functions at three different granularity levels. Figure 4.2 shows the domain of these singleton feature functions at the fine-level (5cm×5cm), middle-level (15cm × 15cm), and coarse-level (45cm × 45cm) by patches of respectively gray, green, and blue color on a fitting mesh. If the fine-level features were indexed 1 to M then $f_i(\omega) = \omega_i, 1 \leq i \leq M$. The middle-level and coarse-level features are equal to the maximum of their domain variables; obviously, an object instance centered in a given cell is also centered in any coarser cell enclosing it. Again, the corresponding singleton feature functions are basically presence indicators of an object instance from the corresponding category centered anywhere in their domain. The table cells at all levels are non-overlapping and they collectively cover the area of the table completely.

In addition to the singleton features at three resolution levels we also include singleton

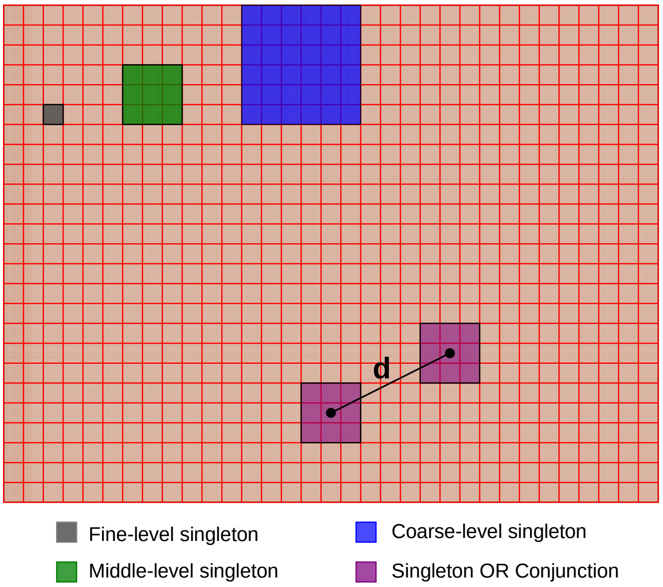


Figure 4.2: Domain of various types of feature functions.

feature functions whose domain is shown by the two purple boxes in Figure 4.2. Such feature functions are ON ($= 1$) if there are instances of the corresponding category in both boxes and OFF ($= 0$) otherwise. These singleton feature functions are intended to encode co-occurrence of instances from the same category. For example, these type of features prevent two plate instances to be very close to each other but they encourage local co-occurrence of two utensil instances. The conjunction feature functions have similar domain and definition as this type of singleton feature functions except that each of the purple patches shown in Figure 4.2 has to include an instance from a different category for the conjunction feature to be ON. Note that these type of features are defined on the middle-level patches ($15\text{cm} \times 15\text{cm}$). Since the number of conjunction features is of $O(n^2)$ in terms of the number of cells n, defining them on the fine-level cells would result in a high number of features which would require orders of magnitude higher numbers of samples to learn the model parameters. Two patches at middle-level constitute a conjunction feature function for c_1 and c_2 categories if the distance between centers of the two patches, denoted by d, is less than a threshold distance $d_{(c_1,c_2)}$; the threshold distance value depends on the involved categories and is larger for bigger object categories but for table-setting categories it is chosen to be always smaller than 45 cm. We have a set of conjunction feature functions for the pair of object categories c_1 and c_2 if there is an edge, no matter in what direction, between these two categories in the master graph.

4.1 Invariance Property

The number of parameters M can become very large depending on the table geometry and the number of considered categories. However, many of these parameters are expected to be equal due to the existing symmetry in table-settings, for instance, a plate is usually placed around the table at any side where someone is expected to sit with the same likelihood. The Gibbs model in (4.1) with M parameters corresponding to the feature functions described in the previous section is over-parameterized. Parameter estimation for an over-parameterized model can lead to poor parameter estimation as compared to a model with fewer number of parameters using the same number of training samples and computational resources. To incorporate invariance and improve estimation, we group features whose weights are expected to be the same and factor out these unique weights in the exponent of the Gibbs distribution in (4.1); we assume that the singleton feature functions whose domain is of the same size and located at the same distance from the closest table edge have the same weights and fall in the same group. In practice, we cluster singleton feature functions whose domain centers are at roughly the same distance from the closest table edge to the same group *i.e.,* quantization. We quantize coarser level feature functions to fewer numbers of groups. Our grouping strategy for the conjunction feature functions is quantizing the first patch based on its distance from the edge of table and quantizing the second patch based on its distance and angular location with respect to the first patch *i.e.,* sides (left, right, front, or back), and then clustering the similarly quantized feature functions to the same group. We assign the same weight to the feature functions from the same group and factor out these weights. This will result in a new set of feature functions which are the sum of the original feature functions from the same groups:

$$p_{\lambda}(\omega) = \frac{1}{Z(\lambda)} \exp\bigl(\lambda^T.\mathbf{f}(\omega)\bigr) = \frac{1}{Z(\lambda_{\mathrm{R}})} \exp\bigl(\lambda_{\mathrm{R}}^T.\mathbf{f}_{\mathrm{R}}(\omega)\bigr), \tag{4.2}$$

where, λ_{R} and $\mathbf{f}_{\mathrm{R}}$ are the reduced set of parameters and feature functions of size M_{R} ($M_{\mathrm{R}} \ll M$), respectively:

$$\lambda_{\mathrm{R}} = \bigl[\lambda'_1, \lambda'_2, \cdots, \lambda'_{M_{\mathrm{R}}}\bigr]^{\top}, \quad \mathbf{f}_{\mathrm{R}}(\omega) = \bigl[f'_1(\omega), f'_2(\omega), \cdots, f'_{M_{\mathrm{R}}}(\omega)\bigr]^{\top}.$$

Denoting the j-th index group by B_j we have:

$$\lambda'_j = \lambda_i, i \in B_j \text{ , and } \; f'_j(\omega) = \sum_{i \in B_j} f_i(\omega).$$

It can now be seen clearly that applying the invariance property basically reduces to grouping the symmetric feature functions and defining a new set of features which are the sums of the original feature functions. We will discuss conditional sampling from the Gibbs distribution in the next section.

4.2 Conditional Sampling

We propose a way to conditionally sample the Markov Random Field (MRF) model of table-settings (Gibbs distribution). The proposed sampler has three nested loops; the outer loop samples table geometry T, the loop in the middle samples camera parameters given table geometry from the outer loop, and the inner loop samples the Gibbs distribution given table geometry and camera parameters. The two outer loops use a Metropolis-Hastings sampling strategy and the inner loop sampling is based on Gibbs sampling. We start with the inner loop in the next subsection.

4.2.1 Conditional Sampling of the Gibbs Distribution

Assume T and $\mathcal{W}$ are given and let's drop T and $\mathcal{W}$ from our notations for convenience. We will use T and $\mathcal{W}$ explicitly whenever it makes our statements more clear. The posterior distribution can be written as below according to the Bayes' rule:

$$p_\lambda(\omega|\mathbf{e}_{k-1}) = \frac{p_\lambda(\omega)\, p(\mathbf{e}_{k-1}|\omega)}{p(\mathbf{e}_{k-1})} \propto p_\lambda(\omega)\, p(\mathbf{e}_{k-1}|\omega), \tag{4.3}$$

where, $\mathbf{e}_{k-1} = (\mathbf{q}_{k-1}, \mathbf{a}_{k-1})$ denotes the accumulated evidence up to step k of EP in either binary or composite representation, $\mathbf{q}_{k-1} = (q_1, \cdots, q_{k-1})$ is the list of queries, and $\mathbf{a}_{k-1} = (X_{q_1}, \cdots, X_{q_{k-1}})$ is the corresponding list of answers, namely classifiers' output. Assuming that classifiers' outputs are independent given the category and 2D pose of objects in the image coordinate system (c_V, ξ_V), we have:

$$p(\mathbf{e}_{k-1}|c_V, \xi_V) = \prod_{i=1}^{k-1} p(X_{q_i}|c_V, \xi_V), \tag{4.4}$$

where each of the terms in the right hand side of (4.4) is determined by our data model. We choose an appropriate representative for every binary random variable in ω that is ON ($= 1$). An appropriate representative is from the right category and at its typical size (in meters), centered anywhere in the corresponding table cell (placed randomly). These appropriate representatives specify the category and pose of objects in the world coordinate system (c_V, θ_V), and thereby (c_V, ξ_V) given the homography projection matrix H which is a deterministic function of $\mathcal{W}$. Figure 4.3 shows the functional relations between these variables. Hence, we have:

$$p(\mathbf{e}_{k-1}|\omega) = \prod_{i=1}^{k-1} p(X_{q_i}|c_V(\omega), \xi_V(\omega)). \tag{4.5}$$

where, the category labels are calculated based on the index of active (ON) variables in ω. Assuming N table cells, the first N elements of ω are associated with the first category, the second N elements of ω are associated with the second category, and so on. Note that in (4.5) we ignored the role of the uniform distributions we used to go from θ_V to ω; but this wont cause any inaccuracy during sampling since the uniform distributions will cancel out from the numerator and denominator of the acceptance ratios.

The set of interpretation units $\mathbf{Y}$ are determined given the 2D pose of objects in the image coordinate system and their categories *i.e.,* $\mathbf{Y}$ can be calculated given (c_V, ξ_V). If the interpretation units are sufficient statistic from (c_V, ξ_V), we will have:

$$p(\mathbf{e}_{k-1}|\omega) = \prod_{i=1}^{k-1} p(X_{q_i}|\mathbf{Y}(\omega)). \tag{4.6}$$

In the case that the classifiers' output are binary, $X_i \in \{0, 1\}$, the data model is in its simplest form and basically determined by the classifiers' error statistics *i.e.,* their specificity $p(X_i = 0|Y_i(\omega) = 0)$ and sensitivity $p(X_i = 1|Y_i(\omega) = 1)$. In the case of CNN classifiers with SoftMax output (Multinomial Logistic Regression) that target annoints the data distribution $p(X|Y)$ can be modeled using a Dirichlet distribution.

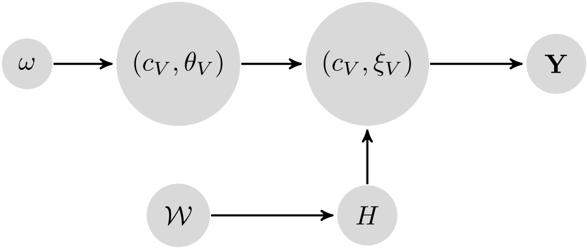


Figure 4.3: The functional relation between variables.

Finally, we sample the i-th element of ω according to the following probabilities:

$$p_\lambda(\omega_j = 0 | \mathbf{e}_{k-1}, \{\omega_l, \forall l \backslash j\}) = \frac{p_0}{p_0 + p_1}, \tag{4.7}$$

$$p_\lambda(\omega_j = 1 | \mathbf{e}_{k-1}, \{\omega_l, \forall l \backslash j\}) = \frac{p_1}{p_0 + p_1}, \tag{4.8}$$

where:

$$\begin{aligned} p_0 &= p_\lambda\big(\omega = (\omega_1, \cdots, \omega_j = 0, \cdots, \omega_L)\big) \prod_{i=1}^{k-1} p\big(X_{q_i} | (c_V, \theta_V)(\omega_1, \cdots, \omega_j = 0, \cdots, \omega_L)\big), \\ p_1 &= p_\lambda\big(\omega = (\omega_1, \cdots, \omega_j = 1, \cdots, \omega_L)\big) \prod_{i=1}^{k-1} p\big(X_{q_i} | (c_V, \theta_V)(\omega_1, \cdots, \omega_j = 1, \cdots, \omega_L)\big). \end{aligned} \tag{4.9}$$

4.2.2 Conditional Sampling of the T and $\mathcal{W}$

Replacing the GAG model with the Gibbs distribution in (3.2) results in:

$$p(\omega, T, \mathcal{W}) = p_\lambda(\omega | T) \, p(\mathcal{W}) \, p(T), \tag{4.10}$$

where, as a reminder, $\mathcal{W}$ is the set of camera's extrinsic and intrinsic parameters that specify the homography projection matrix H deterministically (see Appendix A). Note that the category label and 2D pose of objects *i.e.,* (c_V, ξ_V), can be computed based on $(\omega, \mathcal{W})$ by projecting the 3D poses to the image coordinate system using homography H. As was discussed before we learn $p_\lambda(\omega|T)$ for a number of different table geometries T (say 10 to 20) from the GAG model sample statistics whose corresponding table geometry match with T. Each of these table geometries has a probability $p(T)$ which is estimated based on the occurrence frequency of T in the dataset. We also have a proposal distribution $Q_T(T_{\text{current}}, T_{\text{try}})$ that proposes a change from the current table geometry T_{current} to T_{try}. Note that, for the transition to be well defined, we need to ensure that Q satisfies the weak symmetry condition:

$$Q_T(T_{\text{current}}, T_{\text{try}}) > 0 \Leftrightarrow Q_T(T_{\text{try}}, T_{\text{current}}) > 0. \tag{4.11}$$

We choose the table geometry proposal distribution to be a uniform distribution over the set of learned table geometries excluding the current table geometry. The proposed table geometry T_{try} gets accepted with probability:

$$\pi_T = \min\left(1, \frac{Q_T(T_{\text{try}}, T_{\text{cur}}) \times p_\lambda(\omega^{\text{try}} \mid T_{\text{try}}) \times p(T_{\text{try}}) \times \prod_{i=1}^{k-1} p(X_{q_i}|c_V^{\text{try}}, \xi_V^{\text{try}})}{Q_T(T_{\text{cur}}, T_{\text{try}}) \times p_\lambda(\omega^{\text{cur}} \mid T_{\text{cur}}) \times p(T_{\text{cur}}) \times \prod_{i=1}^{k-1} p(X_{q_i}|c_V^{\text{cur}}, \xi_V^{\text{cur}})}\right), \tag{4.12}$$

and rejected with probability $(1 - \pi_T)$. The variables in (4.12) are related according to:

$$\begin{tikzcd}
\omega^{\text{cur}} \arrow[r] & \theta_V^{\text{cur}} \arrow[r, shift left, "H_{\text{cur}}"] & \xi_V^{\text{cur}} \arrow[l, shift left, "H_{\text{cur}}^{-1}"] \arrow[r, shift left, "H_{\text{cur}}^{-1}"] & \theta_V^{\text{cur}} \arrow[l, shift left, "H_{\text{cur}}"] \arrow[r] & \omega^{\text{try}} \arrow[r] & \theta_V^{\text{try}} \arrow[r, shift left, "H_{\text{cur}}"] & \xi_V^{\text{try}} \arrow[l, shift left, "H_{\text{cur}}^{-1}"]
\end{tikzcd}$$

Note that we used the current homography H_{cur} for both "current" and "try" projections in above (H_{cur} is based on $\mathcal{W}_{\text{cur}}$). Also, we again skipped showing the uniformly distributed variables used to go from ω to ξ_V. Accepting the proposed table geometry sets $T_{\text{current}} = T_{\text{try}}$ and rejecting it sets $T_{\text{current}} = T_{\text{current}}$.

As was mentioned earlier, homography is a deterministic function of $\mathcal{W}$. We introduced a prior model on $\mathcal{W}$ in section 3.5 which will be used to calculate the proposal acceptance probability on camera parameters as follows. We define a camera parameters proposal distribution $Q_{\mathcal{W}}(\mathcal{W}_{\text{current}}, \mathcal{W}_{\text{try}})$ that chooses one of the camera parameters at random and proposes a change to it according to a normal distribution centered at its current value. However, as we will see in the next chapter the proposed changes to the camera parameters are constrained to result in a 2D table projection overlapping well-enough with the detected table area in the image. The proposed camera parameters $\mathcal{W}_{\text{try}}$ get accepted with probability:

$$\pi_{\mathcal{W}} = \min\left(1, \frac{Q_{\mathcal{W}}(\mathcal{W}_{\text{try}}, \mathcal{W}_{\text{cur}}) \times p(\mathcal{W}_{\text{try}}) \times \prod_{i=1}^{k-1} p(X_{q_i} | c_V^{\text{try}}, \xi_V^{\text{try}})}{Q_{\mathcal{W}}(\mathcal{W}_{\text{cur}}, \mathcal{W}_{\text{try}}) \times p(\mathcal{W}_{\text{cur}}) \times \prod_{i=1}^{k-1} p(X_{q_i} | c_V^{\text{cur}}, \xi_V^{\text{cur}})}\right), \tag{4.13}$$

and rejected with probability $(1 - \pi_{\mathcal{W}})$. The variables in (4.13) are related according to:

$$\begin{tikzcd} \xi_V^{\text{cur}} \arrow[r, shift left, "H_{\text{cur}}^{-1}"] & \theta_V^{\text{cur}} \arrow[l, shift left, "H_{\text{cur}}"] \arrow[r, shift left, "H_{\text{try}}"] & \xi_V^{\text{try}} \arrow[l, shift left, "H_{\text{try}}^{-1}"] \end{tikzcd}$$

Note that there is a difference in the way that ξ_V^{try} is compute in (4.13) and (4.12). The "try" 2D object poses ξ_V^{try} in (4.12) are calculated using the homography projection based on the "current" camera parameters $\mathcal{W}_{\text{cur}}$, whereas in (4.13) they are calculated using the homography projection based on the "try" camera parameters $\mathcal{W}_{\text{try}}$. Again, accepting the proposed camera parameters sets $\mathcal{W}_{\text{current}} = \mathcal{W}_{\text{try}}$ and rejecting it sets $\mathcal{W}_{\text{current}} = \mathcal{W}_{\text{current}}$.

Algorithm 1 summarizes the previous two subsections in three nested loops. We proposed a way to sample $p(\xi_V, \omega, T, \mathcal{W} \mid \mathbf{e}_{k-1})$. We can simply evaluate annobits for each drawn sample to estimate the marginal annobit posteriors $\{p(Y_i \mid \mathbf{e}_{k-1})\}_{i=1}^{K}$ required by EP.

Algorithm 1: Posterior Sampling

```
Initialize $T_{\text{cur}}$, $\mathcal{W}_{\text{cur}}$, and $\omega^{\text{cur}}$.
for $i \leftarrow 1$ to $N_T$ do
    Propose a table geometry $T_{\text{try}}$ and compute the acceptance ratio:
        $$\pi_T = \min\left(1, \frac{Q_T(T_{\text{try}}, T_{\text{cur}}) \times p_\lambda(\omega^{\text{try}} \mid T_{\text{try}}) \times p(T_{\text{try}}) \times \prod_{i=1}^{k-1} p(X_{q_i}|c_V^{\text{try}}, \xi_V^{\text{try}})}{Q_T(T_{\text{cur}}, T_{\text{try}}) \times p_\lambda(\omega^{\text{cur}} \mid T_{\text{cur}}) \times p(T_{\text{cur}}) \times \prod_{i=1}^{k-1} p(X_{q_i}|c_V^{\text{cur}}, \xi_V^{\text{cur}})}\right).$$
    Accept the proposed table geometry $T_{\text{cur}} = T_{\text{try}}$ with probability $\pi_T$ and reject $T_{\text{cur}} = T_{\text{cur}}$ with
    probability $(1 - \pi_T)$.
    for $n \leftarrow 1$ to $N_{\mathcal{W}}$ do
        Propose new camera parameters $\mathcal{W}_{\text{try}}$, and compute the acceptance probability:
            $$\pi_{\mathcal{W}} = \min\left(1, \frac{Q_{\mathcal{W}}(\mathcal{W}_{\text{try}}, \mathcal{W}_{\text{cur}}) \times p(\mathcal{W}_{\text{try}}) \times \prod_{i=1}^{k-1} p(X_{q_i}|c_V^{\text{try}}, \xi_V^{\text{try}})}{Q_{\mathcal{W}}(\mathcal{W}_{\text{cur}}, \mathcal{W}_{\text{try}}) \times p(\mathcal{W}_{\text{cur}}) \times \prod_{i=1}^{k-1} p(X_{q_i}|c_V^{\text{cur}}, \xi_V^{\text{cur}})}\right)$$
        Accept the proposed homography $\mathcal{W}_{\text{cur}} = \mathcal{W}_{\text{try}}$ with probability $\pi_{\mathcal{W}}$ and reject $\mathcal{W}_{\text{cur}} = \mathcal{W}_{\text{cur}}$
        with probability $(1 - \pi_{\mathcal{W}})$.
        for $t \leftarrow 1$ to $N_\omega$ do
            Sample the conditional Gibbs model for the current camera parameters and table
            geometry according to the following probabilities:
            $$p_\lambda(\omega_j = 0|\mathbf{e}_{k-1}, \{\omega_l, \forall l \setminus j\}) = \frac{p_0}{p_0 + p_1}, \quad p_\lambda(\omega_j = 1|\mathbf{e}_{k-1}, \{\omega_l, \forall l \setminus j\}) = \frac{p_1}{p_0 + p_1},$$
            where $p_0$ and $p_1$ are calculated based on (4.9). Update $\omega_{\text{current}}$, project the sample to the
            image coordinate system, compute the corresponding annobits, and update the
            estimated annobit posteriors $\hat{p}(\mathbf{Y} \mid \mathbf{e}_{k-1})$ accordingly.
        end
    end
end
```

4.3 Parameter Estimation

Assume we have N *iid*[1] samples from the Gibbs distribution in (4.2): $\omega^{(1)}, ..., \omega^{(N)}$. Let $\tilde{p}(\omega)$ denote the empirical distribution based on these N samples. The likelihood function

[1] Independent and Identically Distributed

of these samples is:

$$\mathcal{L}(\lambda_{\mathrm{R}}|\omega^{(1)},\cdots,\omega^{(N)})=\prod_{j=1}^{N}p_{\lambda}(\omega^{(j)}). \tag{4.14}$$

We will remove the "R" subscript hereafter for notational convenience from all feature functions, parameters, and the number of feature functions even though we use the reduced Gibbs distribution according to the invariance property. The maximum-likelihood estimator of the Gibbs model parameters is:

$$\lambda_{\mathrm{MLE}}=\underset{\lambda}{\operatorname{argmax}}\,\mathcal{L}(\lambda|\omega^{(1)},\cdots,\omega^{(N)}). \tag{4.15}$$

On the other hand, the log-likelihood function is proportional to the Kullback-Leibler (KL) divergence between the empirical distribution $\tilde{p}$ and the Gibbs model p_{λ} [16], as below:

$$\mathcal{L}(\lambda|\omega^{(1)},\cdots,\omega^{(N)})\propto e^{-N\times D_{\mathrm{KL}}(\tilde{p}\|p_{\lambda})}, \tag{4.16}$$

where $D_{\mathrm{KL}}(.\parallel.)$ denotes KL divergence. Hence:

$$\lambda_{\mathrm{MLE}}=\underset{\lambda}{\operatorname{argmin}}\,D_{\mathrm{KL}}(\tilde{p}\parallel p_{\lambda})=\underset{\lambda}{\operatorname{argmin}}\sum_{\omega}\tilde{p}(\omega)\log\frac{\tilde{p}(\omega)}{p_{\lambda}(\omega)}. \tag{4.17}$$

The loss function of the above optimization problem for p_{λ} being from the exponential family is always convex with respect to the model parameters λ [7]. However, this loss function is not necessarily strongly convex as the Hessian is only positive-semidefinite and not necessarily positive definite for every set of feature functions [7]. In another words, for the hessian matrix:

$$H(l) = \begin{bmatrix} \frac{\partial^2 l(\lambda)}{\partial \lambda_1^2} & \frac{\partial^2 l(\lambda)}{\partial \lambda_1 \partial \lambda_2} & \cdots & \frac{\partial^2 l(\lambda)}{\partial \lambda_1 \partial \lambda_M} \\ \frac{\partial^2 l(\lambda)}{\partial \lambda_2 \partial \lambda_1} & \frac{\partial^2 l(\lambda)}{\partial \lambda_2^2} & \cdots & \frac{\partial^2 l(\lambda)}{\partial \lambda_2 \partial \lambda_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 l(\lambda)}{\partial \lambda_M \partial \lambda_1} & \frac{\partial^2 l(\lambda)}{\partial \lambda_M \partial \lambda_2} & \cdots & \frac{\partial^2 l(\lambda)}{\partial \lambda_M^2} \end{bmatrix} = \mathbb{E}_{p_\lambda}\Big\{\big(\mathbf{f} - \mathbb{E}_{p_\lambda}\{\mathbf{f}\}\big)\big(\mathbf{f} - \mathbb{E}_{p_\lambda}\{\mathbf{f}\}\big)^\top\Big\} = \mathbf{cov}(\mathbf{f}),$$

where $l(\lambda) = D_{\text{KL}}(\tilde{p} \parallel p_\lambda)$, there is no $\alpha > 0$ such that for every set of feature functions $(\mathbf{cov}(\mathbf{f}) - \alpha I_M)$ is positive-definite where $\mathbb{E}_{p_\lambda}\{.\}$ and I_M, respectively, denote expectation with respect to probability distribution p_λ and an $M \times M$ identity matrix. We normally solve (4.17) for differentiable loss functions by taking the partial derivatives and equating them to zero

$$\nabla l(\lambda) = (\frac{\partial l}{\partial \lambda_1}, \cdots, \frac{\partial l}{\partial \lambda_M}) = \mathbf{0}. \tag{4.18}$$

If there exists a solution to (4.18) then this solution will in fact be the unique minimizer of (4.17) due to the convexity assumption of $l(.)$. Therefore, there is a close relationship between the optimization problem in (4.17) and the root-finding problem in (4.18). To solve the root-finding problem in (4.18), or equivalently the optimization problem in (4.17), we may iteratively take *gradient descent* steps as below:

$$\lambda^{t+1} = \lambda^t - \Gamma_t \, \nabla l(\lambda^t), \tag{4.19}$$

where Γ_t represents a diagonal matrix whose diagonal elements are step-sizes taken along different dimensions *i.e.,* $\Gamma_t = \text{diag}(\gamma_1^t, \cdots, \gamma_M^t)$. We denote the i-th gradient element by $g_i(\lambda)$ which for the objective function $l(\lambda) = D_{\text{KL}}(\tilde{p} \parallel p_\lambda)$ is given by:

$$g_i(\lambda) = \frac{\partial l}{\partial \lambda_i} = \mathbb{E}_{p_\lambda}\{f_i\} - \mathbb{E}_{\tilde{p}}\{f_i\}. \tag{4.20}$$

Computing the expectation $\mathbb{E}_{p_\lambda}\{f_i\}$ in (4.20) at every iteration according to (4.19) can become computationally challenging when the dimension of the random variable ω is large. To mitigate this issue we instead use an unbiased measurement of the gradient computed by Monte-Carlo integration:

$$G_i(\lambda, \zeta) = \frac{1}{|\zeta|} \sum_{\omega \in \zeta} f_i(\omega) - \mathbb{E}_{\tilde{p}}\{f_i\}, \tag{4.21}$$

where ζ denotes a set of samples from p_λ used to estimate the expectation (mini-batch). The stochastic subgradient $G_i(\lambda, \zeta)$ is a noisy but unbiased estimator of $g_i(\lambda)$ *i.e.,* $g(\lambda) := \mathbb{E}_\zeta\{G(\lambda, \zeta)\}$.

Stochastic Approximation (SA), sometimes referred to as the Stochastic Gradient (SG), was first proposed by Robbins and Monro [75] for solving nonlinear root-finding problems in the presence of noisy gradient measurements. The Stochastic Gradient method is the stochastic counterpart of the deterministic gradient descent method with the following iteration:

$$\lambda^{t+1} = \lambda^t - \Gamma_t \, G(\lambda^t, \zeta). \tag{4.22}$$

Since the first stochastic optimization algorithm by Robbins and Monro, numerous variations have been proposed [2,10,21,37,48,66,67,88,89,102]. This area, especially recently with the dawn of Big Data, has received a flurry of interest. These algorithms are variations of the original Robins-Monro algorithm. An important variant is the averaged stochastic gradient descent (ASGD), also referred to as iterate averaging (IA) [69, 80], where the converging sequence is:

$$\overline{\lambda^t} = \sum_{t'=0}^{t} \lambda^{t'}/(t+1). \tag{4.23}$$

We used this method to estimate the Gibbs model parameters. The convergence rate of the ASGD algorithm is asymptotically optimal [69, 80] and smoothness of the ASGD makes it attractive in cases where the measurements have large variance. However, ASGD is successful only if a proportionally large number of the iterates hover, in a balanced way, around the solution $\lambda^\star$ [92, p. 119]. Hence, to ensure convergence, the bulk of iterates should fall uniformly within a relatively tight proximity of the solution such that the average of them is close to $\lambda^\star$ with high confidence. Convergence of the averaged sequence, $\overline{\lambda^t}$, can be sluggish if the iterates do not reach this point relatively quickly. Furthermore, note that the issue of convergence with iterate averaging is even more pronounced in robust stochastic approximation (RSA) [66] where the converging sequence is a weighted average of the SGD iterates with decreasing weights. The ARSA's converging sequence (as $Q \to \infty$) of the i-th dimension is computed as below:

$$(\tilde{\lambda}_i)_K^Q := \sum_{t=K}^{Q} \tilde{\gamma}_i^t \lambda_i^t \ , \tag{4.24}$$

where $\tilde{\gamma}_i^t = \gamma_i^t / \sum_{t'=K}^{Q} \gamma_i^{t'}$ and $\gamma_i^t = \theta_0 / t^p$ for $p \in [1/2, 1]$ and K being the number of initial iterates excluded in computing the weighted average. The initial K iterates are excluded since these iterates are likely to produce the poorest estimates. The decreasing weights lower the significance of later iterates which are more likely to fall within a close proximity of the solution compared to the initial iterates. Hence, having poor initial iterates can negatively affect the performance of RSA even more severely compared to ASGD. Therefore, an acceleration of convergence, even in the early iterates, can substantially benefit RSA.

We propose an accelerating step-size based on the evidence collected from the most recent iterates via an oscillation measure. This accelerating step-size adds only negligible computational burden which makes is particularly attractive for large-scale problems. The proposed step-size gave substantial convergence speed-up in the parameter estimation of our Gibbs model. In addition, we learned singleton feature functions for each category separately and then put them together as a starting point to learn them jointly in the final learning step.

Consider an optimization problem where the loss function $l(\lambda)$ is univariate and convex as shown in Figure 4.4. Assume two scenarios where, in the first, the iterates at times $t-2$, $t-1$, and t (shown by red circles) lie on one side of the solution $\lambda^{\star}$, whereas in the second, these iterates (shown by blue circles) oscillate around the solution. In the first scenario, the iterates λ^{t-2} and λ^{t-1} are both incremented in the same direction to λ^{t-1} and λ^{t}, respectively. In this case, the step-size is perhaps too small and it should be increased. On the other hand, if consecutive iterates are oscillating back and forth (as in the second scenario) the step-size is perhaps too large and should be decreased. Consider the following recursive step-size strategy:

$$\gamma^{t} = \gamma^{t-1} \exp\Big(\mu \times \mathrm{sgn}\big((\lambda^{t} - \lambda^{t-1})(\lambda^{t-1} - \lambda^{t-2})\big)\Big), \tag{4.25}$$

where $\mu > 0$ and sgn(.) denotes the *sign* function serving as an oscillation measure. This step-size strategy scales down the previous step-size by $e^{-\mu}$ in the case of observing oscillation and scales it up by e^{μ} in the case that the iterates are incremented in the same direction. The step-size in (4.25), if computed recursively, is:

$$\begin{aligned}\gamma^{t} &= \gamma^{0}\gamma^{1} \prod_{t'=2}^{t} \exp\Big(\mu \times \mathrm{sgn}\big((\lambda^{t'} - \lambda^{t'-1})(\lambda^{t'-1} - \lambda^{t'-2})\big)\Big) \\ &= \gamma^{0}\gamma^{1}\ \exp\Big(\mu \times \sum_{t'=2}^{t} \mathrm{sgn}\big((\lambda^{t'} - \lambda^{t'-1})(\lambda^{t'-1} - \lambda^{t'-2})\big)\Big), \quad t \geq 2. \end{aligned} \tag{4.26}$$

However, noisy measurements of the gradient can result in an unbounded growth of step-sizes if computed based on (4.26) which leads to divergence of the algorithm in practice. A reasonable strategy to prevent divergence due to unbounded growth of step-sizes in (4.26) is to consider only the most recent pieces of evidence obtained from the oscillation measure:

$$\gamma^t = \theta_0 \, \exp\Bigg(\mu \times \sum_{t'=\check{T}(t-h)}^{t} \operatorname{sgn}\big((\lambda^{t'} - \lambda^{t'-1})(\lambda^{t'-1} - \lambda^{t'-2})\big) \Bigg), \;\; t \geq 2, \tag{4.27}$$

where h is the length of history, θ_0 is a positive scaling parameter, and $\check{T}(t-h) = \max(t-h+1, 2)$. In order to guarantee convergence, a decaying factor *e.g.,* $\frac{1}{t^p}$, is required to damp down the noise effect. Therefore:

$$\gamma^t = \frac{\theta_0}{t^p} \, \exp\Bigg(\mu \times \sum_{t'=\check{T}(t-h)}^{t} \operatorname{sgn}\big((\lambda^{t'} - \lambda^{t'-1})(\lambda^{t'-1} - \lambda^{t'-2})\big) \Bigg), \;\; t \geq 2. \tag{4.28}$$

The free parameters of the proposed accelerating step-size are (θ_0, μ, h, p) which should be chosen appropriately for the given problem. We chose $(\theta_0 = 1, \mu = 0.1, h = 10, p = 0.5)$ for the Gibbs parameter estimation.

The proposed accelerating step-size can adapt itself in case of too small or too large steps by appropriate scaling leading to improved finite-iterate convergence behavior of stochastic approximation algorithms. Storing the last h oscillation history requires very small memory since the history values are binary.

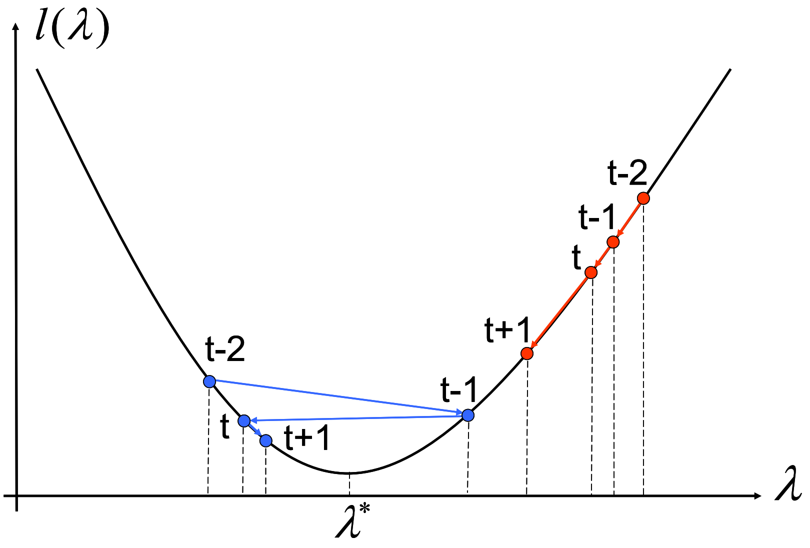


Figure 4.4: Univariate convex loss function.

The (4.22) update formula in the i^{th} dimension is:

$$\lambda_i^{t+1} = \lambda_i^t - \gamma_i^t G_i(\lambda^t, \zeta), \tag{4.29}$$

where γ_i^t is the step-size along the i-th dimension calculated according to the accelerating step-size in (4.28). The sequence $(\tilde{\lambda}_i)_K^Q$ computed according to (4.24) based on the proposed accelerating step-sizes converges to the solution denoted by $\lambda_i^\star$ as $Q \to \infty$. This is stated more formally in the following theorem. The assumptions that we make are very mild. We do not assume smoothness or strong convexity for the loss function that is common in the convergence proof of stochastic optimization algorithms. The given non-asymptotic upper bound is valid for a wide range of convex loss functions.

Theorem 4.3.1. *The loss function $l(\cdot)$ evaluated at $\tilde{\lambda}_K^Q = [(\tilde{\lambda}_1)_K^Q, ..., (\tilde{\lambda}_M)_K^Q]$ converges (in*

L^1-norm) to $l(\lambda^\star)$ when $Q \to \infty$ if the loss function is convex and the iterates are computed based on unbiased subgradient estimators G_i^t with finite energy i.e., $\mathbb{E}\{[G_i^t]^2\} < \infty$ and accelerating step-sizes in (4.28) *along each dimension with a bounded memory length $h < \infty$ and $p \in (1/2, 1]$. The suboptimality of the loss function at $\tilde{\lambda}_K^Q$ is upper bounded as*

$$0 \leq \mathbb{E}\Big\{l\big((\tilde{\lambda}_i)_K^Q\big) - l(\lambda_i^\star)\Big\} \leq e^{\mu.h}\left(\frac{d_i^K + \frac{1}{2}(\theta_0 e^{h\mu} M_i)^2\big(\frac{K^{1-2p}-Q^{1-2p}}{2p-1} + \frac{1}{K^{2p}}\big)}{\theta_0\frac{(Q+1)^{1-p}-K^{1-p}}{1-p}}\right),$$

which suggests a non-asymptotic convergence rate of $O(\frac{1}{\sqrt{t}})$ if $p \to 1/2$.

The proof of above theorem is given in below for an interested reader. However, there is one caveat; the reader should note that we have made assumptions that could be violated if the samples drawn from the model are dependent. This is the case for MCMC sampling where consecutive samples are strongly dependent. To mitigate this issue one can subsample the sequences of dependent samples generated by MCMC.

Proof. The update formula in (4.22) implies that λ^t is a deterministic function of the initial point λ^0 and all the ζ-samples used in the calculations up to $t-1$, denoted by $\zeta_{\lfloor t-1 \rfloor}$. Hence, assuming that λ^0 is constant we have:

$$\lambda^t = \lambda^t(\lambda^0, \zeta_{\lfloor t-1 \rfloor}) = \lambda^t(\zeta_{\lfloor t-1 \rfloor}). \tag{4.30}$$

Consequently the subgradient $g(\lambda^t)$ could be written as:

$$g(\lambda^t) = \mathbb{E}_{\zeta^t}\{G(\lambda^t, \zeta^t)\} = \mathbb{E}_{\zeta^t}\{G(\lambda^t(\zeta_{\lfloor t-1 \rfloor}), \zeta^t)|\zeta_{\lfloor t-1 \rfloor}\}, \tag{4.31}$$

where $\mathbb{E}_{\zeta^t}\{.\}$ denotes expectation with respect to ζ^t. In the following, we prove convergence along each individual dimension. Let:

$$D_i^t = \frac{1}{2}(\lambda_i^t - \lambda_i^\star)^2, \tag{4.32}$$

and:

$$d_i^t = \mathbb{E}\{D_i^t\} = \frac{1}{2}\mathbb{E}\big\{(\lambda_i^t - \lambda_i^\star)^2\big\}. \tag{4.33}$$

Note that $\mathbb{E}\{.\}$ (with no subscript) denotes expectation with respect to all of the random variables. Hence, in (4.33) the expectation is with respect to $\zeta_{\lfloor t-1 \rfloor}$. Using (4.29) we can write:

$$\begin{aligned} D_i^{t+1} &= \frac{1}{2}(\lambda_i^{t+1} - \lambda_i^\star)^2 = \frac{1}{2}\big(\lambda_i^t - \gamma_i^t G_i(\lambda^t, \zeta^t) - \lambda_i^\star\big)^2 \\ &= D_i^t + \frac{1}{2}(\gamma_i^t)^2 G_i^2(\lambda^t, \zeta^t) - \gamma_i^t(\lambda_i^t - \lambda_i^\star)G_i(\lambda^t, \zeta^t). \end{aligned}$$

Since (4.28) implies $\gamma_i^t \leq \frac{\theta_0 e^{h\mu}}{t^p}$ and the subgradient measurements are bounded we have:

$$\mathbb{E}\big\{(\gamma_i^t)^2 G_i^2(\lambda^t,\zeta^t)\big\} \leq \mathbb{E}\big\{(\frac{\theta_0 e^{h\mu}}{t^p})^2 G_i^2(\lambda^t,\zeta^t)\big\} \leq \big(\frac{\theta_0 e^{h\mu}}{t^p} M_i\big)^2. \tag{4.34}$$

Hence:

$$\mathbb{E}\big\{\gamma_i^t(\lambda_i^t - \lambda_i^\star) G_i(\lambda^t,\zeta^t)\big\} \leq d_i^t - d_i^{t+1} + \frac{1}{2}\big(\frac{\theta_0 e^{h\mu}}{t^p} M_i\big)^2. \tag{4.35}$$

On the other hand, by convexity of the loss function we have:

$$l(\lambda_i^t) + (\lambda_i - \lambda_i^t) g_i(\lambda^t) \leq l(\lambda_i), \tag{4.36}$$

where in (4.36), $l(\lambda_i)$ denotes the loss function $l(\cdot)$ as a function of λ_i while fixing the rest of parameters $\{\lambda_{(j)}, j \in \{1,2,\cdots,n\}\backslash i\}$. By letting $\lambda_i = \lambda_i^\star$ and using (4.31):

$$l(\lambda_i^t) - l(\lambda_i^\star) \leq (\lambda_i^t - \lambda_i^\star)\mathbb{E}_{\zeta^t}\big\{G_i(\lambda^t,\zeta^t)|\zeta_{\lfloor t-1 \rfloor}\big\}. \tag{4.37}$$

The solution $\lambda_i^\star$ is not a random variable for a given problem and λ_i^t depends only on $\zeta_{\lfloor t-1 \rfloor}$, therefore we can push the $(\lambda_i^t - \lambda_i^\star)$ term inside the expectation in (4.37); we then multiply both sides by γ_i^t and since $\gamma_i^t > 0$ we get:

$$\gamma_i^t\big(l(\lambda_i^t) - l(\lambda_i^\star)\big) \leq \gamma_i^t\, \mathbb{E}_{\zeta^t}\big\{(\lambda_i^t - \lambda_i^\star) G_i(\lambda^t,\zeta^t)|\zeta_{\lfloor t-1 \rfloor}\big\}. \tag{4.38}$$

The step-size γ_i^t does not depend on the random variable ζ^t and is fully determined by $\zeta_{\lfloor t-1 \rfloor}$. This is because γ_i^t depends on λ_i^t, λ_i^{t-1}, ..., λ_i^{t-h-1} and they are all determined by $\zeta_{\lfloor t-1 \rfloor}$. Hence, the step-size γ_i^t could also be pushed inside the conditional expectation to get:

$$\gamma_i^t \big(l(\lambda_i^t) - l(\lambda_i^\star) \big) \leq \mathbb{E}_{\zeta^t} \big\{ \gamma_i^t (\lambda_i^t - \lambda_i^\star) G_i(\lambda^t, \zeta^t) | \zeta_{\lfloor t-1 \rfloor} \big\}. \tag{4.39}$$

Finally by taking expectation with respect to $\zeta_{\lfloor t-1 \rfloor}$ we get:

$$\mathbb{E} \big\{ \gamma_i^t \, \big(l(\lambda_i^t) - l(\lambda_i^\star) \big) \big\} \leq \mathbb{E} \big\{ \gamma_i^t (\lambda_i^t - \lambda_i^\star) G_i(\lambda^t, \zeta^t) \big\}. \tag{4.40}$$

Hence, from (4.40) and (4.35) and since $\lambda_i^\star$ is the minimum solution *i.e.,* $l(\lambda_i^\star) \leq l(\lambda_i^t)$, we have:

$$0 \leq \mathbb{E} \big\{ \gamma_i^t \, \big(l(\lambda_i^t) - l(\lambda_i^\star) \big) \big\} \leq d_i^t - d_i^{t+1} + \frac{1}{2} \big(\frac{\theta_0 e^{h\mu}}{t^p} M_i \big)^2. \tag{4.41}$$

Applying $\sum_{t=K}^{Q} [.]$ to all sides of (4.41):

$$0 \leq \sum_{t=K}^{Q} \mathbb{E} \big\{ \gamma_i^t \, \big(l(\lambda_i^t) - l(\lambda_i^\star) \big) \big\} \leq \underbrace{\sum_{t=K}^{Q} [d_i^t - d_i^{t+1}]}_{=d_i^K - d_i^{Q+1}} + \frac{1}{2} (\theta_0 e^{h\mu} M_i)^2 \sum_{t=K}^{Q} \frac{1}{t^{2p}}. \tag{4.42}$$

By swapping the order by which the expectation and summation are applied in (4.42) and using $d_i^{Q+1} \geq 0$ we get:

$$0 \leq \mathbb{E}\Big\{\sum_{t=K}^{Q} \gamma_i^t l(\lambda_i^t)\Big\} - \mathbb{E}\Big\{\sum_{t=K}^{Q} \gamma_i^t l(\lambda_i^\star)\Big\} \leq d_i^K + \frac{1}{2}(\theta_0 e^{h\mu} M_i)^2 \sum_{t=K}^{Q} \frac{1}{t^{2p}}. \tag{4.43}$$

Recall that:

$$\tilde{\gamma}_i^t := \frac{\gamma_i^t}{\sum_{t'=K}^{Q} \gamma_i^{t'}}, \quad \text{and} \quad (\tilde{\lambda}_i)_K^Q := \sum_{t=K}^{Q} \tilde{\gamma}_i^t \lambda_i^t. \tag{4.44}$$

According to the Jensen's inequality for the convex loss function $l(\cdot)$ we have:

$$\sum_{t=K}^{Q} \gamma_i^t l(\lambda_i^t) \geq \Big(\sum_{t'=K}^{Q} \gamma_i^{t'}\Big) l\big((\tilde{\lambda}_i)_K^Q\big). \tag{4.45}$$

By taking expectation from both sides of (4.45):

$$\mathbb{E}\Big\{\sum_{t=K}^{Q} \gamma_i^t l(\lambda_i^t)\Big\} \geq \mathbb{E}\Big\{\Big(\sum_{t'=K}^{Q} \gamma_i^{t'}\Big) l\big((\tilde{\lambda}_i)_K^Q\big)\Big\}. \tag{4.46}$$

Therefore:

$$\begin{aligned}
0 &\leq \mathbb{E}\bigg\{\bigg(\sum_{t'=K}^{Q}\gamma_i^{t'}\bigg)l\big((\tilde{\lambda}_i)_K^Q\big)\bigg\} - \mathbb{E}\bigg\{\sum_{t=K}^{Q}\gamma_i^t l(\lambda_i^\star)\bigg\} \\
&= \mathbb{E}\bigg\{\bigg(\sum_{t'=K}^{Q}\gamma_i^{t'}\bigg)\bigg(l\big((\tilde{\lambda}_i)_K^Q\big) - l(\lambda_i^\star)\bigg)\bigg\} \\
&\leq \mathbb{E}\bigg\{\sum_{t=K}^{Q}\gamma_i^t l(\lambda_i^t)\bigg\} - \mathbb{E}\bigg\{\sum_{t=K}^{Q}\gamma_i^t l(\lambda_i^\star)\bigg\} \\
&\leq d_i^K + \frac{1}{2}(\theta_0 e^{h\mu}M_i)^2\sum_{t=K}^{Q}\frac{1}{t^{2p}}.
\end{aligned}$$

Succinctly:

$$0 \leq \mathbb{E}\bigg\{\bigg(\sum_{t'=K}^{Q}\gamma_i^{t'}\bigg)\bigg(l\big((\tilde{\lambda}_i)_K^Q\big) - l(\lambda_i^\star)\bigg)\bigg\} \leq d_i^K + \frac{1}{2}(\theta_0 e^{h\mu}M_i)^2\sum_{t=K}^{Q}\frac{1}{t^{2p}}. \tag{4.47}$$

Finally, we get an upper bound on the expected inaccuracy (in L^1-norm sense) as below:

$$0 \leq \mathbb{E}\bigg\{l\big((\tilde{\lambda}_i)_K^Q\big) - l(\lambda_i^\star)\bigg\} \leq e^{\mu.h}\bigg(\frac{d_i^K + \frac{1}{2}(\theta_0 e^{h\mu}M_i)^2\sum_{t=K}^{Q}\frac{1}{t^{2p}}}{\theta_0\sum_{t=K}^{Q}\frac{1}{t^p}}\bigg). \tag{4.48}$$

The upper bound on the expected inaccuracy when $Q \rightarrow \infty$ could be written as a function of the *Riemann Zeta* function:

$$\zeta(s) = \sum_{t=1}^{\infty}\frac{1}{t^s}, \quad s = \sigma + ic, \tag{4.49}$$

where in the general case s is a complex variable with σ and c being the real and imaginary parts, respectively. The value of this function is finite for $\sigma > 1$. For $\frac{1}{2} < p \leq 1$, a finite constant K, and $d_i^K < \infty$, the upper bound in (4.48) converges to zero as $Q \to \infty$. Therefore:

$$\lim_{Q\to\infty} \mathbb{E}\Big\{ l\big((\tilde{\lambda}_i)_K^Q\big) - l(\lambda_i^\star)\Big\} = 0. \tag{4.50}$$

which proves convergence in L^1-norm when $\frac{1}{2} < p \leq 1$. It will be seen that the upper bound still converges to zero when $p \to 1/2$ which completes the proof. We know from probability theory that convergence in norm guarantees convergence in probability.

The terms in the upper bound of the L^1-norm expected inaccuracy in (4.48) that depend on the iterate number Q are:

$$\mathcal{A}(p, K, Q) = \sum_{t=K}^{Q} \frac{1}{t^{2p}} \quad \text{and} \quad \mathcal{B}(p, K, Q) = \sum_{t=K}^{Q} \frac{1}{t^{p}}. \tag{4.51}$$

Let $\mathcal{A}^+$ and $\mathcal{B}^-$ respectively denote an upper bound on $\mathcal{A}$ *i.e.,* $\mathcal{A} \leq \mathcal{A}^+$, and a lower bound on $\mathcal{B}$ *i.e.,* $\mathcal{B}^- \leq \mathcal{B}$. Hence:

$$0 \leq \mathbb{E}\Big\{ l\big((\tilde{\lambda}_i)_K^Q\big) - l(\lambda_i^\star)\Big\} \leq e^{\mu.h}\left(\frac{d_i^K + \frac{1}{2}(\theta_0 e^{h\mu} M_i)^2 \mathcal{A}^+}{\theta_0 \mathcal{B}^-} \right). \tag{4.52}$$

We can calculate the following bounds (see Appendix C):

$$\mathcal{A}^+ = \frac{K^{1-2p} - Q^{1-2p}}{2p-1} + \frac{1}{K^{2p}}, \qquad \mathcal{B}^- = \frac{(Q+1)^{1-p} - K^{1-p}}{1-p}. \tag{4.53}$$

Finally, by plugging $\mathcal{A}^+$ and $\mathcal{B}^-$ into (4.52) we get the following bound:

$$0 \leq \mathbb{E}\Big\{ l\big((\tilde{\lambda}_i)_K^Q\big) - l(\lambda_i^\star)\Big\} \leq e^{\mu.h}\left(\frac{d_i^K + \frac{1}{2}(\theta_0 e^{h\mu} M_i)^2\big(\frac{K^{1-2p}-Q^{1-2p}}{2p-1} + \frac{1}{K^{2p}}\big)}{\theta_0 \frac{(Q+1)^{1-p}-K^{1-p}}{1-p}}\right). \quad (4.54)$$

An interesting case is when $p \to 1/2$. In this case, both the numerator and denominator of the fraction $(K^{1-2p} - Q^{1-2p})/(2p-1)$ in (4.54) go to zero. After disambiguation of the upper bound by L'Hopital's rule we get an asymptotic convergence rate of $O(\frac{1}{\sqrt{t}})$ as $p \to 1/2$. This proves the theorem. □

Note that in the case of a constrained optimization problem where $\lambda_i \in \Lambda_i$, a projection onto Λ_i i.e., $\Pi_\Lambda(\lambda_i) = \text{argmin}_{\lambda_i' \in \Lambda_i}(\lambda_i - \lambda_i')^2$, is often used to update the iterates:

$$\lambda_i^{t+1} = \Pi_\Lambda\big(\lambda_i^t - \gamma_i^t G_i(\lambda^t, \zeta^t)\big).$$

If the projection $\Pi_\Lambda(.)$ is non-expanding, namely:

$$\big(\Pi_\Lambda(\lambda_i) - \Pi_\Lambda(\lambda_i')\big)^2 \leq (\lambda_i - \lambda_i')^2 \quad \forall \lambda_i, \lambda_i' \in \mathbb{R},$$

then, considering that $\Pi_\Lambda(\lambda_i^\star) = \lambda_i^\star$ (because $\lambda_i^\star \in \Lambda_i$) we have:

$$\begin{aligned} D_i^{t+1} &= \frac{1}{2}(\lambda_i^{t+1} - \lambda_i^\star)^2 = \frac{1}{2}\Big(\Pi_\Lambda\big(\lambda_i^t - \gamma_i^t G_i(\lambda^t, \zeta^t)\big) - \lambda_i^\star\Big)^2 \\ &= \frac{1}{2}\Big(\Pi_\Lambda\big(\lambda_i^t - \gamma_i^t G_i(\lambda^t, \zeta^t)\big) - \Pi_\Lambda(\lambda_i^\star)\Big)^2 \\ &\leq \frac{1}{2}\big(\lambda_i^t - \gamma_i^t G_i(\lambda^t, \zeta^t) - \lambda_i^\star\big)^2 \\ &= D_i^t + \frac{1}{2}(\gamma_i^t)^2 G_i^2(\lambda^t, \zeta^t) - \gamma_i^t(\lambda_i^t - \lambda_i^\star) G_i(\lambda^t, \zeta^t). \end{aligned}$$

Hence, in both constrained and unconstrained cases the following is true:

$$\mathbb{E}\{D_i^{t+1}\} \leq \mathbb{E}\{D_i^t\} + \frac{1}{2}\mathbb{E}\big\{(\gamma_i^t)^2 G_i^2(\lambda^t, \zeta^t)\big\} - \mathbb{E}\big\{\gamma_i^t(\lambda_i^t - \lambda_i^\star) G_i(\lambda^t, \zeta^t)\big\}. \tag{4.55}$$

The above leads to (4.35) and the rest of proof remains unchanged.

Chapter 5

Classifiers and Data Model

We used classifiers based on state-of-the-art deep Convolutional Neural Networks (CNN). Since a thorough description of deep neural networks is out of the scope of this dissertation, and because we use CNNs from “off-the-shelf”, we only describe them briefly and just to the extent necessary to understand their role in the EP framework.

Deep neural networks are *loosely* inspired by how the brain works. Deep learning methods aim at learning feature hierarchies where features from higher levels are compositions of lower level features. Th expression of higher level abstractions in terms of raw sensory input has always been a challenge for many machine learning applications. As the amount of data continues to grow, an automatic and end-to-end learning framework that does not depend on human-crafted features becomes increasingly attractive. The depth of a neural network architecture refers to the number of levels of composition of non-linear operations in the function learned. Inspired by the multistage information processing in the primate visual system (detection of edges, primitive shapes, and moving up to more complex visual shapes) and also the multilayer structure of the mammalian brain [5, 87], researchers had always wanted to train neural networks with multiple hidden layers; however, their attempts were largely unsuccessful until Hinton *et al.* introduced Deep Belief Networks (DBNs) with a learning algorithm using an unsupervised pre-training that greedily targeted one layer at a time [38]. Although deep neural networks were found too difficult to train before Hinton's unsupervised pre-training, the CNN was an exception. The CNNs were inspired by the mechanism of the visual system and in particular were designed based on the models proposed in 1962 by Hubel and Wiesel for the visual system of cats [43]. The first computational model, called the *Neocognitron*, was proposed by Fukushima [24]. Later, following up on this idea, LeCun *et al.* designed and trained CNNs using the error gradients [52, 53]. Our understanding of the visual system seems to be roughly consistent with convolutional networks [87], at least for quick object recognition without benefiting from visual selective attention and top-down contextual feedback connections [5]. In this dissertation, we use a model-based approach in an attempt to incorporate top-down contextual information, and analyze scenes in a coarse-to-fine fashion inspired by the visual selective attention property of the human visual system.

The proper training of a fully connected neural network when initialized randomly is much harder than a CNN with the same number of layers. One hypothesis for this according to [5] is that the small fan-in of the convolutional neurons *i.e.,* few inputs per neuron, helps gradients propagate through so many layers without diffusing so much as to become useless. Note that gradient diffusion can occur as the result of propagation through many paths so that the credit or blame for the output error is distributed too widely and thinly to be effective during optimization [5].

The current CNNs are designed based on the same principles introduced years ago in the early works [42, 53]. The famous LeNet-5 network [53] was a successful implementation of CNNs for digit classification. However, the LeNet approach did not perform well in training larger networks (in both breadth and depth) required for more complex problems such as object classification. This led to abandonment of neural networks by the majority of machine learning community. However, more efficient ways to train neural networks with more layers [6, 38, 73] together with far larger (annotated) training sets and efficient implementations on high-performance computing systems, such as GPUs and large-scale distributed clusters [14, 15], resulted in impressive performance of CNNs on a number of benchmarks. The sucess of deep networks was largely due to the large public image repositories such as ImageNet [19].

Deep CNN classifiers received a lot of attention following the good performance of AlexNet [49] reported for the 2010 and 2012 *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [81]. The ILSVRC-2010 competition provided 1.2 million high-resolution training images from 1000 different classes with roughly 1000 images per class; the validation and test data for this competition consisted of 200,000 images (50,000 validation and 150,000 test). The AlexNet achieved top-1 and top-5 error rates of 37.5% and 17.0% on the test data which was considerably better than the previous state-of-the-art. A

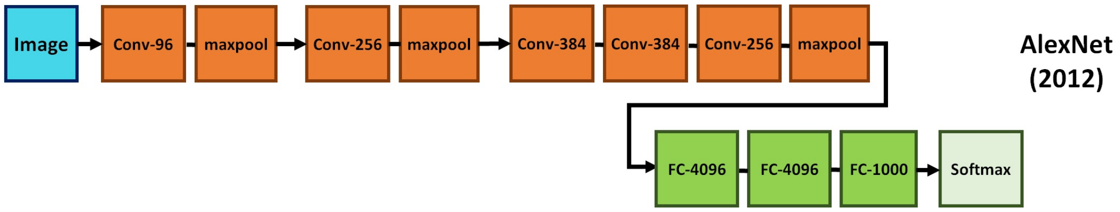


Figure 5.1: Block diagram of AlexNet architecture. The numbers following each block's label indicates the number of output for that block.

variant of the AlexNet was also entered in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry. The AlexNet's architecture, shown in Figure 5.1, consists of five convolutional layers (some of which are followed by max-pooling layers), three fully-connected layers and a final 1000-way softmax; this architecture resulted in 650,000 nodes (neurons) and 60 million parameters. The network's size is mainly limited by the amount of memory available on the GPU in addition to the amount of training time that one is willing to tolerate. The AlexNet was too large to fit in the memory of GPUs that were available at the time; therefore, the net was spread across two GTX 580 3GB GPUs by putting half of the kernels (or neurons) on each GPU.

AlexNet inputs a 256×256 pixels patch whereas the ImageNet dataset consists of variable-resolution images; therefore, they first rescaled the image such that the shorter side was of length 256, and then cropped out the central 256×256 patch from the resulting image.

The classifiers based on CNNs in general require a constant input dimensionality. After rescaling, processing any patch requires the same amount of computations; hence, the classifiers based on CNNs deploying the same network architecture will have *unit cost* making them theoretically ideal for the entropy pursuit framework that relies on unit cost tests.

The size of networks has grown constantly in the past few years taking advantage of the newer generations of GPUs with larger memory and higher computing capability (higher FLOPS [1]). The performance has gotten better with the use of faster GPUs and larger training datasets. The network architecture that we used to design our classifiers is based on the VGG-16 [91] with 16 layers that won first and second places in the ILSVRC-2014 competition for localisation and classification tasks, respectively. Figure 5.2 shows the block diagram of the VGG-16 network. The input to this network is a fixed size 224×224 RGB image patch. The image is passed through a stack of convolutional layers with fixed 1 pixel stride [2]. Spatial pooling is carried out by five max-pooling layers (over 2×2 pixel windows with stride 2) which follow only some of the convolutional layers. The convolutional layers are followed by three Fully Connected (FC) layers, where the first two have 4096 channels each and the third has 1000 channels (one for each of the 1000 ILSVRC-2014 classes). The 1000 outputs of the last fully connected layers are finally passed to a softmax layer. This network architecture has 138 million parameters. The VGG network uses very small 3×3 filters in convolutional layers which was reported to show significant improvement when used with 16-19 weight layers. The top-performing entries of the ILSVRC-2012 [49] and ILSVRC-2013 [86, 103] competitions used larger filters of size 11×11 with stride 4, and 7×7 with stride 2, respectively. Note that a stack of two 3×3 convolutional layers (without

[1] floating-point operations per second

[2] The shifting step for convolutional filters is usually refered to as “stride”.

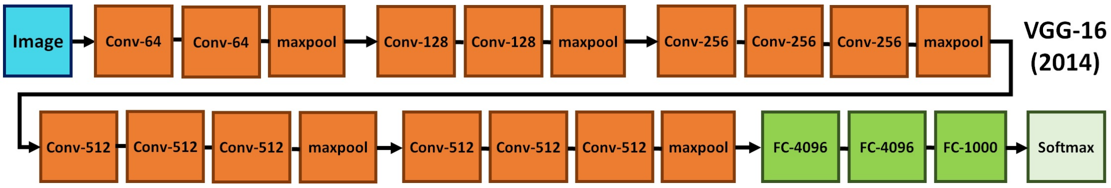


Figure 5.2: Block diagram of VGG-16 network architecture. The numbers following each block's label indicates the number of output for that block.

spatial pooling in between) has an effective receptive field of 5×5, and three such layers have a 7×7 effective receptive field. However, since each convolutional layer is followed by a non-linear rectification layer, the decision function of three cascaded convolutional layers will be more discriminative than a single convolutional layer with 7×7 filter.

The final layer *i.e.,* the softmax layer, simply performs multinomial logistic regression and is used for making prediction. Letting the 4096-dimensional column vector $\mathbf{x}$ denote the last FC-4096 layer's output, and the 4096-dimensional column vector $\mathbf{w}_j$ denote the weights associated with the j-th output of the FC-1000 layer offsetted by b_j, the softmax layer's j-th output is:

$$p_j = \frac{e^{\mathbf{x}^\top \mathbf{w}_j + b_j}}{\sum_k e^{\mathbf{x}^\top \mathbf{w}_k + b_k}}, \tag{5.1}$$

where p_j is an estimator of class j proportion given the input patch I, *i.e.,* $p(J = j|I)$. Obviously, the estimated proportions for every class are non-negative and sum to one: $\sum_j p_j = 1$. See Appendix D for a brief description about how the network weights are learned.

5.1 CNN Classifiers for EP

We trained three deep CNNs, the first one called “CatNet” which we used for object category classification, the second one called “ScaleNet” to estimate the scale of detected object instances, and the third we call “TableNet” which was used to detect the table surface area in a given image. All of these CNNs borrow their network architecture, up to the last weight layer, *i.e.,* layer 15, from the VGG-16 network. The last fully-connected layer (16-th weight layer) and the following softmax layer of these three CNNs are modified to accommodate our design needs. All CNNs rely on “transfer learning” by initializing the first 15 weight-layers to the corresponding weights from the VGG-16 network [3] trained on 1.2 million images from the ImageNet dataset [18]. However, since the last layer’s architecture for each of the three CNNs is different from VGG-16, they were randomly initialized during training. Because the loss function of deep neural networks is highly non-convex, appropriate weight initialization is important to achieve faster convergence to a good local or perhaps global optima. The input to the VGG-16 network is a fixed size 224×224 RGB image patch. By design, an input patch of any size is first resized to 256×256 before being processed; then, 5 patches of size 224×224 are cropped from the resized patch, one for each of the four corners and the center of the 256×256 patch (see Figure 5.3). Finally, the horizontal mirror of these five patches are added resulting in 10 total 224×224 patches from the input patch. The final output is the average output for these 10 patches. All of the CNNs were trained and tested using the Cafe Deep Learning

[3] Available at: `http://www.robots.ox.ac.uk/~vgg/research/very_deep/`

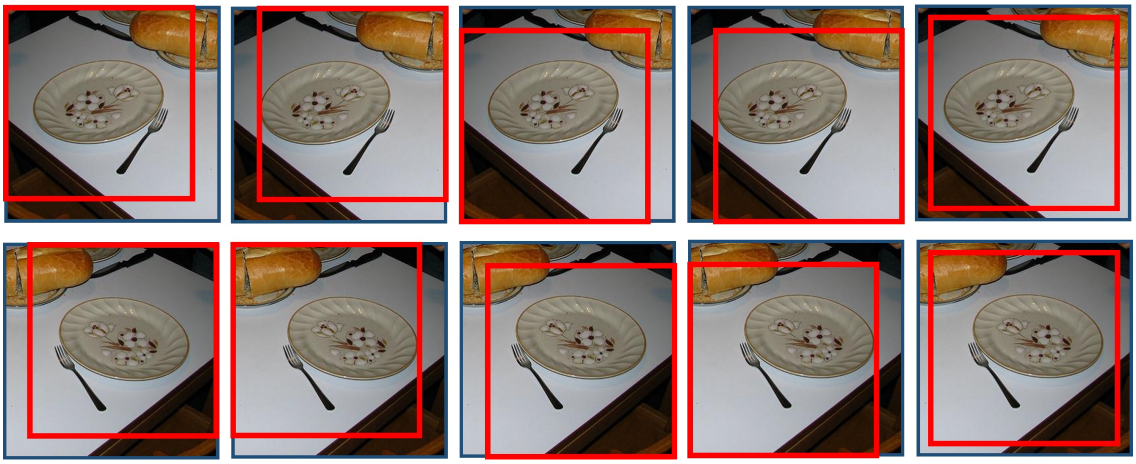


Figure 5.3: Ten 224 × 224 patches extracted from a 256 × 256 patch. The red boxes illustrate the cropped patches. The first row shows the original patch and its crops and the second row shows the horizontal mirror and its crops.

framework [45] using a Nvidia Tesla K40 GPU on a desktop computer with Intel i7-4790K Quad-Core processor (8M Cache and up to 4.40 GHz clock rate) and 32-GB RAM running Kubuntu 15.04 operating system. The processing time for one 256 × 256 patch (resulting in 10 patches of size 224 × 224) is about 12 seconds on our end-of-the-line Intel i7-4790K CPU and 0.2 seconds on the Tesla K40 GPU. Again, since the input patches are of the same size and pass through the same network the classifiers all have the same computational cost during test time. We describe the design, training, and performance evaluation of these CNNs in the following subsections.

5.1.1 CatNet

We are interested in detecting if at least one instance from each of the four object categories “Plate”, “Bottle”, “Glass”, and “Utensil” exists in a given patch. CatNet will be used to predict the annoint associated with the patch that basically takes values on the set $\{0, 1, ..., 15\}$, where 0 represents no instance from any of the four object categories being fully contained in the patch, and 15 indicates the presence of at least one instance from every object category contained in the patch. The CatNet is a CNN with a 5-way softmax output layer where the first output corresponds to the class of "No Object" instance from any of the four categories and the second, third, fourth, and fifth outputs correspond to "Plate", "Bottle", "Glass", and "Utensil" classes, respectively. Note that reducing the $2^4 = 16$ possible states of a patch to only 5, whereas crude, does scale linearly with the number of categories (rather than exponential $2^{|\mathcal{C}|}$).

We trained CatNet such that the softmax outputs predict the relative existence probabilities of the corresponding categories in the input patch. Note that the outputs of CatNet do not predict the probability of each category having at least one instance. Assume A_1, A_2, A_3, A_4, and A_5 denote, respectively, the events that "No Object", "Plate", "Bottle", "Glass", and "Utensil" categories exist in the input patch (in the sense of at least one instance for the object classes). Since A_1 is the complement of union event $\cup_{i=2}^{5} A_i$, we have $p(\cup_{i=2}^{5} A_i) = 1 - p(A_1)$. On the other hand, the softmax layer's outputs sum to one: $p(A_1) + \sum_{i=2}^{5} p(A_i) = 1$. Hence, $p(\cup_{i=2}^{5} A_i) = \sum_{i=2}^{5} p(A_i)$. Therefore, every pair of events must be disjoint and cannot co-occur ($p(A_i \cap A_j) = 0, \forall i \neq j$). Obviously, this is not the case since the input patch may contain multiple categories. Hence, the softmax outputs are interpreted to predict relative probabilities (proportions) for the existence of different categories. For instance, the outputs corresponding to plate and utensil categories, namely

OUTPUT 2 and OUTPUT 5, will take higher values if instances of both of these categories co-exist in the input patch.

We trained CatNet using a set of 344,149 patches, and evaluated the performance on a test set of 62,157 patches. The training set contained 170,830 patches from the “No Object” category, 36,429 patches from the “Plate” category, 2,074 patches from the “Bottle” category, 49,401 patches from the “Glass” category, and 85,415 patches from the “Utensil” category. Figure 5.4 illustrates a snapshot of the training set where each patch is framed with one or multiple colored frames where each colored frame indicates existence of one object category in the patch; black for “No Object”, red for “Plate”, yellow for “Bottle”, blue for “Glass”, and green for “Utensil”. Each train and test patch I_i is labeled with a category l_i, where $1 \leq l_i \leq 5$. A patch with label “1” contains no object instance from the four considered categories. A patch with label “2” contains at least one plate instance completely inside the patch. A patch with label “3” contains at least one bottle instance completely inside the patch, and so on. Note that since training patches are from different levels of resolution, each patch may contain multiple object categories depending on its resolution level and the pose of objects. A patch including multiple object instances appears multiple times in the training set, each time with the category label of one of the existing instances. For instance, assume that there are one plate and two glasses in a given patch; then, this patch will be repeated three times in the training set, one time with label “2” and two times with label “4”. Repeating the same patch with label say “4” as the result of multiple glass instances being present in the patch gives a higher weight to the “Glassiness” of this patch in the loss function (D.1).

The train and test patches were extracted from the “JHU Table-Setting Dataset” using the annocell hierarchy image partitioning explained in section 2.2. The “No Object” category patches were selected from the set of annocell patches whose overlap with the table area is less than 10% of the patch. The number of background (No Object) training patches was chosen to be twice the number of patches from the most frequent category (the utensil category) as this seemed to result in better performance by minimizing the number of false positives in the off-table areas and still detect other categories decently. If we include too many background patches in the training dataset, say 99 times patches from all other categories, then, even if we classified all training patches to class “1” (namely background or “No Object”) we would still get a very high 99% training accuracy, as the loss function minimization in (D.1) would be biased toward class “1”. On the other hand, if we lower the number of background patches then we will start to see more false positives in off-table areas.

The CatNet was trained by minimizing the cross-entropy loss function using the SGD with momentum with min-batch size of 50 patches (see Appendix D). Training took about 24 hours when the first 15 weight layers were initializing by the first 15 weight layers from the VGG-16 network.

To obtain binary classification for the existence of at least one instance from each category in a given patch we process the softmax layer output relative probabilities as follows. Assuming that the net is trained well, we expect that the output proportional scores corresponding to the existing categories in the input patch to be close and amongst the top scores. That is, we consider the probabilities outputted by the CatNet as scores for the corresponding categories. We define two parameters (k, S_g) for considering the top-k scores with less than S_g consecutive score gap (distance). For instance, assume we are considering the top-3 scores with score gap $S_g = 0.2$, and the CatNet outputs are $(s_1 = 0.05, s_2 = 0.45, s_3 = 0.05, s_4 = 0.1, s_5 = 0.35)$; since $(s_2 - s_5) < S_g$ but $(s_5 - s_4) \not< S_g$, the binary classification per category will only recognize “2” and “5” categories, *i.e.,* plate and utensil are classified to be the only existing categories in the input patch. Figure 5.5 and Figure 5.6 show, respectively, the confusion matrices for the train and test set with patches from different levels of resolution from the annocell hierarchy for different score gap values. As a reminder, the 1, 2, 3, 4, and 5 classes correspond to the No Object, Plate, Bottle, Glass, and Utensil categories, respectively. In the confusion matrices, the rows show the predicted class statistics, and the columns show the true class statistics. The diagonal cells show where the true class and predicted class match and the off diagonal cells show the number of points where the CatNet classifier has made a mistake. The column on the right hand side of the plot shows the accuracy for each predicted class, whereas the row at the bottom shows the accuracy for each true class. The overall accuracy is shown in the cell in the bottom right of the plots (yellow box) in green text. Note that the score gap $S_g = 0$ is equivalent to choosing only the max score. One can see that the accuracy increases as the score gaps grows. We compare elements of the target and prediction vectors to calculate a confusion matrix; an element in the prediction vector is set

Table 5.1: Average score at different levels of resolution observed at each output when CatNet is applied to an input patch from the corresponding class.

Category	Level-0	Level-1	Level-2	Level-3
“No Object”	0.31	0.72	0.96	0.99
“Plate”	0.32	0.34	0.39	0.44
“Bottle”	0.08	0.19	0.31	0.36
“Glass”	0.33	0.44	0.56	0.68
“Utensil”	0.48	0.54	0.71	0.81

to its target class if the target class is among the top-k scores with score gap not exceeding S_g (*i.e.,* match), and set to the top scoring class otherwise (*i.e.,* mistake). It is important to note that the accuracies reported by the confusion matrices indicate the classifiers' “sensitivity” also called true positive rate. The classifier's sensitivity always grows by increasing the score gap; but, their “specificity” (also called the true negative rate) drops by increasing the score gap. Usually, there is a trade-off between sensitivity and specificity of classifiers. We struck a balance between sensitivity and specificity by choosing $S_g = 0.3$ considering the top-3 scores.

We expect to achieve higher classification accuracy for patches from finer levels of the annocell hierarchy since finer resolutions usually include fewer object categories and in relatively larger scale compared to the patch size. Hence, we expect to observe a higher average score on the CatNet's i-th output if it is fed with a finer patch from the i-th class. This is confirmed by Table 5.1 which shows the average score on the CatNet's outputs when it is fed by patches from the corresponding class at different resolution levels from the annocell hierarchy. Figure 5.7 shows the distribution of scores whose means are listed in Table 5.1. One can see that as the input patch gets finer (from left columns to the right) the distributions moves more toward higher values. Figures 5.8, 5.9, 5.10, 5.11 show confusion matrices on the test set broken down for different levels of resolution and score gaps; see Appendix E for the corresponding confusion matrices on training set. As we could expect, the overall accuracy (sensitivity) is: (1) higher at finer levels compared to coarse levels, (3) higher for larger score gaps, and (3) higher on the training dataset compared to the test set.

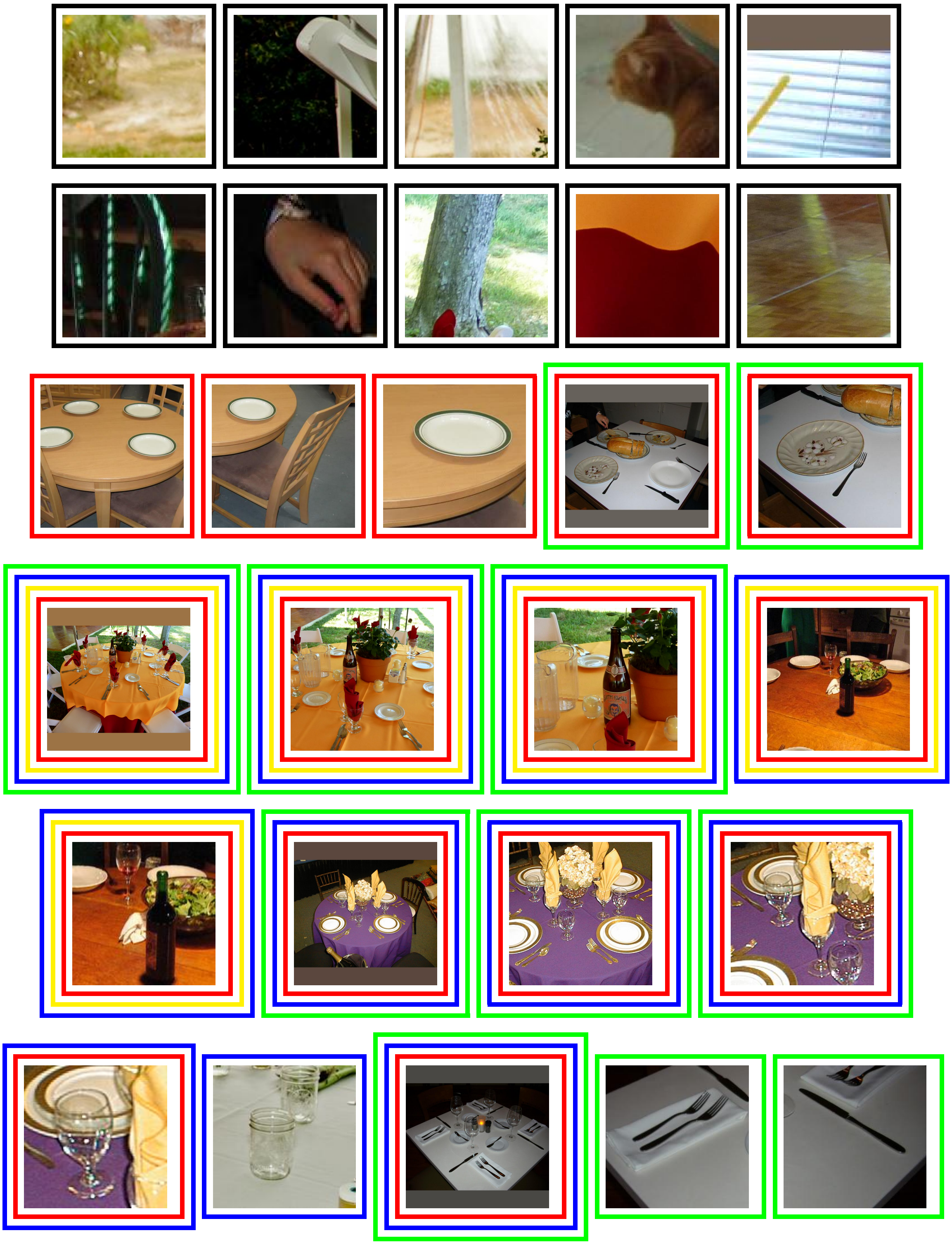
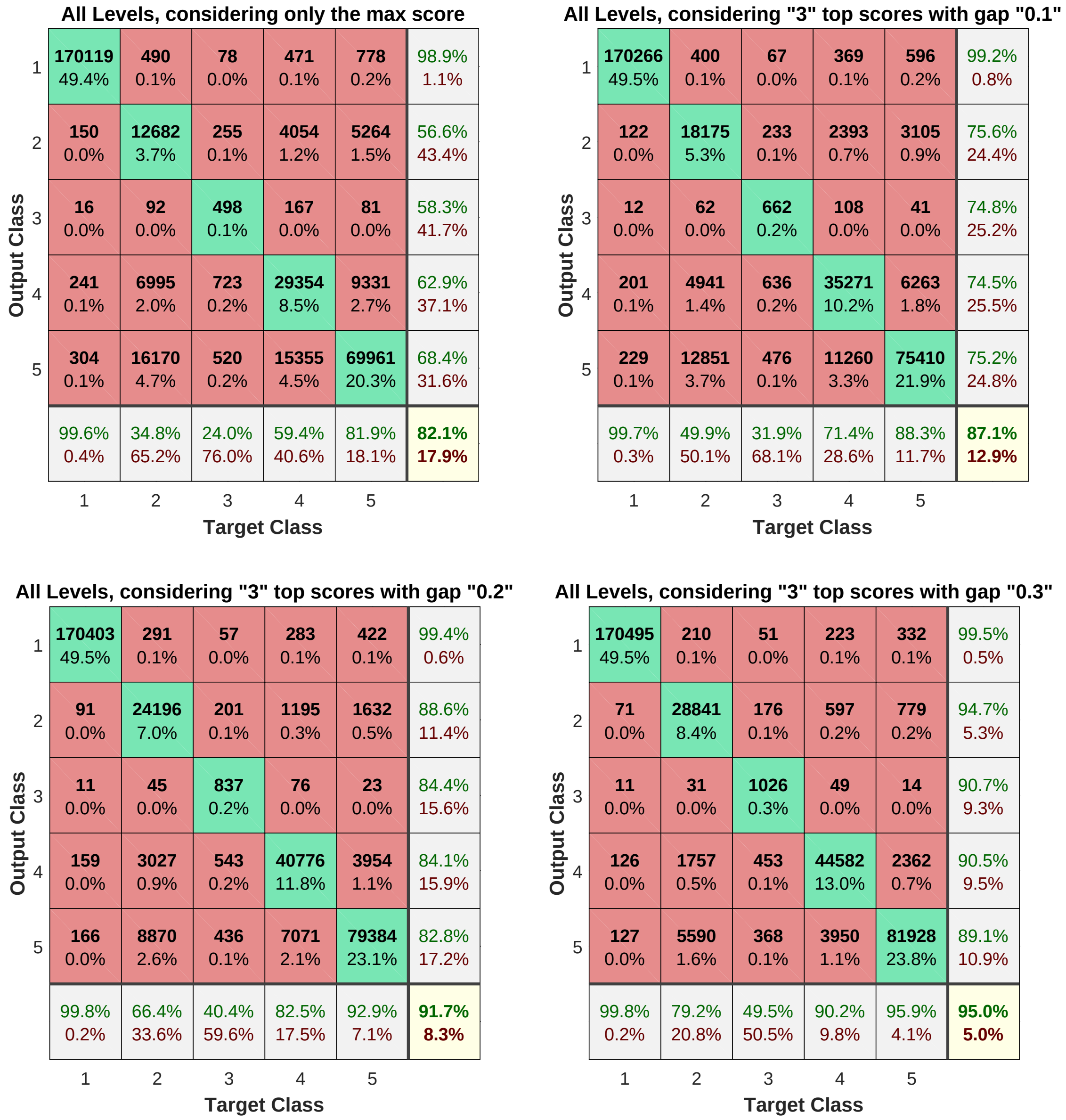


Figure 5.4: A snapshot of the CatNet datset set. Each colored frame indicates existence of one object category in the patch: black indicates “No Object”, red indicates “Plate”, yellow indicates “Bottle”, blue indicates “Glass”, and green indicates “Utensil” category.



All Levels, considering only the max score

	1	2	3	4	5	
1	170119 49.4%	490 0.1%	78 0.0%	471 0.1%	778 0.2%	98.9% 1.1%
2	150 0.0%	12682 3.7%	255 0.1%	4054 1.2%	5264 1.5%	56.6% 43.4%
3	16 0.0%	92 0.0%	498 0.1%	167 0.0%	81 0.0%	58.3% 41.7%
4	241 0.1%	6995 2.0%	723 0.2%	29354 8.5%	9331 2.7%	62.9% 37.1%
5	304 0.1%	16170 4.7%	520 0.2%	15355 4.5%	69961 20.3%	68.4% 31.6%
	99.6% 0.4%	34.8% 65.2%	24.0% 76.0%	59.4% 40.6%	81.9% 18.1%	**82.1%** **17.9%**

All Levels, considering "3" top scores with gap "0.1"

	1	2	3	4	5	
1	170266 49.5%	400 0.1%	67 0.0%	369 0.1%	596 0.2%	99.2% 0.8%
2	122 0.0%	18175 5.3%	233 0.1%	2393 0.7%	3105 0.9%	75.6% 24.4%
3	12 0.0%	62 0.0%	662 0.2%	108 0.0%	41 0.0%	74.8% 25.2%
4	201 0.1%	4941 1.4%	636 0.2%	35271 10.2%	6263 1.8%	74.5% 25.5%
5	229 0.1%	12851 3.7%	476 0.1%	11260 3.3%	75410 21.9%	75.2% 24.8%
	99.7% 0.3%	49.9% 50.1%	31.9% 68.1%	71.4% 28.6%	88.3% 11.7%	**87.1%** **12.9%**

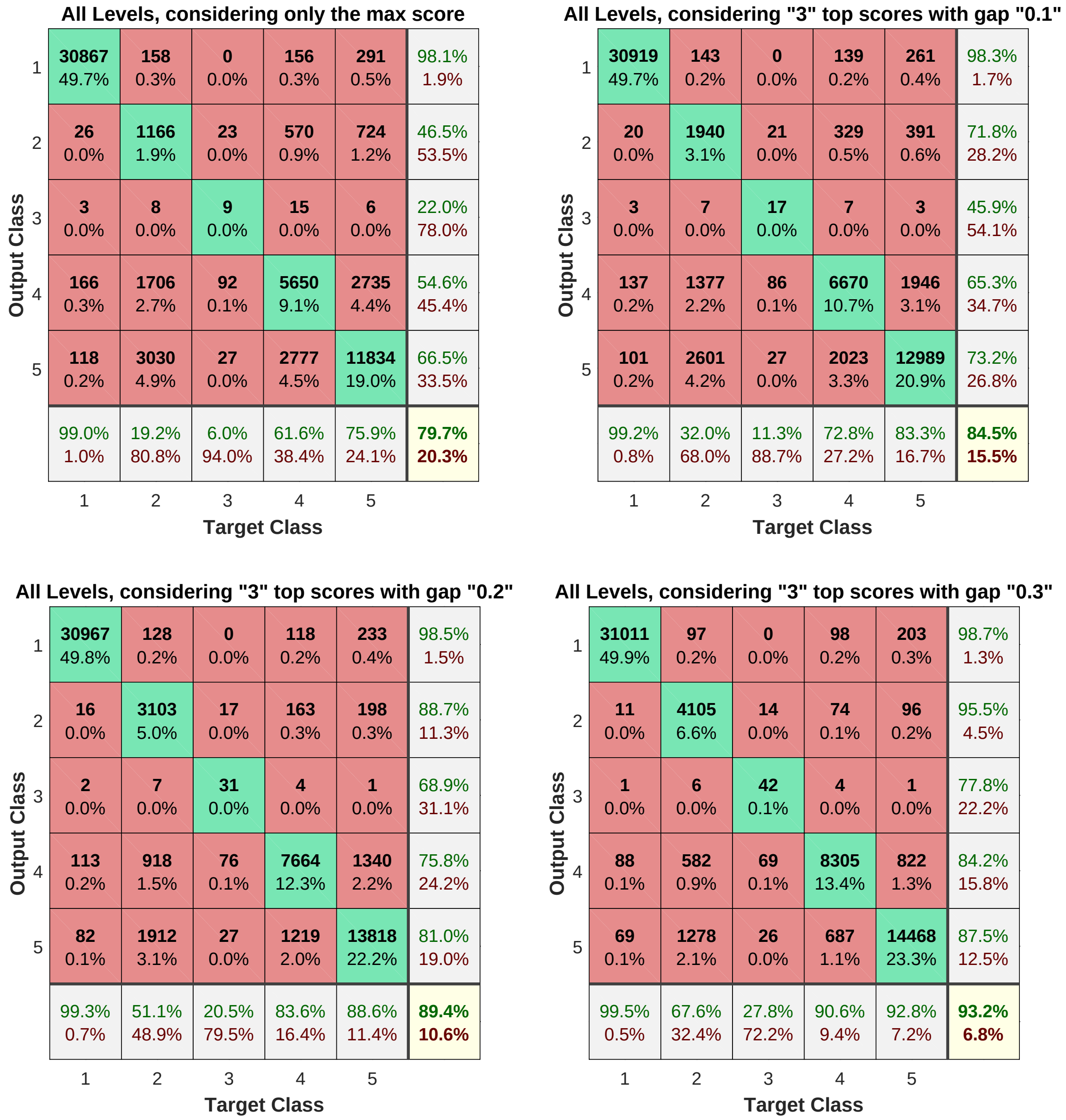
All Levels, considering "3" top scores with gap "0.2"

	1	2	3	4	5	
1	170403 49.5%	291 0.1%	57 0.0%	283 0.1%	422 0.1%	99.4% 0.6%
2	91 0.0%	24196 7.0%	201 0.1%	1195 0.3%	1632 0.5%	88.6% 11.4%
3	11 0.0%	45 0.0%	837 0.2%	76 0.0%	23 0.0%	84.4% 15.6%
4	159 0.0%	3027 0.9%	543 0.2%	40776 11.8%	3954 1.1%	84.1% 15.9%
5	166 0.0%	8870 2.6%	436 0.1%	7071 2.1%	79384 23.1%	82.8% 17.2%
	99.8% 0.2%	66.4% 33.6%	40.4% 59.6%	82.5% 17.5%	92.9% 7.1%	**91.7%** **8.3%**

All Levels, considering "3" top scores with gap "0.3"

	1	2	3	4	5	
1	170495 49.5%	210 0.1%	51 0.0%	223 0.1%	332 0.1%	99.5% 0.5%
2	71 0.0%	28841 8.4%	176 0.1%	597 0.2%	779 0.2%	94.7% 5.3%
3	11 0.0%	31 0.0%	1026 0.3%	49 0.0%	14 0.0%	90.7% 9.3%
4	126 0.0%	1757 0.5%	453 0.1%	44582 13.0%	2362 0.7%	90.5% 9.5%
5	127 0.0%	5590 1.6%	368 0.1%	3950 1.1%	81928 23.8%	89.1% 10.9%
	99.8% 0.2%	79.2% 20.8%	49.5% 50.5%	90.2% 9.8%	95.9% 4.1%	**95.0%** **5.0%**

Figure 5.5: CatNet confusion matrix on the “training” set for all levels of resolution.



All Levels, considering only the max score

Output Class	1	2	3	4	5	
1	30867 49.7%	158 0.3%	0 0.0%	156 0.3%	291 0.5%	98.1% 1.9%
2	26 0.0%	1166 1.9%	23 0.0%	570 0.9%	724 1.2%	46.5% 53.5%
3	3 0.0%	8 0.0%	9 0.0%	15 0.0%	6 0.0%	22.0% 78.0%
4	166 0.3%	1706 2.7%	92 0.1%	5650 9.1%	2735 4.4%	54.6% 45.4%
5	118 0.2%	3030 4.9%	27 0.0%	2777 4.5%	11834 19.0%	66.5% 33.5%
	99.0% 1.0%	19.2% 80.8%	6.0% 94.0%	61.6% 38.4%	75.9% 24.1%	**79.7%** **20.3%**

Target Class

All Levels, considering "3" top scores with gap "0.1"

Output Class	1	2	3	4	5	
1	30919 49.7%	143 0.2%	0 0.0%	139 0.2%	261 0.4%	98.3% 1.7%
2	20 0.0%	1940 3.1%	21 0.0%	329 0.5%	391 0.6%	71.8% 28.2%
3	3 0.0%	7 0.0%	17 0.0%	7 0.0%	3 0.0%	45.9% 54.1%
4	137 0.2%	1377 2.2%	86 0.1%	6670 10.7%	1946 3.1%	65.3% 34.7%
5	101 0.2%	2601 4.2%	27 0.0%	2023 3.3%	12989 20.9%	73.2% 26.8%
	99.2% 0.8%	32.0% 68.0%	11.3% 88.7%	72.8% 27.2%	83.3% 16.7%	**84.5%** **15.5%**

Target Class

All Levels, considering "3" top scores with gap "0.2"

Output Class	1	2	3	4	5	
1	30967 49.8%	128 0.2%	0 0.0%	118 0.2%	233 0.4%	98.5% 1.5%
2	16 0.0%	3103 5.0%	17 0.0%	163 0.3%	198 0.3%	88.7% 11.3%
3	2 0.0%	7 0.0%	31 0.0%	4 0.0%	1 0.0%	68.9% 31.1%
4	113 0.2%	918 1.5%	76 0.1%	7664 12.3%	1340 2.2%	75.8% 24.2%
5	82 0.1%	1912 3.1%	27 0.0%	1219 2.0%	13818 22.2%	81.0% 19.0%
	99.3% 0.7%	51.1% 48.9%	20.5% 79.5%	83.6% 16.4%	88.6% 11.4%	**89.4%** **10.6%**

Target Class

All Levels, considering "3" top scores with gap "0.3"

Output Class	1	2	3	4	5	
1	31011 49.9%	97 0.2%	0 0.0%	98 0.2%	203 0.3%	98.7% 1.3%
2	11 0.0%	4105 6.6%	14 0.0%	74 0.1%	96 0.2%	95.5% 4.5%
3	1 0.0%	6 0.0%	42 0.1%	4 0.0%	1 0.0%	77.8% 22.2%
4	88 0.1%	582 0.9%	69 0.1%	8305 13.4%	822 1.3%	84.2% 15.8%
5	69 0.1%	1278 2.1%	26 0.0%	687 1.1%	14468 23.3%	87.5% 12.5%
	99.5% 0.5%	67.6% 32.4%	27.8% 72.2%	90.6% 9.4%	92.8% 7.2%	**93.2%** **6.8%**

Target Class

Figure 5.6: CatNet confusion matrix on the “test” set for all levels of resolution.

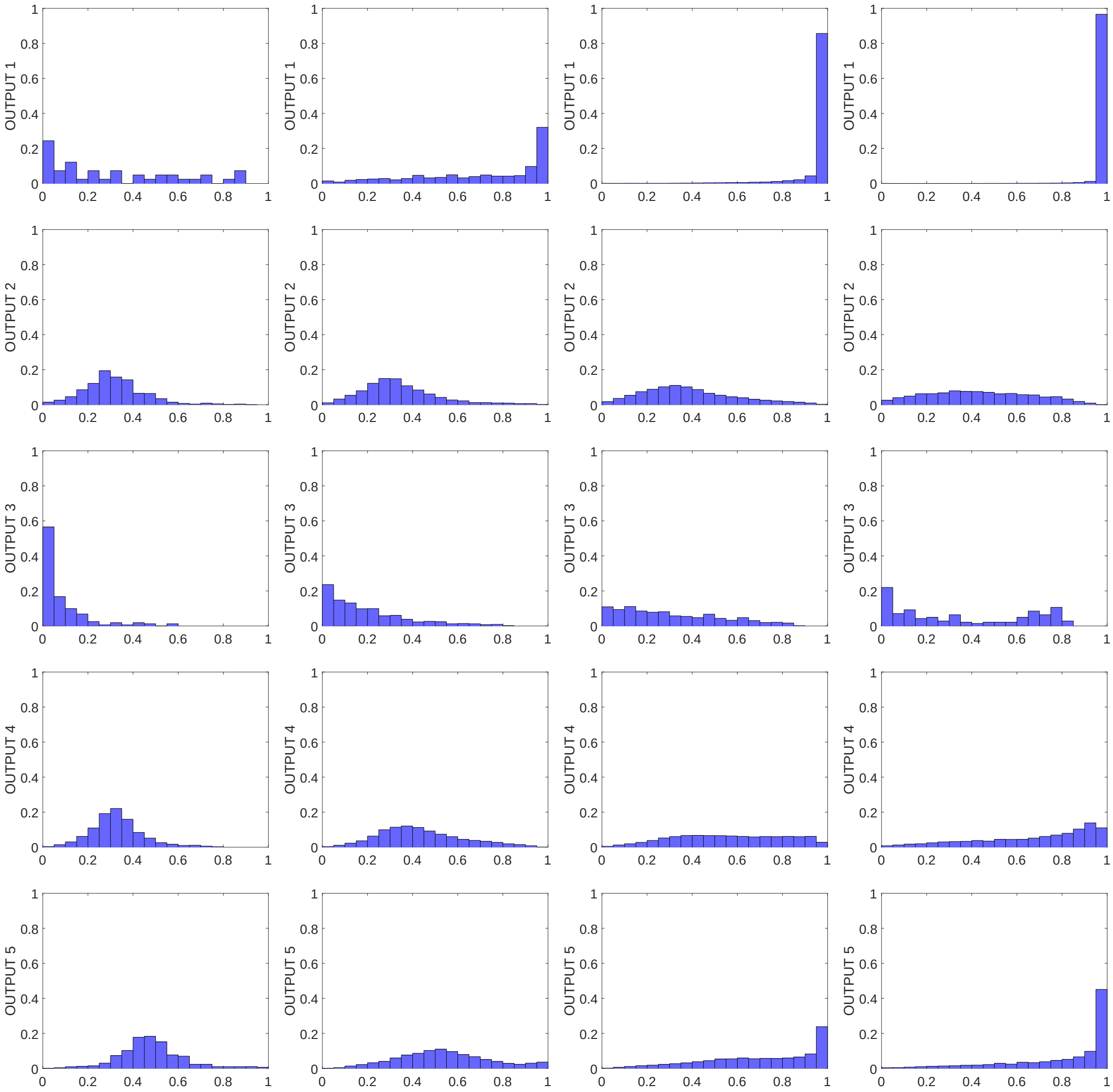
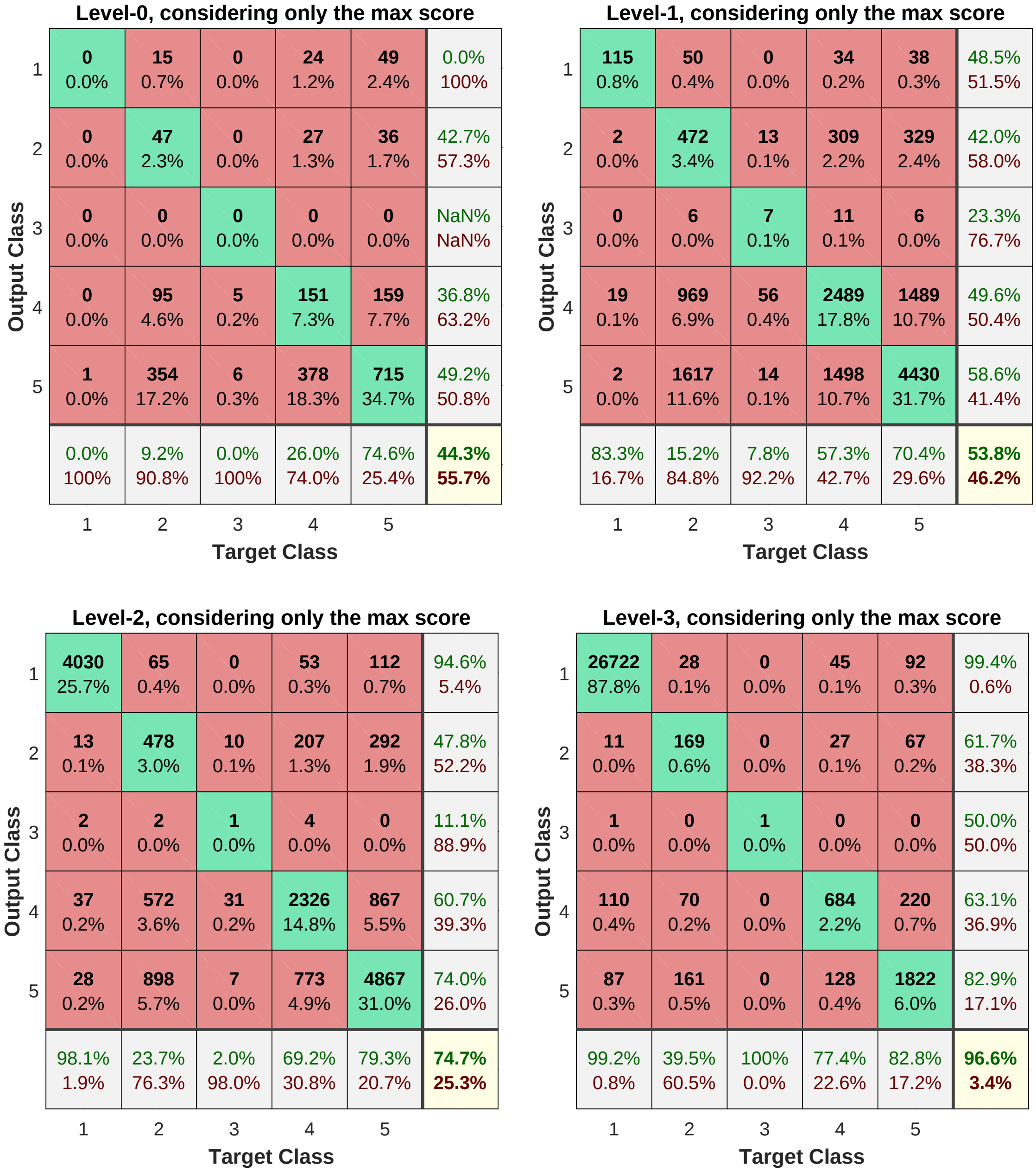


Figure 5.7: The distribution of CatNet output for different levels of resolution that gets finer from left to right. The i row distributions correspond to the i-th output when the net is fed with a patch from class i.



Level-0, considering only the max score

Output Class	1	2	3	4	5	
1	0 0.0%	15 0.7%	0 0.0%	24 1.2%	49 2.4%	0.0% 100%
2	0 0.0%	47 2.3%	0 0.0%	27 1.3%	36 1.7%	42.7% 57.3%
3	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
4	0 0.0%	95 4.6%	5 0.2%	151 7.3%	159 7.7%	36.8% 63.2%
5	1 0.0%	354 17.2%	6 0.3%	378 18.3%	715 34.7%	49.2% 50.8%
	0.0% 100%	9.2% 90.8%	0.0% 100%	26.0% 74.0%	74.6% 25.4%	44.3% 55.7%

Target Class

Level-1, considering only the max score

Output Class	1	2	3	4	5	
1	115 0.8%	50 0.4%	0 0.0%	34 0.2%	38 0.3%	48.5% 51.5%
2	2 0.0%	472 3.4%	13 0.1%	309 2.2%	329 2.4%	42.0% 58.0%
3	0 0.0%	6 0.0%	7 0.1%	11 0.1%	6 0.0%	23.3% 76.7%
4	19 0.1%	969 6.9%	56 0.4%	2489 17.8%	1489 10.7%	49.6% 50.4%
5	2 0.0%	1617 11.6%	14 0.1%	1498 10.7%	4430 31.7%	58.6% 41.4%
	83.3% 16.7%	15.2% 84.8%	7.8% 92.2%	57.3% 42.7%	70.4% 29.6%	53.8% 46.2%

Target Class

Level-2, considering only the max score

Output Class	1	2	3	4	5	
1	4030 25.7%	65 0.4%	0 0.0%	53 0.3%	112 0.7%	94.6% 5.4%
2	13 0.1%	478 3.0%	10 0.1%	207 1.3%	292 1.9%	47.8% 52.2%
3	2 0.0%	2 0.0%	1 0.0%	4 0.0%	0 0.0%	11.1% 88.9%
4	37 0.2%	572 3.6%	31 0.2%	2326 14.8%	867 5.5%	60.7% 39.3%
5	28 0.2%	898 5.7%	7 0.0%	773 4.9%	4867 31.0%	74.0% 26.0%
	98.1% 1.9%	23.7% 76.3%	2.0% 98.0%	69.2% 30.8%	79.3% 20.7%	74.7% 25.3%

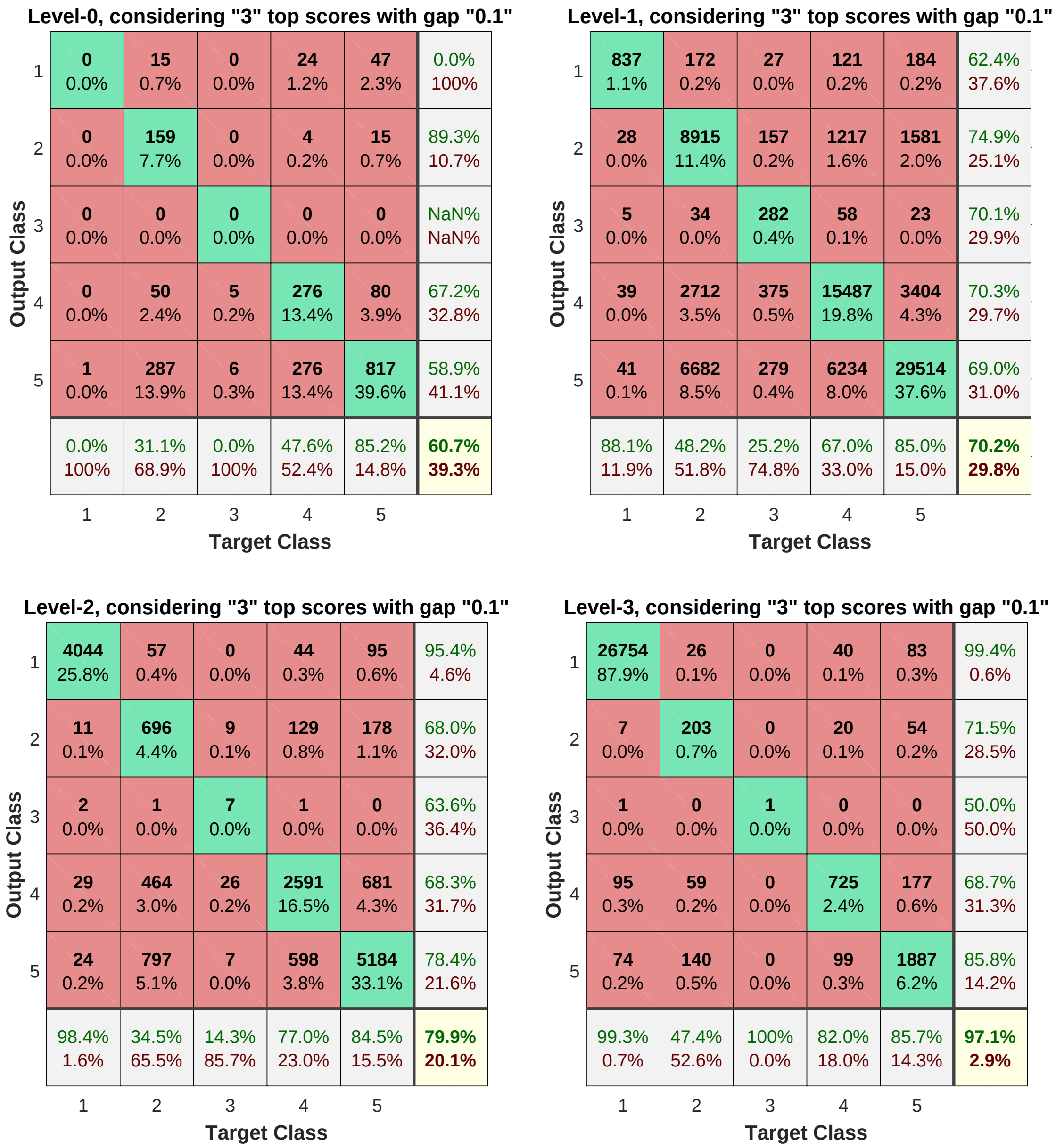
Target Class

Level-3, considering only the max score

Output Class	1	2	3	4	5	
1	26722 87.8%	28 0.1%	0 0.0%	45 0.1%	92 0.3%	99.4% 0.6%
2	11 0.0%	169 0.6%	0 0.0%	27 0.1%	67 0.2%	61.7% 38.3%
3	1 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	50.0% 50.0%
4	110 0.4%	70 0.2%	0 0.0%	684 2.2%	220 0.7%	63.1% 36.9%
5	87 0.3%	161 0.5%	0 0.0%	128 0.4%	1822 6.0%	82.9% 17.1%
	99.2% 0.8%	39.5% 60.5%	100% 0.0%	77.4% 22.6%	82.8% 17.2%	96.6% 3.4%

Target Class

Figure 5.8: CatNet confusion matrix on the “test” set broken down for different levels of resolution when classification is only based on the top score.



Level-0, considering "3" top scores with gap "0.1"

	1	2	3	4	5	
1	0 0.0%	15 0.7%	0 0.0%	24 1.2%	47 2.3%	0.0% 100%
2	0 0.0%	159 7.7%	0 0.0%	4 0.2%	15 0.7%	89.3% 10.7%
3	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
4	0 0.0%	50 2.4%	5 0.2%	276 13.4%	80 3.9%	67.2% 32.8%
5	1 0.0%	287 13.9%	6 0.3%	276 13.4%	817 39.6%	58.9% 41.1%
	0.0% 100%	31.1% 68.9%	0.0% 100%	47.6% 52.4%	85.2% 14.8%	60.7% 39.3%

Output Class

Target Class

Level-1, considering "3" top scores with gap "0.1"

	1	2	3	4	5	
1	837 1.1%	172 0.2%	27 0.0%	121 0.2%	184 0.2%	62.4% 37.6%
2	28 0.0%	8915 11.4%	157 0.2%	1217 1.6%	1581 2.0%	74.9% 25.1%
3	5 0.0%	34 0.0%	282 0.4%	58 0.1%	23 0.0%	70.1% 29.9%
4	39 0.0%	2712 3.5%	375 0.5%	15487 19.8%	3404 4.3%	70.3% 29.7%
5	41 0.1%	6682 8.5%	279 0.4%	6234 8.0%	29514 37.6%	69.0% 31.0%
	88.1% 11.9%	48.2% 51.8%	25.2% 74.8%	67.0% 33.0%	85.0% 15.0%	70.2% 29.8%

Output Class

Target Class

Level-2, considering "3" top scores with gap "0.1"

	1	2	3	4	5	
1	4044 25.8%	57 0.4%	0 0.0%	44 0.3%	95 0.6%	95.4% 4.6%
2	11 0.1%	696 4.4%	9 0.1%	129 0.8%	178 1.1%	68.0% 32.0%
3	2 0.0%	1 0.0%	7 0.0%	1 0.0%	0 0.0%	63.6% 36.4%
4	29 0.2%	464 3.0%	26 0.2%	2591 16.5%	681 4.3%	68.3% 31.7%
5	24 0.2%	797 5.1%	7 0.0%	598 3.8%	5184 33.1%	78.4% 21.6%
	98.4% 1.6%	34.5% 65.5%	14.3% 85.7%	77.0% 23.0%	84.5% 15.5%	79.9% 20.1%

Output Class

Target Class

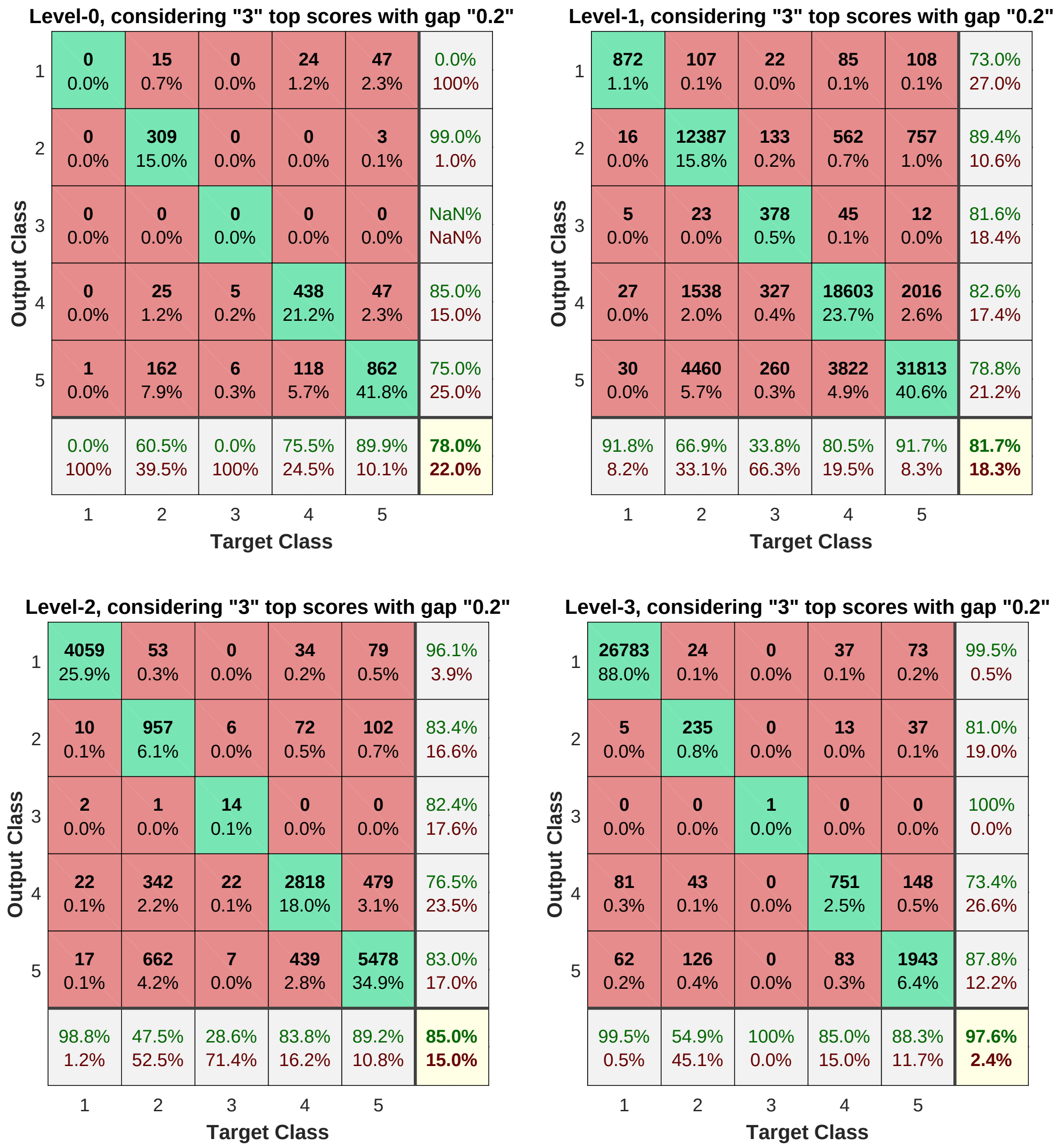
Level-3, considering "3" top scores with gap "0.1"

	1	2	3	4	5	
1	26754 87.9%	26 0.1%	0 0.0%	40 0.1%	83 0.3%	99.4% 0.6%
2	7 0.0%	203 0.7%	0 0.0%	20 0.1%	54 0.2%	71.5% 28.5%
3	1 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	50.0% 50.0%
4	95 0.3%	59 0.2%	0 0.0%	725 2.4%	177 0.6%	68.7% 31.3%
5	74 0.2%	140 0.5%	0 0.0%	99 0.3%	1887 6.2%	85.8% 14.2%
	99.3% 0.7%	47.4% 52.6%	100% 0.0%	82.0% 18.0%	85.7% 14.3%	97.1% 2.9%

Output Class

Target Class

Figure 5.9: CatNet confusion matrix on the "test" set broken down for different levels of resolution when classification is based on the top-3 scores and score gap 0.1.



Level-0, considering "3" top scores with gap "0.2"

Output Class \ Target Class	1	2	3	4	5	
1	0 0.0%	15 0.7%	0 0.0%	24 1.2%	47 2.3%	0.0% 100%
2	0 0.0%	309 15.0%	0 0.0%	0 0.0%	3 0.1%	99.0% 1.0%
3	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
4	0 0.0%	25 1.2%	5 0.2%	438 21.2%	47 2.3%	85.0% 15.0%
5	1 0.0%	162 7.9%	6 0.3%	118 5.7%	862 41.8%	75.0% 25.0%
	0.0% 100%	60.5% 39.5%	0.0% 100%	75.5% 24.5%	89.9% 10.1%	**78.0%** **22.0%**

Level-1, considering "3" top scores with gap "0.2"

Output Class \ Target Class	1	2	3	4	5	
1	872 1.1%	107 0.1%	22 0.0%	85 0.1%	108 0.1%	73.0% 27.0%
2	16 0.0%	12387 15.8%	133 0.2%	562 0.7%	757 1.0%	89.4% 10.6%
3	5 0.0%	23 0.0%	378 0.5%	45 0.1%	12 0.0%	81.6% 18.4%
4	27 0.0%	1538 2.0%	327 0.4%	18603 23.7%	2016 2.6%	82.6% 17.4%
5	30 0.0%	4460 5.7%	260 0.3%	3822 4.9%	31813 40.6%	78.8% 21.2%
	91.8% 8.2%	66.9% 33.1%	33.8% 66.3%	80.5% 19.5%	91.7% 8.3%	**81.7%** **18.3%**

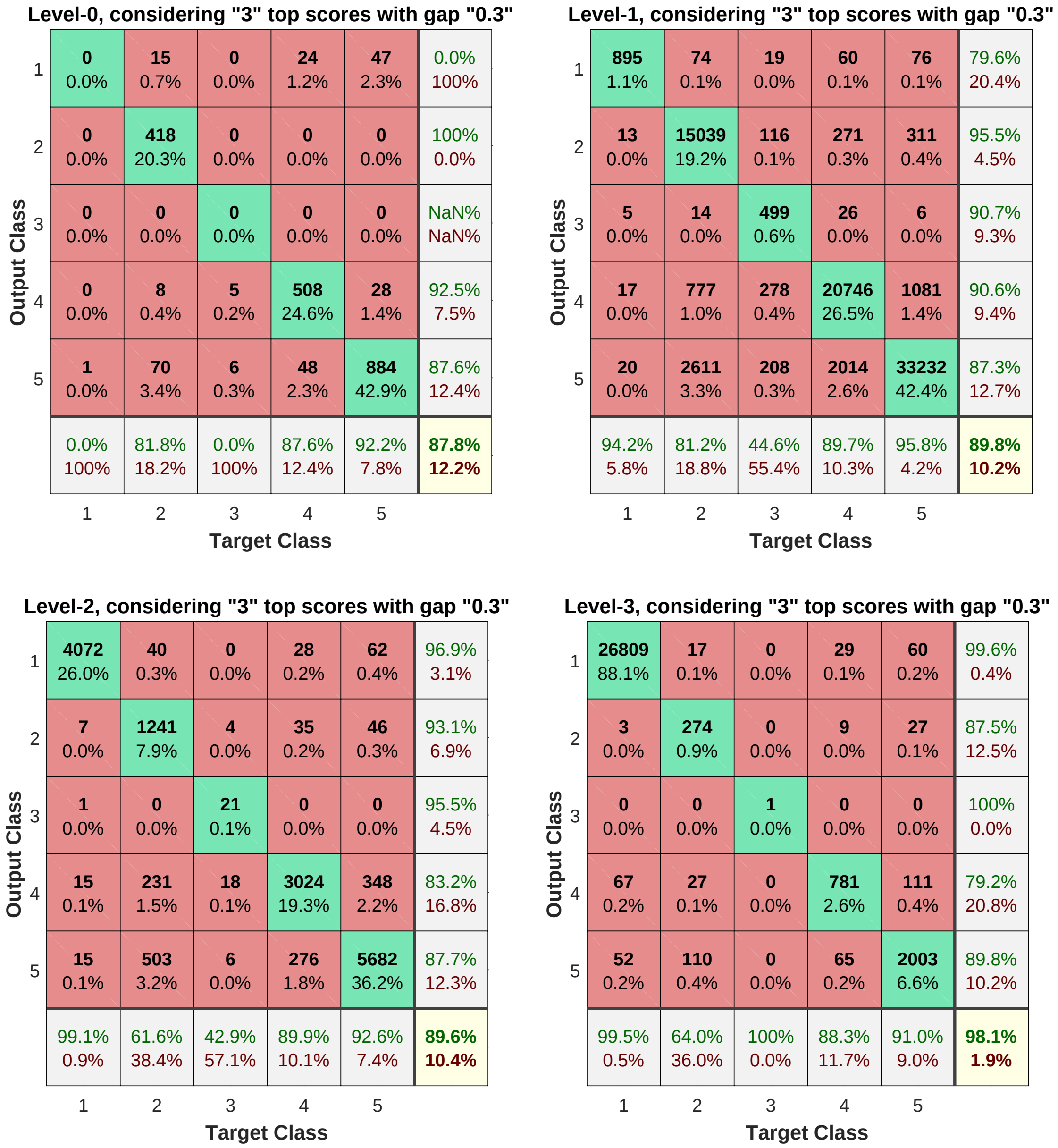
Level-2, considering "3" top scores with gap "0.2"

Output Class \ Target Class	1	2	3	4	5	
1	4059 25.9%	53 0.3%	0 0.0%	34 0.2%	79 0.5%	96.1% 3.9%
2	10 0.1%	957 6.1%	6 0.0%	72 0.5%	102 0.7%	83.4% 16.6%
3	2 0.0%	1 0.0%	14 0.1%	0 0.0%	0 0.0%	82.4% 17.6%
4	22 0.1%	342 2.2%	22 0.1%	2818 18.0%	479 3.1%	76.5% 23.5%
5	17 0.1%	662 4.2%	7 0.0%	439 2.8%	5478 34.9%	83.0% 17.0%
	98.8% 1.2%	47.5% 52.5%	28.6% 71.4%	83.8% 16.2%	89.2% 10.8%	**85.0%** **15.0%**

Level-3, considering "3" top scores with gap "0.2"

Output Class \ Target Class	1	2	3	4	5	
1	26783 88.0%	24 0.1%	0 0.0%	37 0.1%	73 0.2%	99.5% 0.5%
2	5 0.0%	235 0.8%	0 0.0%	13 0.0%	37 0.1%	81.0% 19.0%
3	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	81 0.3%	43 0.1%	0 0.0%	751 2.5%	148 0.5%	73.4% 26.6%
5	62 0.2%	126 0.4%	0 0.0%	83 0.3%	1943 6.4%	87.8% 12.2%
	99.5% 0.5%	54.9% 45.1%	100% 0.0%	85.0% 15.0%	88.3% 11.7%	**97.6%** **2.4%**

Figure 5.10: CatNet confusion matrix on the “test” set broken down for different levels of resolution when classification is based on the top-3 scores and score gap 0.2.



Level-0, considering "3" top scores with gap "0.3"

Output Class \ Target Class	1	2	3	4	5	
1	0 0.0%	15 0.7%	0 0.0%	24 1.2%	47 2.3%	0.0% 100%
2	0 0.0%	418 20.3%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
4	0 0.0%	8 0.4%	5 0.2%	508 24.6%	28 1.4%	92.5% 7.5%
5	1 0.0%	70 3.4%	6 0.3%	48 2.3%	884 42.9%	87.6% 12.4%
	0.0% 100%	81.8% 18.2%	0.0% 100%	87.6% 12.4%	92.2% 7.8%	**87.8%** **12.2%**

Level-1, considering "3" top scores with gap "0.3"

Output Class \ Target Class	1	2	3	4	5	
1	895 1.1%	74 0.1%	19 0.0%	60 0.1%	76 0.1%	79.6% 20.4%
2	13 0.0%	15039 19.2%	116 0.1%	271 0.3%	311 0.4%	95.5% 4.5%
3	5 0.0%	14 0.0%	499 0.6%	26 0.0%	6 0.0%	90.7% 9.3%
4	17 0.0%	777 1.0%	278 0.4%	20746 26.5%	1081 1.4%	90.6% 9.4%
5	20 0.0%	2611 3.3%	208 0.3%	2014 2.6%	33232 42.4%	87.3% 12.7%
	94.2% 5.8%	81.2% 18.8%	44.6% 55.4%	89.7% 10.3%	95.8% 4.2%	**89.8%** **10.2%**

Level-2, considering "3" top scores with gap "0.3"

Output Class \ Target Class	1	2	3	4	5	
1	4072 26.0%	40 0.3%	0 0.0%	28 0.2%	62 0.4%	96.9% 3.1%
2	7 0.0%	1241 7.9%	4 0.0%	35 0.2%	46 0.3%	93.1% 6.9%
3	1 0.0%	0 0.0%	21 0.1%	0 0.0%	0 0.0%	95.5% 4.5%
4	15 0.1%	231 1.5%	18 0.1%	3024 19.3%	348 2.2%	83.2% 16.8%
5	15 0.1%	503 3.2%	6 0.0%	276 1.8%	5682 36.2%	87.7% 12.3%
	99.1% 0.9%	61.6% 38.4%	42.9% 57.1%	89.9% 10.1%	92.6% 7.4%	**89.6%** **10.4%**

Level-3, considering "3" top scores with gap "0.3"

Output Class \ Target Class	1	2	3	4	5	
1	26809 88.1%	17 0.1%	0 0.0%	29 0.1%	60 0.2%	99.6% 0.4%
2	3 0.0%	274 0.9%	0 0.0%	9 0.0%	27 0.1%	87.5% 12.5%
3	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	67 0.2%	27 0.1%	0 0.0%	781 2.6%	111 0.4%	79.2% 20.8%
5	52 0.2%	110 0.4%	0 0.0%	65 0.2%	2003 6.6%	89.8% 10.2%
	99.5% 0.5%	64.0% 36.0%	100% 0.0%	88.3% 11.7%	91.0% 9.0%	**98.1%** **1.9%**

Figure 5.11: CatNet confusion matrix on the “test” set broken down for different levels of resolution when classification is based on the top-3 scores and score gap 0.3.

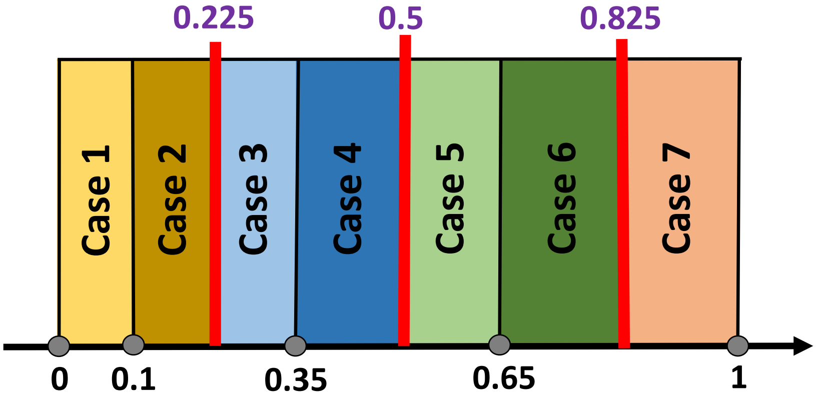


Figure 5.12: Scale ratio intervals.

5.1.2 ScaleNet

We designed the ScaleNet to estimate the scale of detected object instances independent of their category. Since every patch has to be resized to a fixed 224×224 size before being processed by ScaleNet, the estimated scale of an object based on the fixed size input patch has to be scaled appropriately to estimate the scale of the object in the original patch before resizing. Obviously, the ratio of an object's scale (in pixels) to the size of the patch will stay unchanged after resizing. For an object that is fully inside a patch the scale ratio is within the range $(0, 1]$. Note that we assume that the scale of an object instance is the longest side of its enclosing rectangle whose sides are horizontal/vertical. Hence, the scale ratio is always smaller than 1 for an object completely inside a patch.

We changed the number of outputs of the last fully-connected layer as well as the softmax layer of the VGG-16 network to 4, where each output class represents a scale-to-patch-size ratio. The four outputs of ScaleNet correspond to ratios 0.1, 0.35, 0.65,

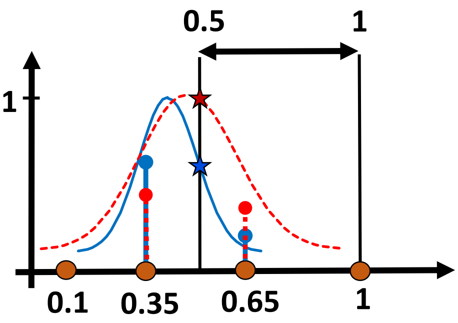


Figure 5.13: Two example scale score functions calculated based on the top-2 ScaleNet scores. The scale scores for the two score function are shown by stars.

and 1. This is basically a quantization of the scale ratio range $(0, 1]$ to four centroids and Voronoi intervals (see Figure 5.12 where Voronoi intervals are separated by the vertical red lines). The centroids were selected with increasing scale gaps from 0.1 to 1: $(0.35 - 0.1 = 0.25) < (0.65 - 0.35 = 0.3) < (1.0 - 0.65 = 0.35)$. Choosing an increasing score gap is because we can tolerate larger errors in larger scales.

We trained ScaleNet on a training set of 171,395 patches. Each patch was labeled by one label $l \in \{1, 2, 3, 4\}$, with 42,567 patches whose scale ratio was quantized to the first class (0.1 scale ratio), 82,509 patches whose scale ratio was quantized to the second class (0.35 scale ratio), 37,443 patches whose scale ratio was quantized to the third class (0.65 scale ratio), and 8,876 patches whose scale ratio was quantized to the fourth class (1 scale ratio). We evaluated ScaleNet on a test set of 30,742 patches. The training took about 24 hours on GPU using SGD with momentum and a mini-batch size of 50 images.

Figure 5.14 shows confusion matrices for training and test sets in two cases of classification based on the maximum score class and top-2 score classes. A match is declared in the case of top-2 score classification if the true class is among the top two scores. It can be seen that the most common mistakes are made in the classification of consecutive classes which makes sense since consecutive classes are associated with consecutive scale ratios which have closer output distributions. Figure 5.15 shows the distribution of ScaleNet's 4 outputs on both the training and test set, where the j-th column corresponds to the j-th output of ScaleNet and the i-th row corresponds to the i-th scale ratio case shown in Figure 5.12 and summarized below:

Case 1: The input is a patch whose true scale ratio is from interval $(0, 0.1]$.
Case 2: The input is a patch whose true scale ratio is from interval $(0.1, 0.225]$.
Case 3: The input is a patch whose true scale ratio is from interval $(0.225, 0.35]$.
Case 4: The input is a patch whose true scale ratio is from interval $(0.35, 0.5]$.
Case 5: The input is a patch whose true scale ratio is from interval $(0.5, 0.65]$.
Case 6: The input is a patch whose true scale ratio is from interval $(0.65, 0.825]$.
Case 7: The input is a patch whose true scale ratio is from interval $(0.825, 1]$.

It can be seen that the distribution of the ScaleNet outputs are close for the training and test sets; and the distribution at higher values is denser on the j-th output if the input to ScaleNet is from corresponding class j. The output distribution gradually changes from the first to the seventh case listed above. Note that two consecutive distributions look closer compared to two non-consecutive distributions. It can be seen that the distributions of cases associated with the same class but scale ratio of smaller and larger than centroid look slightly different in the sense that the later is closer to the distribution of the next class.

Since the top-2 classification has high accuracy (overall 98.1% accuracy on the training set and 95.3% on the test set) we were convinced to use only the top two scores for estimating the scale ratio of detections. The estimated scale ratio is a weighted average based on the top two scores:

$$\hat{SR} = \frac{\mathbf{s}\mathbf{D}\,[0.1, 0.35, 0.65, 1]^\top}{\mathbf{s}\mathbf{D}\,[1, 1, 1, 1]^\top}, \tag{5.2}$$

where "s" is a 4-dimensional row vector denoting the ScaleNet's output and $\mathbf{D}$ is a 4×4 diagonal matrix whose diagonal elements are 1 at locations corresponding to the top two scores and zero elsewhere. We define normal-shaped *score function* centered at the estimated scale ratio $\hat{SR}$:

$$f(t) = \exp\bigl(-\frac{(t - \hat{SR})^2}{2\sigma^2}\bigr), \tag{5.3}$$

where: $\sigma^2 = (A\mathbf{D}A^\top)/(\mathbf{s}\mathbf{D}[1, 1, 1, 1]^\top)$, for $A = [0.1, 0.35, 0.65, 1] - \hat{SR}$. The σ value gets smaller as the score gap between the top two scores widens. The score function will be densely concentrated around the estimated scale ratio for a more confident scale estimation, namely when the top score is much larger that the rest of scores. This becomes more clear after defining the scale score in the following paragraph.

An object instance can be located completely inside multiple annocells from different levels of resolution due to the coarse-to-fine structure of the annocell hierarchy. However, we would like to declare the annocell with smallest size that contains the object (the fittest annocell) as the bounding box for every detection. Hence, we should give a higher score to a detection whose estimated scale ratio is larger than 0.5, *i.e.,* the detected object occupies more than half of the patch size. We define the scale score by $S_{\text{scale}} = \max_{0.5 \leq t \leq 1} f(t)$. Figure 5.13 illustrates two example score functions in “blue” and “dashed–red” based on the top-2 scores shown by the same color bins; the blue score function rolls off faster from its maximum due to the larger discrepancy between the top-2 scores. The scale scores for the two score functions are shown by stars. The estimated scale ratio corresponding to the blue curve is farther away from the $[0.5, 1]$ interval with higher confidence compared to the dashed–red curve; therefore, the blue curve leads to a smaller scale score. The smaller dispersion parameter σ for the blue curve resulted in a considerable gap between the two scores; the larger gap is desirable since the ScaleNet scores shown by the red bins are more likely to correspond to a fitter annocell.

Finally, each patch from the annocell hierarchy is given a mixed “Category–Scale” score per category. The mixed score for a given patch with scale score S_{scale} and the c-th category score s_c is $S^c_{\text{mixed}} = s_c \times S_{\text{scale}}$, where s_c is the corresponding output from the CatNet. We declare an annocell patch as positive detection (bounding box) for the c-th category if both $S_{\text{scale}} \geq 0.5$ and s_c is among the CatNet’s top-3 scores with score gap $S_g = 0.3$. We perform “non-maximum suppression” on the mixed scores of the positive detections per category to obtain a sparse set of boxes. Non-maximum suppression is performed by picking the most confident (maximum score) detection and removing its neighboring detections; then, picking the second most confident detection left and removing its neighbors, and continuing this process until there are no positive detections left. We consider two patches to be neighbors if at least 50% of the smaller patch is covered by (overlaps with) the bigger patch. This neighborhood definition results in the removal of positive detections across different resolution levels of the annocell hierarchy during non-maximum suppression. Figures 6.16, 6.19, 6.22, 6.25, 6.28 show detections per category after non-maximum suppression for some table-setting scenes.

State-of-the-art object detection systems (*e.g.,* see [30]) use the "selective search" algorithm [99] to propose boxes which are likely to contain object instances; these boxes are then processed by a convolutional neural networks for classification, and regressed to obtain the bounding boxes for the positive detections. The selective search algorithm generates candidates by various ways of grouping the output of an initial image segmentation. Note that we did not run any initial segmentation on the input image to obtain the bounding box estimates. The fast region-based CNN (fast R-CNN) [31] does not use the selective search algorithm to generate the candidate boxes; their network generates the bounding boxes internally in the forward path. The drawback of their network is that it does not use any contextual relations between different object instances and they do not account for the presence of multiple instances from the same category.

5.1.3 TableNet

The TableNet servers a binary classifier for predicting whether or not an input patch belongs to the table surface. Detecting the table surface limits the range of homography free parameters by constraining them to result in a table projection consistent (tightly fitted) with the detected table area. An accurate detection of the table surface as a pre-processing step results in significant speed-up during inference by avoiding inconsistent homographies. We trained TableNet on 270,410 training patches (153,812 background and 116,598 table), and tested it on 38,651 test patches (18,888 background and 19,763 table). The background and table patches are defined by having, respectively, at most 10% and at least 50% (of the patch) overlap with the table surface area. All of the training and test patches were selected from level-2 and level-3 of the annocell hierarchy. Figure 5.21 shows some training and test patches where the background patches are framed in black and table patches are framed in green. Figure 5.22 shows the confusion matrices for training and test sets for three cases (levels 2+3, 2, and 3). The overall accuracy on the training set is always higher than on the test set as we expect. The patches from the coarser level-2 of the annocell hierarchy are classified with higher accuracy compared with patches from the finer level-3, perhaps because TableNet has access to more texture information at level-2 compared to level-3, which results in more discriminative features and hence higher classification accuracy.

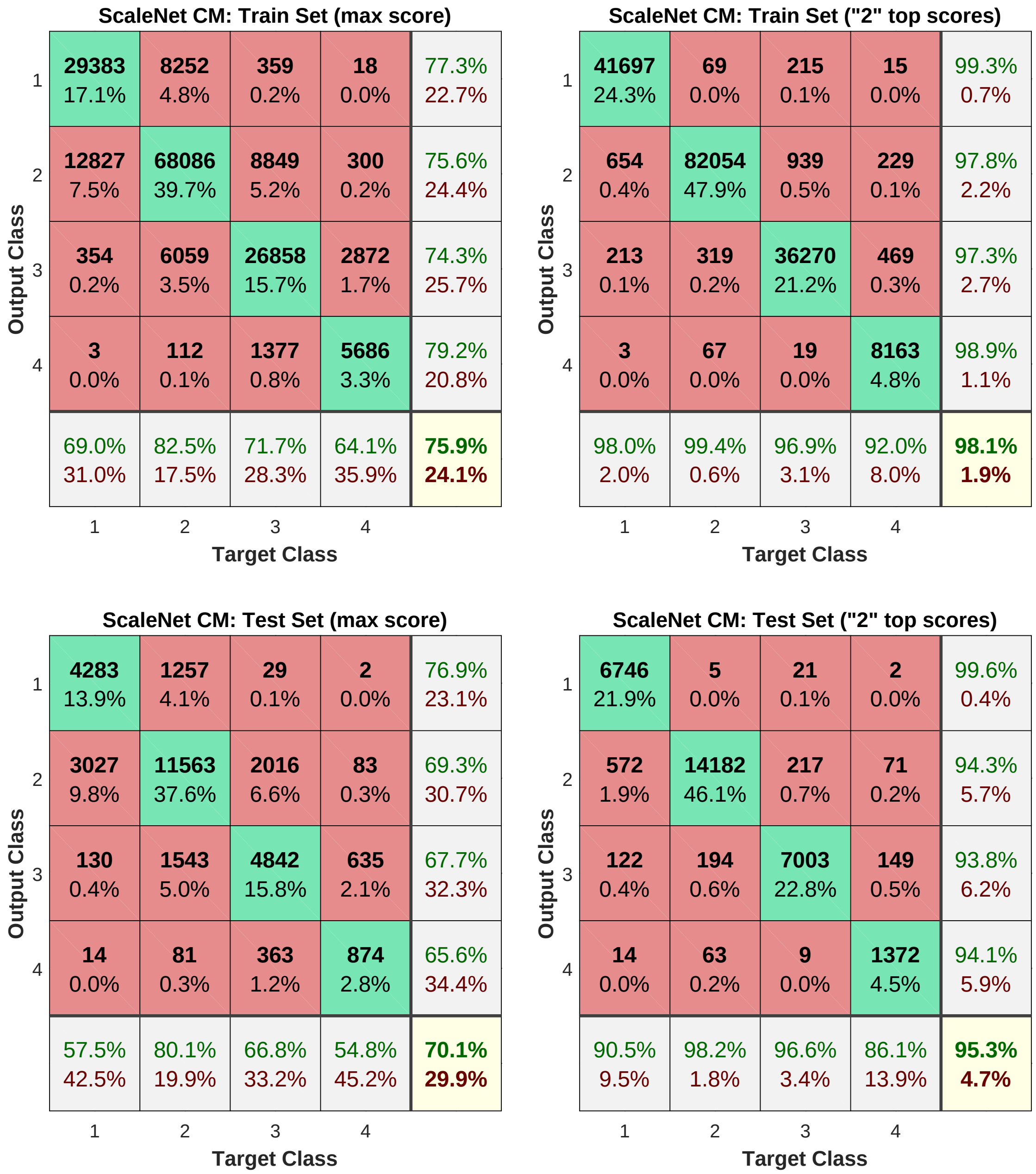
Given a test image we first run TableNet on patches from level-2 with zero overlap (16 patches) and for those classified as "table" we run TableNet on the contained non-overlapping patches from level-3 in order to get a finer localization of the table area. This results in running TableNet on at most 80 ($= 16 \times 4 + 16$) patches. We finally fit a convex shape, such as a polygon, rectangle, or ellipse, to the corner points of patches at level-3 which were classified as table. We use polygons in our experiments. In order to suppress the effect of mistakenly classified patches we fit the convex shape only to the connected region with the maximum number of patches classified as table. Figures 5.23, 5.24 show the estimated table area for some example images. Figure 5.25(top row) shows two examples where some off-table patches were classified as table, but due to the constraint on fitting, the mistakenly classified patches did not affect table detection. Figure 5.25(bottom row) shows two poor table detection examples which seem to happen due to the lack of enough texture on the tables. We ran our table detector on 284 images and observed fewer than 5 poor table detections.

We also estimated the table size by appropriately scaling the length of the longest diameter of the fitting polygon. That appropriate scale was calculated by running ScaleNet on patches from level–2 classified as table, and assuming that the table-setting objects have a 20 cm size on the average. Knowing the size of an object in the real world (in meters) and its scale in the image coordinate system (in pixels) provides a simple way to convert pixels to meters. Figure 5.26 shows the histogram of the absolute and relative errors made by our table size estimator. The histogram is centered roughly around 0 meaning that our table size estimator is relatively unbiased. We calculated the true table size by back-projecting the annotated table surface using the visually estimated homography for every image.

To generate homography samples that conform with the detected table area, assume a rectangular table of size $T = (L, W)$ whose four corner points are $(-L/2, -W/2)$, $(-L/2, W/2)$, $(L/2, -W/2)$, and $(L/2, W/2)$;. We draw samples from the distribution on camera parameters $p(\mathcal{W})$ proposed in section 3.5 and calculate the corresponding homography matrix according to Appendix A. Then, we project the four corners of the table to the image coordinate system using this homography matrix and check if the resulting polygon (quadrilateral) fits well to the detected table area using a similarity measure for 2D–shapes. We declare a "good fit" between two shapes A_1 and A_2 if:

$$d(A_1, A_2) = |(A_1 \cup A_2) - (A_1 \cap A_2)| < 0.25 \min(|A_1|, |A_2|). \tag{5.4}$$

In an attempt to efficiently sample the homography (camera parameter) distribution that is consistent with the detected table area, we first try to find a set of camera parameters that result in a table projection meeting a relaxation of (5.4), namely $d(A_1, A_2) < 0.4 \min(|A_1|, |A_2|)$, and as soon as we find such a sample we start to greedily fine–tune the camera parameters to finally satisfy (5.4). During fine-tuning we randomly choose one camera parameter and change it slightly by sampling a normal distribution with small variance centered at the previous value; we accept this change if it resulted in a smaller distance $d(A_1, A_2)$. We try a total of $10,000$ homographies obtained by sampling the camera model $p(\mathcal{W})$ (to satisfy the relaxed condition) or fine-tuning of parameters $\mathcal{W}$ (to satisfy (5.4)) and exit the loop as soon as (5.4) is met; otherwise, if the condition (5.4) was not met during $10,000$ trials, we output the camera parameters resulting in the minimum $d(A_1, A_2)$. Figures 5.27, 5.28, 5.29 show some consistent homography samples for three images.



ScaleNet CM: Train Set (max score)

	1	2	3	4	
1	**29383** 17.1%	**8252** 4.8%	**359** 0.2%	**18** 0.0%	77.3% 22.7%
2	**12827** 7.5%	**68086** 39.7%	**8849** 5.2%	**300** 0.2%	75.6% 24.4%
3	**354** 0.2%	**6059** 3.5%	**26858** 15.7%	**2872** 1.7%	74.3% 25.7%
4	**3** 0.0%	**112** 0.1%	**1377** 0.8%	**5686** 3.3%	79.2% 20.8%
	69.0% 31.0%	82.5% 17.5%	71.7% 28.3%	64.1% 35.9%	**75.9%** **24.1%**

ScaleNet CM: Train Set ("2" top scores)

	1	2	3	4	
1	**41697** 24.3%	**69** 0.0%	**215** 0.1%	**15** 0.0%	99.3% 0.7%
2	**654** 0.4%	**82054** 47.9%	**939** 0.5%	**229** 0.1%	97.8% 2.2%
3	**213** 0.1%	**319** 0.2%	**36270** 21.2%	**469** 0.3%	97.3% 2.7%
4	**3** 0.0%	**67** 0.0%	**19** 0.0%	**8163** 4.8%	98.9% 1.1%
	98.0% 2.0%	99.4% 0.6%	96.9% 3.1%	92.0% 8.0%	**98.1%** **1.9%**

ScaleNet CM: Test Set (max score)

	1	2	3	4	
1	**4283** 13.9%	**1257** 4.1%	**29** 0.1%	**2** 0.0%	76.9% 23.1%
2	**3027** 9.8%	**11563** 37.6%	**2016** 6.6%	**83** 0.3%	69.3% 30.7%
3	**130** 0.4%	**1543** 5.0%	**4842** 15.8%	**635** 2.1%	67.7% 32.3%
4	**14** 0.0%	**81** 0.3%	**363** 1.2%	**874** 2.8%	65.6% 34.4%
	57.5% 42.5%	80.1% 19.9%	66.8% 33.2%	54.8% 45.2%	**70.1%** **29.9%**

ScaleNet CM: Test Set ("2" top scores)

	1	2	3	4	
1	**6746** 21.9%	**5** 0.0%	**21** 0.1%	**2** 0.0%	99.6% 0.4%
2	**572** 1.9%	**14182** 46.1%	**217** 0.7%	**71** 0.2%	94.3% 5.7%
3	**122** 0.4%	**194** 0.6%	**7003** 22.8%	**149** 0.5%	93.8% 6.2%
4	**14** 0.0%	**63** 0.2%	**9** 0.0%	**1372** 4.5%	94.1% 5.9%
	90.5% 9.5%	98.2% 1.8%	96.6% 3.4%	86.1% 13.9%	**95.3%** **4.7%**

Figure 5.14: ScaleNet confusion matrix on “training” and “test” set considering both max score classification and top-2 classification.

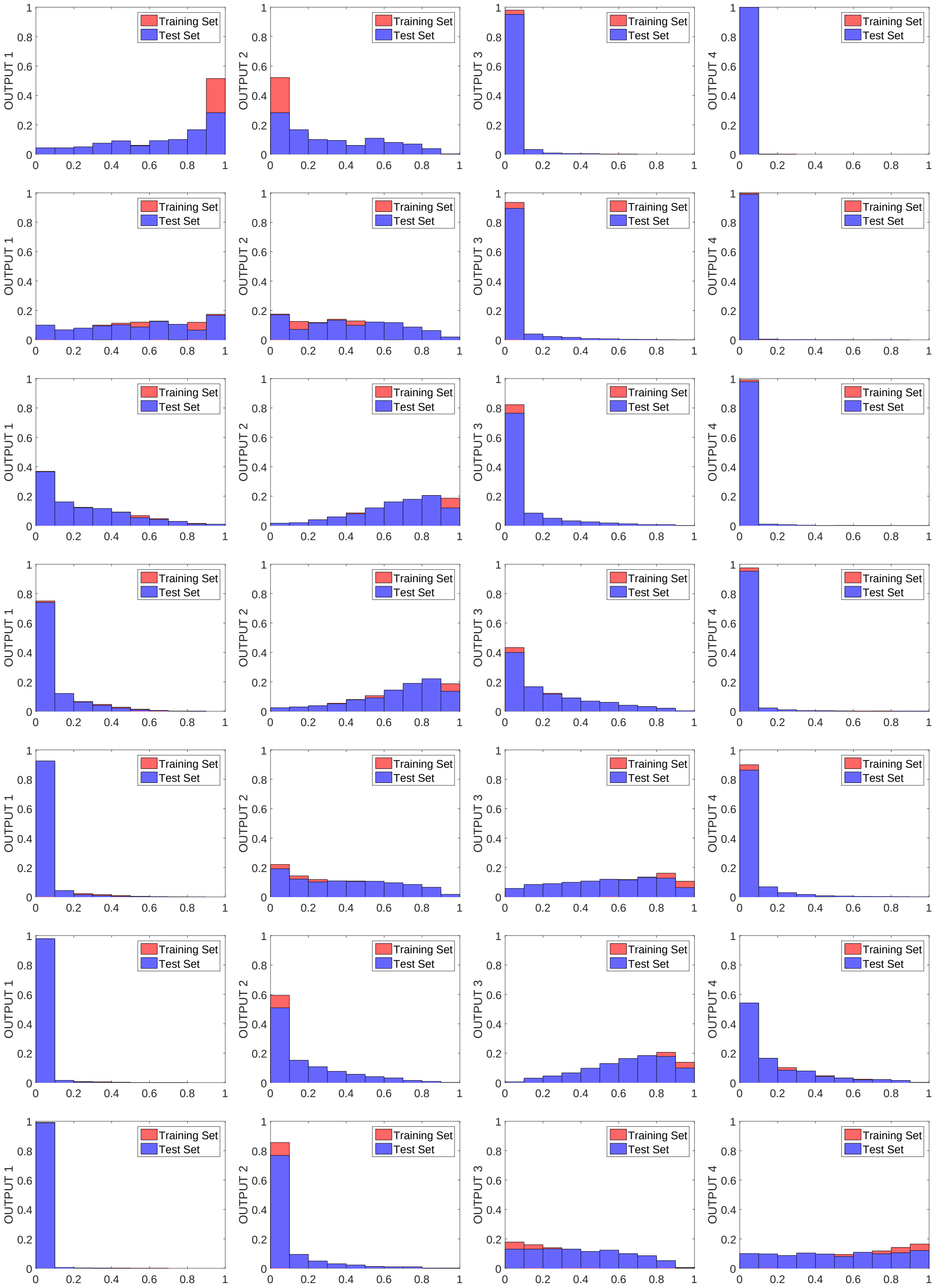


Figure 5.15: The distribution of ScaleNet's 4 outputs on both the training and test set, where the j-th column corresponds to the j-th output of ScaleNet and each row corresponds to a different case (see text for various cases).

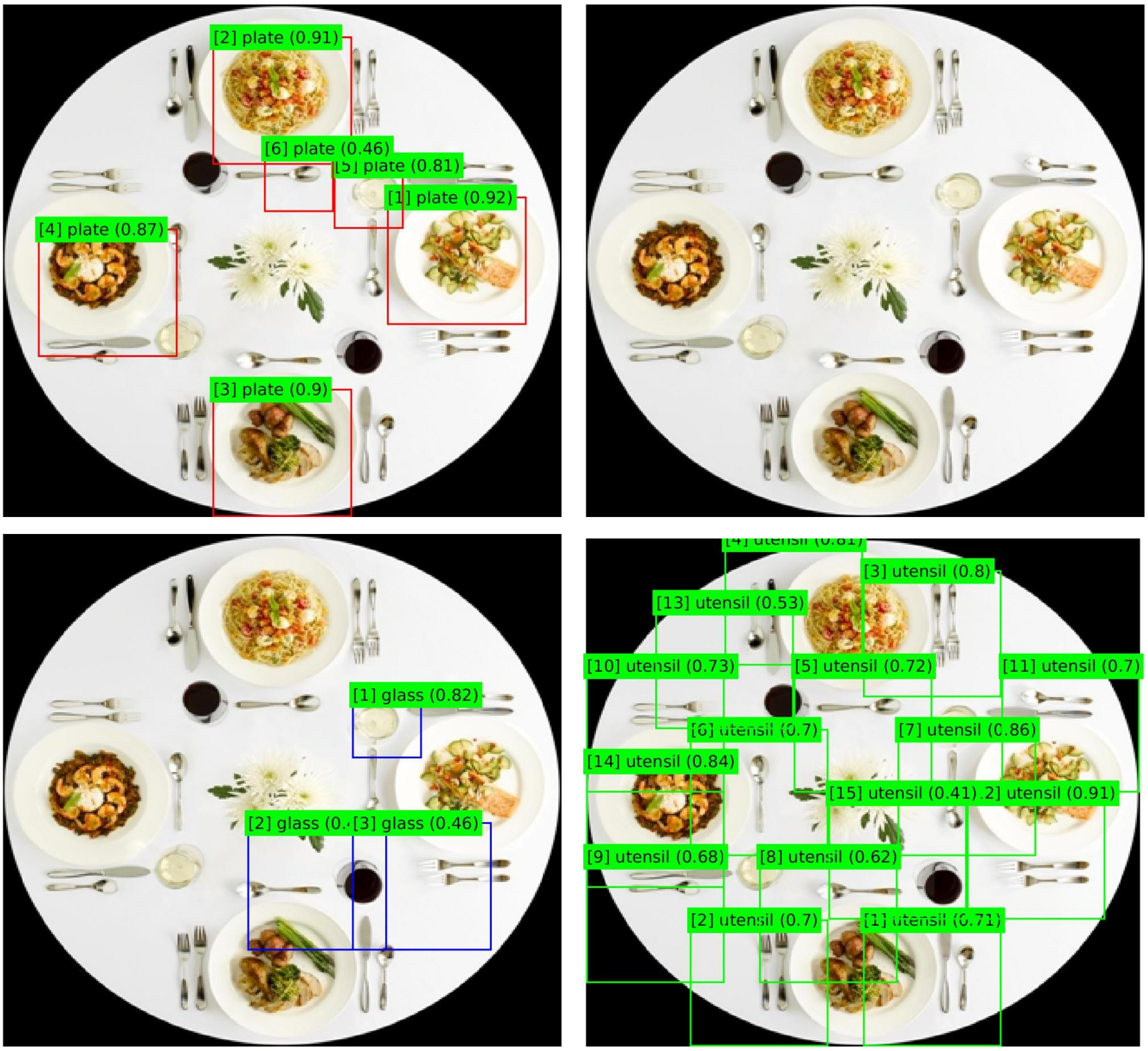


Figure 5.16: CNN classifier detections for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category and the fractional values in parentheses indicate the scale ratio of detections.

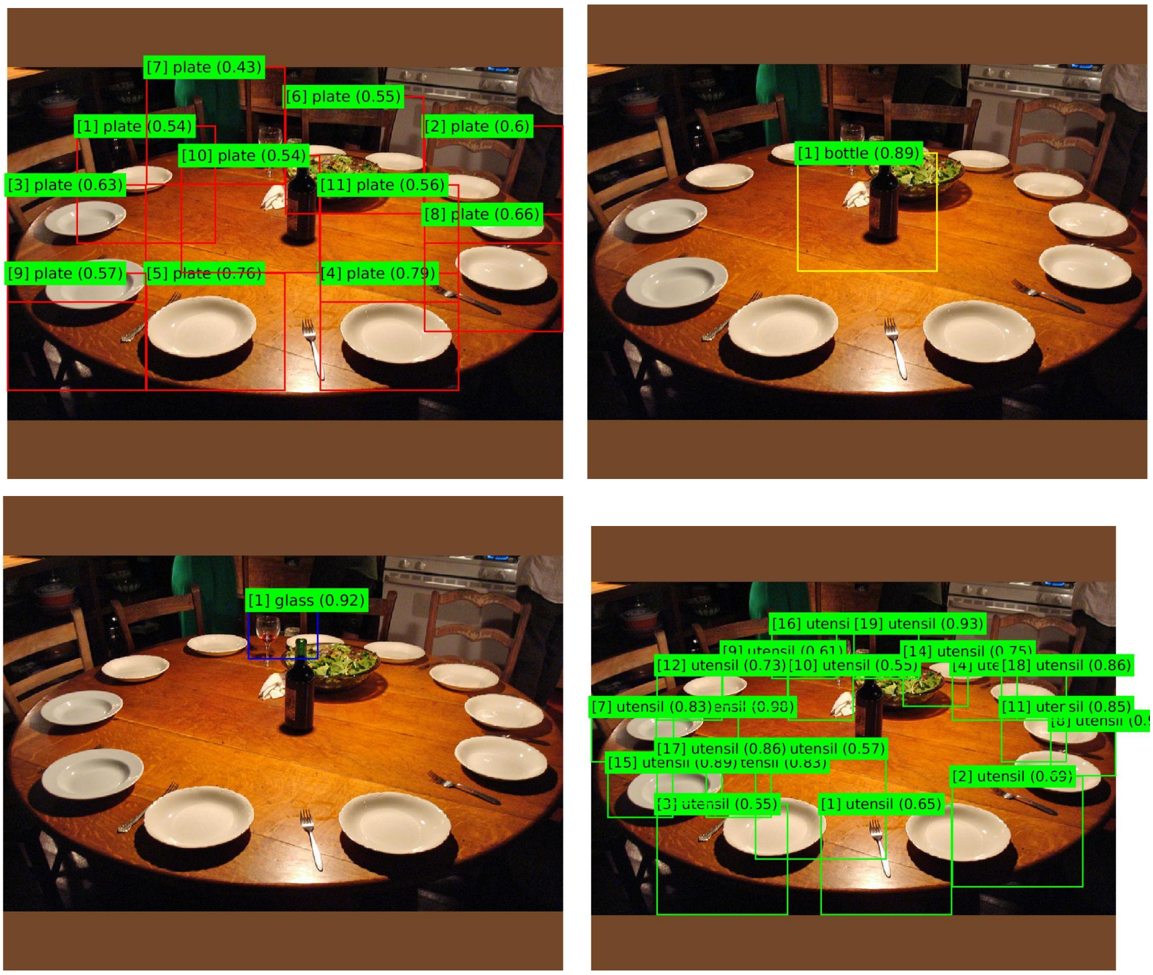


Figure 5.17: CNN classifier detections for "plate" (top left), "bottle" (top right), and "glass" (bottom left), and "utensil" (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category and the fractional values in parentheses indicate the scale ratio of detections.

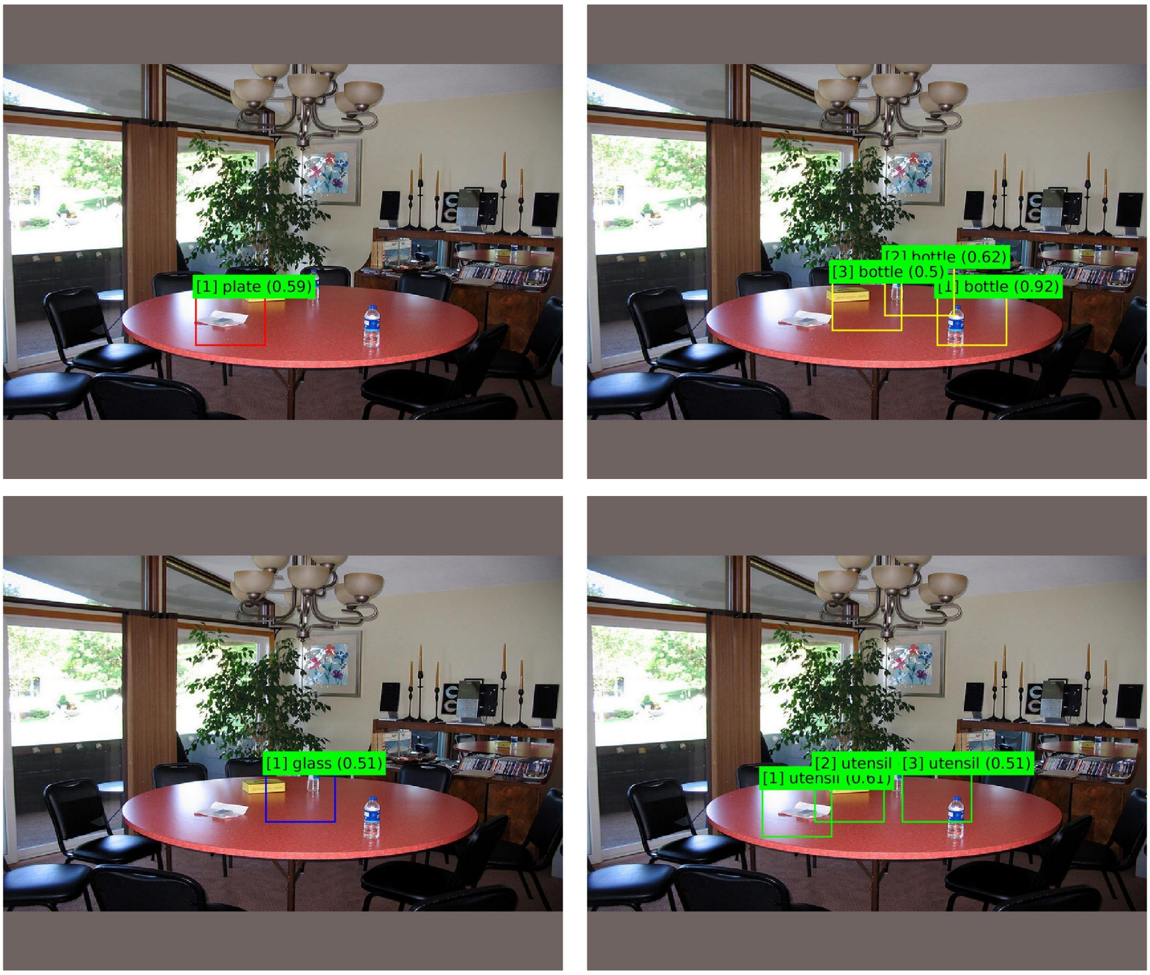


Figure 5.18: CNN classifier detections for "plate" (top left), "bottle" (top right), and "glass" (bottom left), and "utensil" (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category and the fractional values in parentheses indicate the scale ratio of detections.

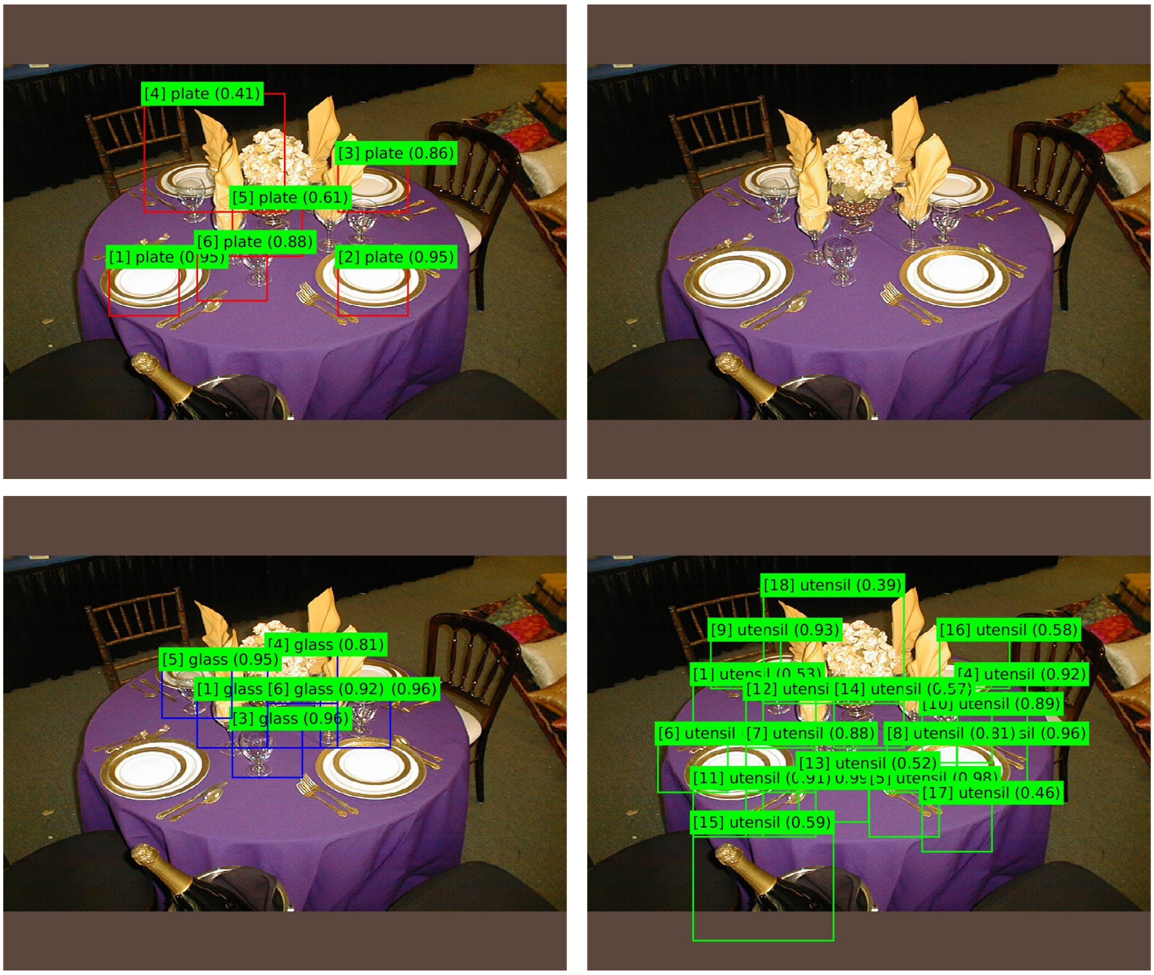


Figure 5.19: CNN classifier detections for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category and the fractional values in parentheses indicate the scale ratio of detections.

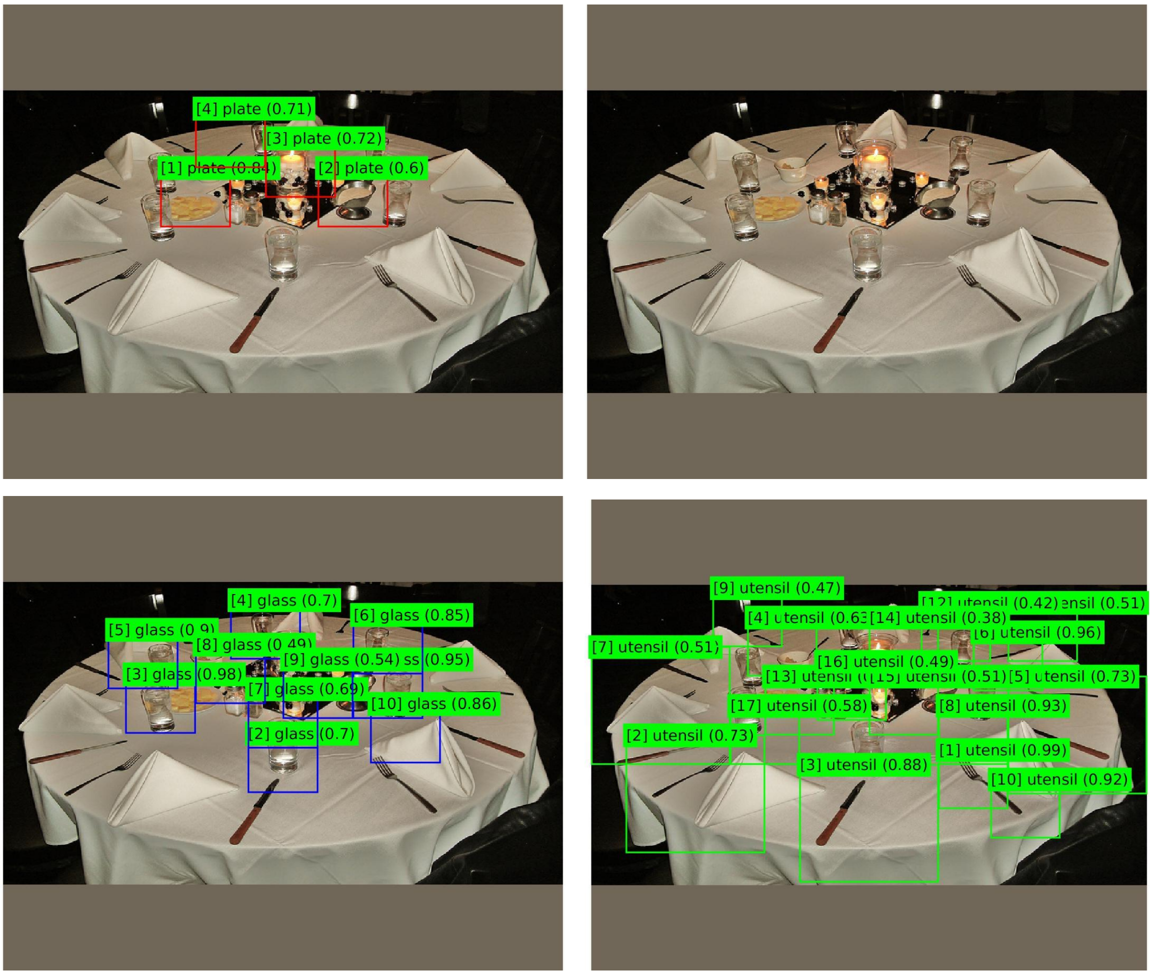


Figure 5.20: CNN classifier detections for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category and the fractional values in parentheses indicate the scale ratio of detections.

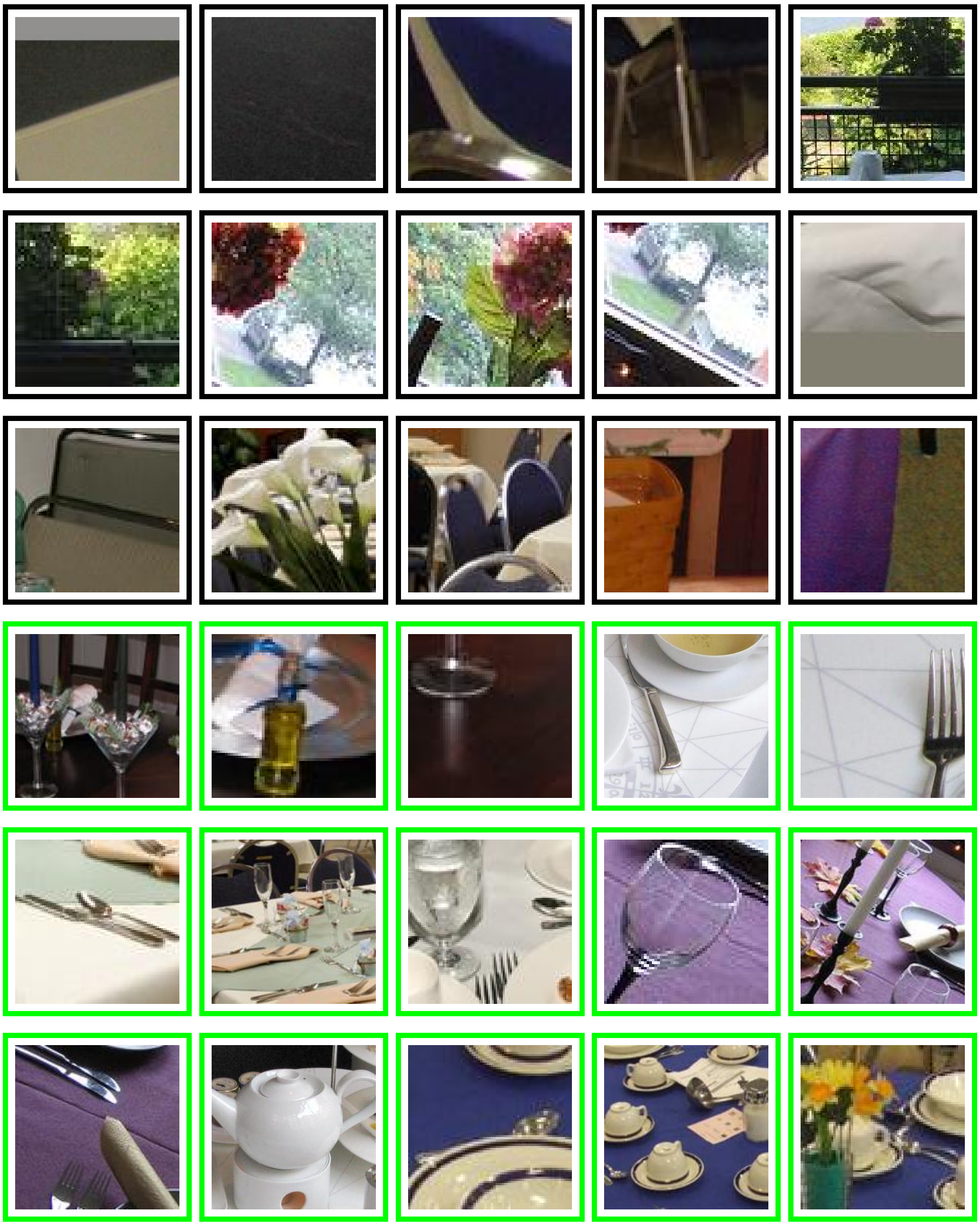
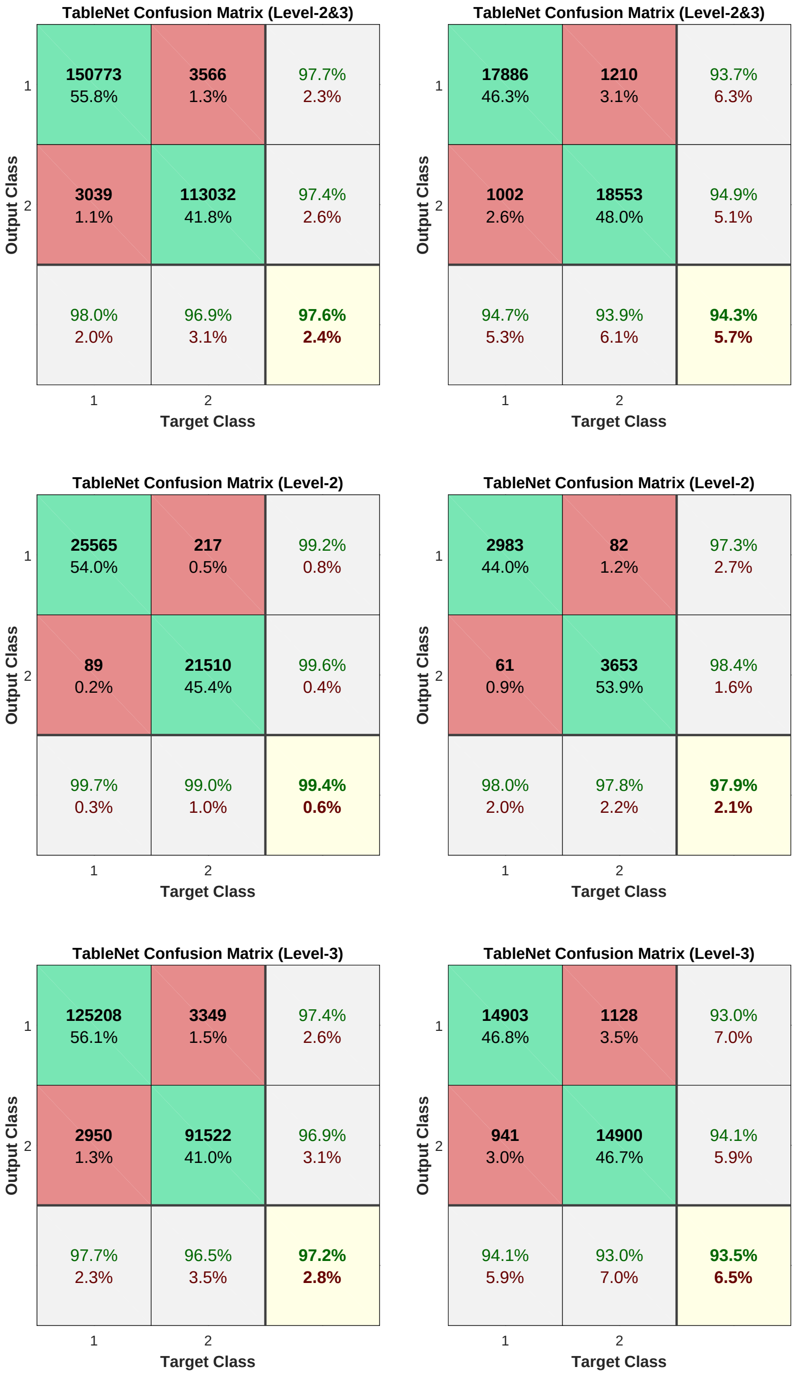


Figure 5.21: A snapshot of TableNet training/test patches including background (with black frames) and table (with green frames) patches.



TableNet Confusion Matrix (Level-2&3)

Output Class \ Target Class	1	2	
1	**150773** 55.8%	**3566** 1.3%	97.7% 2.3%
2	**3039** 1.1%	**113032** 41.8%	97.4% 2.6%
	98.0% 2.0%	96.9% 3.1%	**97.6%** **2.4%**

TableNet Confusion Matrix (Level-2&3)

Output Class \ Target Class	1	2	
1	**17886** 46.3%	**1210** 3.1%	93.7% 6.3%
2	**1002** 2.6%	**18553** 48.0%	94.9% 5.1%
	94.7% 5.3%	93.9% 6.1%	**94.3%** **5.7%**

TableNet Confusion Matrix (Level-2)

Output Class \ Target Class	1	2	
1	**25565** 54.0%	**217** 0.5%	99.2% 0.8%
2	**89** 0.2%	**21510** 45.4%	99.6% 0.4%
	99.7% 0.3%	99.0% 1.0%	**99.4%** **0.6%**

TableNet Confusion Matrix (Level-2)

Output Class \ Target Class	1	2	
1	**2983** 44.0%	**82** 1.2%	97.3% 2.7%
2	**61** 0.9%	**3653** 53.9%	98.4% 1.6%
	98.0% 2.0%	97.8% 2.2%	**97.9%** **2.1%**

TableNet Confusion Matrix (Level-3)

Output Class \ Target Class	1	2	
1	**125208** 56.1%	**3349** 1.5%	97.4% 2.6%
2	**2950** 1.3%	**91522** 41.0%	96.9% 3.1%
	97.7% 2.3%	96.5% 3.5%	**97.2%** **2.8%**

TableNet Confusion Matrix (Level-3)

Output Class \ Target Class	1	2	
1	**14903** 46.8%	**1128** 3.5%	93.0% 7.0%
2	**941** 3.0%	**14900** 46.7%	94.1% 5.9%
	94.1% 5.9%	93.0% 7.0%	**93.5%** **6.5%**

Figure 5.22: TableNet confusion matrices. The left and right columns correspond to, respectively, the training and test sets. The top, middle, and bottom rows correspond to the cases where patches are from both levels (2 and 3), level–2, and level–3, respectively.

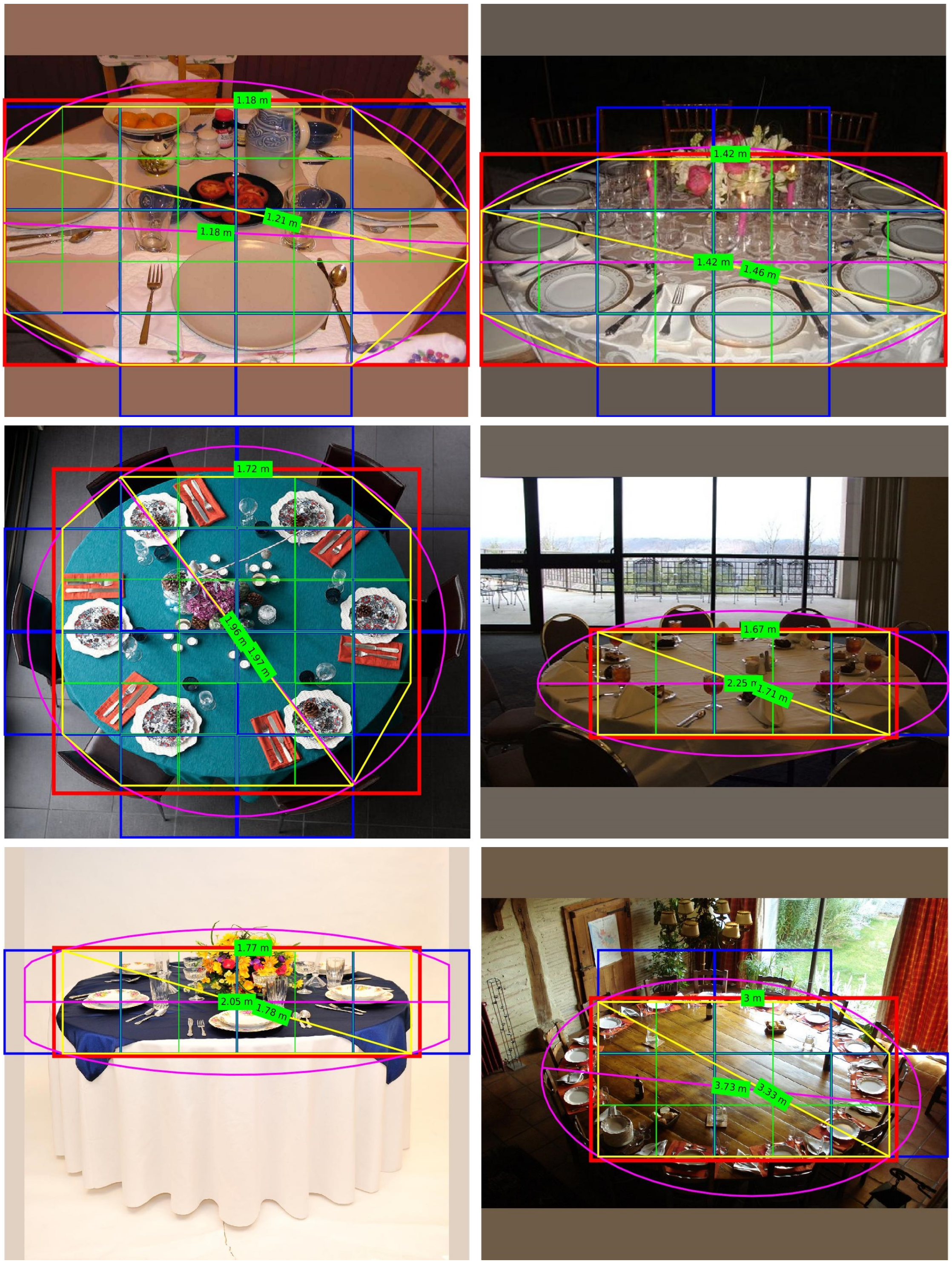


Figure 5.23: Table detection examples using TableNet: The fitting polygon, rectangle, and ellipse to the corner the points of patches at level–3 which were classified as table are shown in yellow, red, and magenta, respectively. The blue and green boxes show patches from, respectively, level-2 and level-3 classified as table. The estimated table size (in meters) based on each shape is shown on the green text boxes.

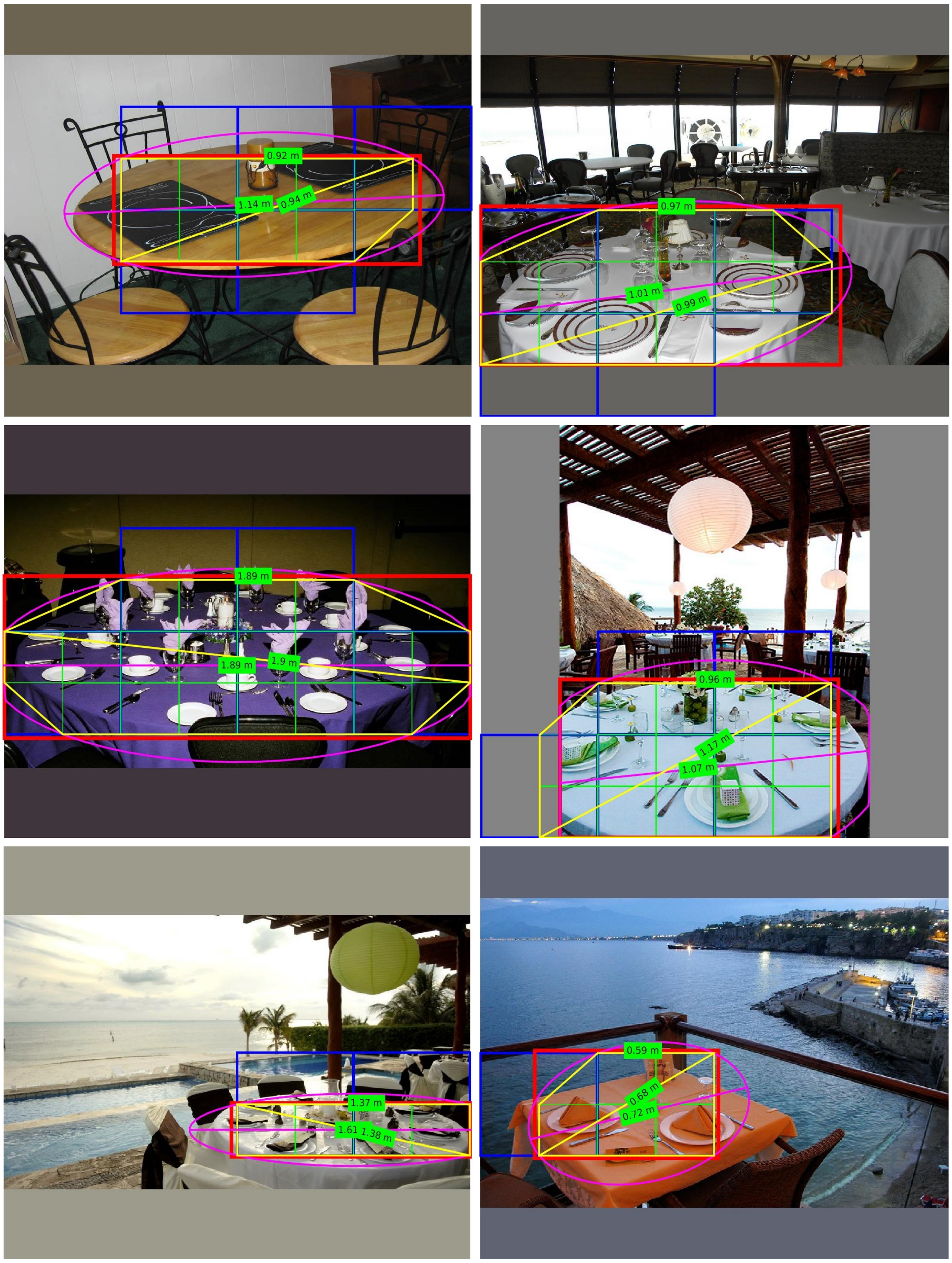


Figure 5.24: Table detection examples using TableNet: The fitting polygon, rectangle, and ellipse to the corner the points of patches at level–3 which were classified as table are shown in yellow, red, and magenta, respectively. The blue and green boxes show patches from, respectively, level-2 and level-3 classified as table. The estimated table size (in meters) based on each shape is shown on the green text boxes.

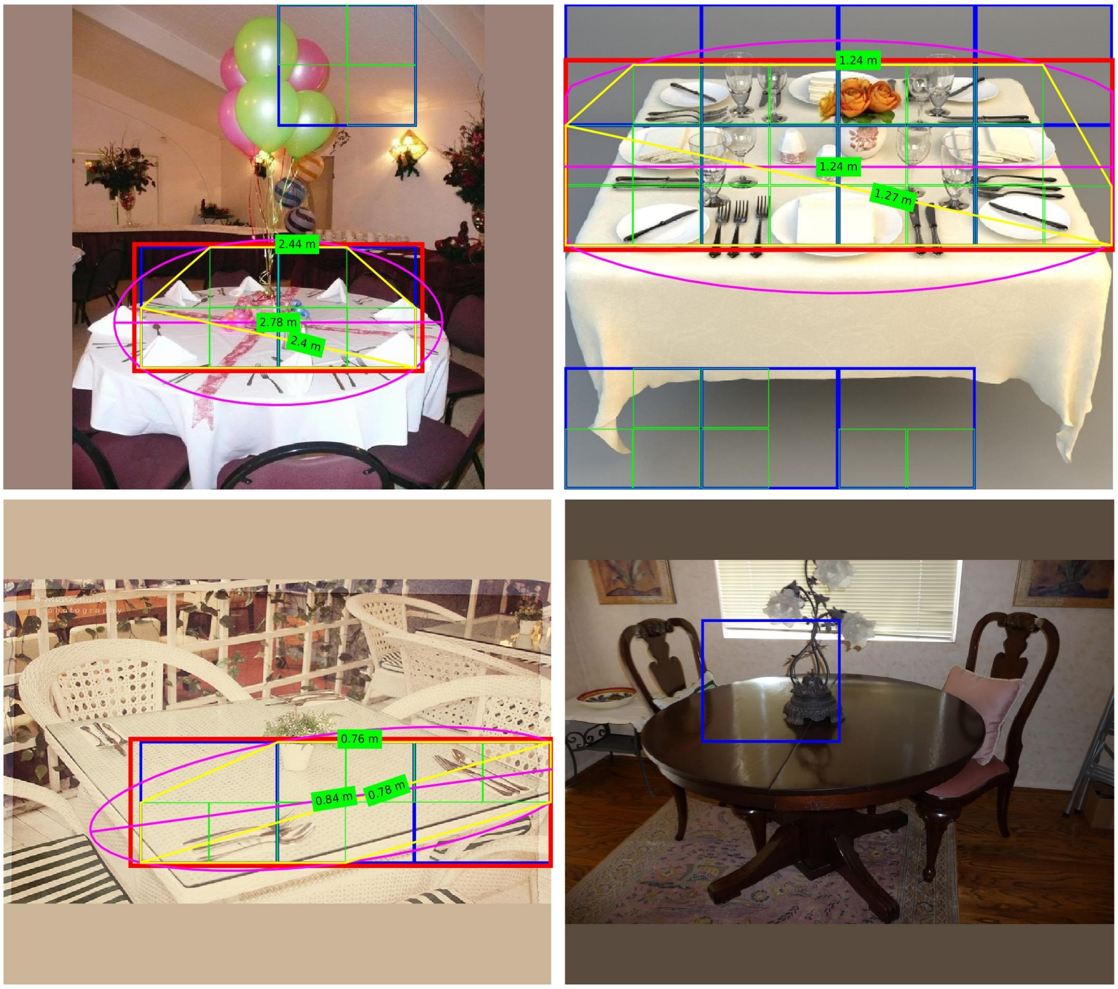


Figure 5.25: Noisy table detection examples using TableNet: The top row shows two examples with off-table false positives which were suppressed by considering the region with the maximum number of connected positive detections. The bottom row shows two poor table detection examples, perhaps due to the insufficient texture on the tables.

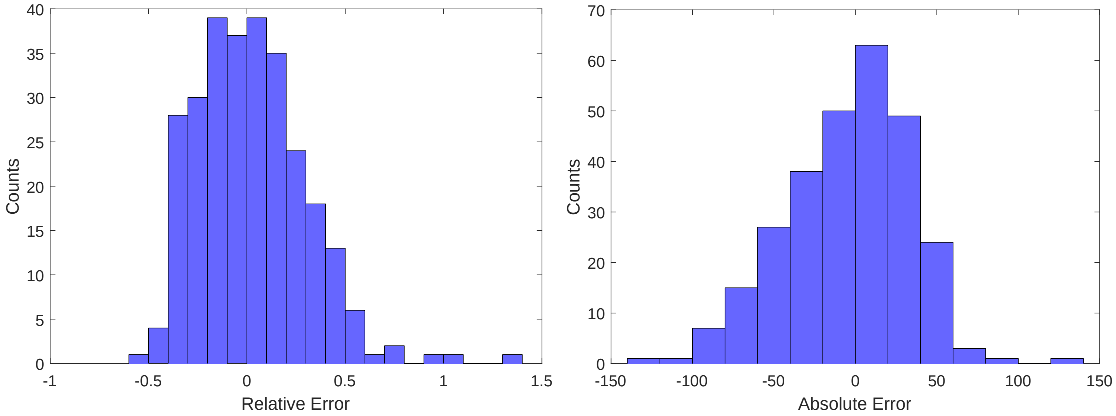


Figure 5.26: Histogram of the relative (left) and absolute (right) error made by the table–size estimator.

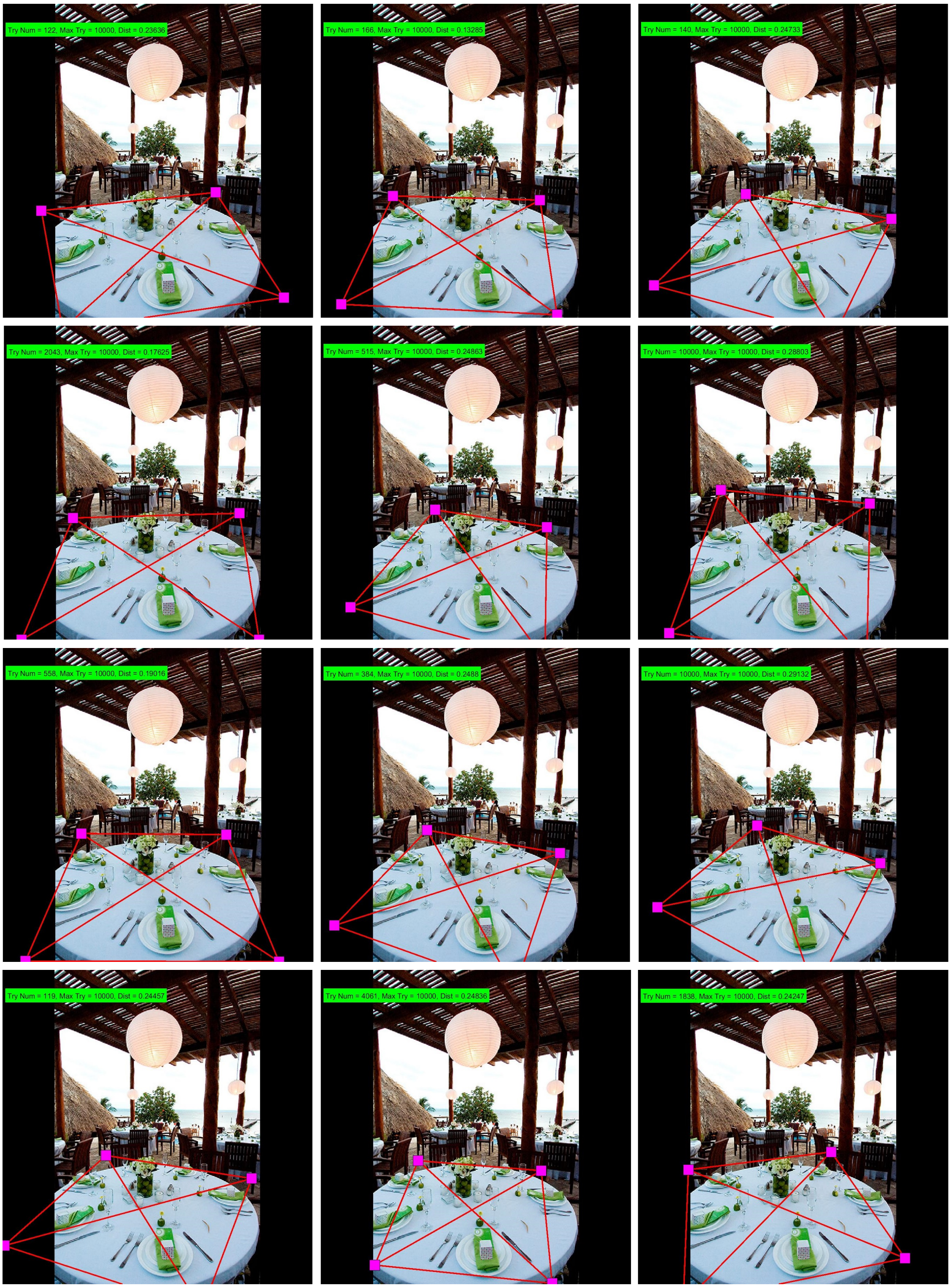

Try Num = 122, Max Try = 10000, Dist = 0.23636
Try Num = 166, Max Try = 10000, Dist = 0.13285
Try Num = 140, Max Try = 10000, Dist = 0.24733
Try Num = 2043, Max Try = 10000, Dist = 0.17625
Try Num = 515, Max Try = 10000, Dist = 0.24863
Try Num = 10000, Max Try = 10000, Dist = 0.28803
Try Num = 558, Max Try = 10000, Dist = 0.19016
Try Num = 384, Max Try = 10000, Dist = 0.2488
Try Num = 10000, Max Try = 10000, Dist = 0.29132
Try Num = 119, Max Try = 10000, Dist = 0.24457
Try Num = 4061, Max Try = 10000, Dist = 0.24836
Try Num = 1838, Max Try = 10000, Dist = 0.24247

Figure 5.27: Consistent homography samples satisfy condition (5.4)(set 1).

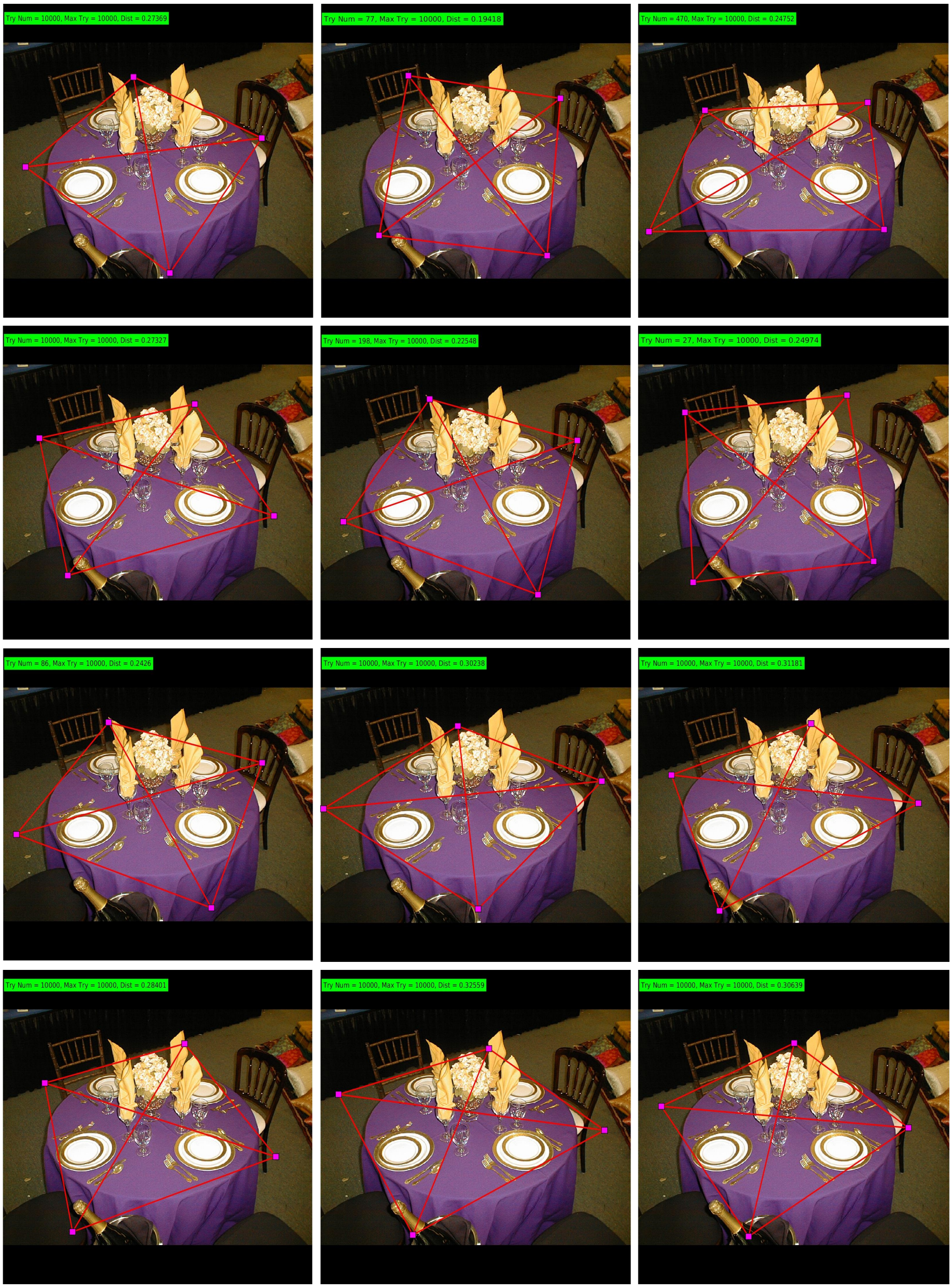

Try Num = 10000, Max Try = 10000, Dist = 0.27369
Try Num = 77, Max Try = 10000, Dist = 0.19418
Try Num = 470, Max Try = 10000, Dist = 0.24752
Try Num = 10000, Max Try = 10000, Dist = 0.27327
Try Num = 198, Max Try = 10000, Dist = 0.22548
Try Num = 27, Max Try = 10000, Dist = 0.24974
Try Num = 86, Max Try = 10000, Dist = 0.2426
Try Num = 10000, Max Try = 10000, Dist = 0.30238
Try Num = 10000, Max Try = 10000, Dist = 0.31181
Try Num = 10000, Max Try = 10000, Dist = 0.28401
Try Num = 10000, Max Try = 10000, Dist = 0.32559
Try Num = 10000, Max Try = 10000, Dist = 0.30639

Figure 5.28: Consistent homography samples satisfy condition (5.4)(set 2).

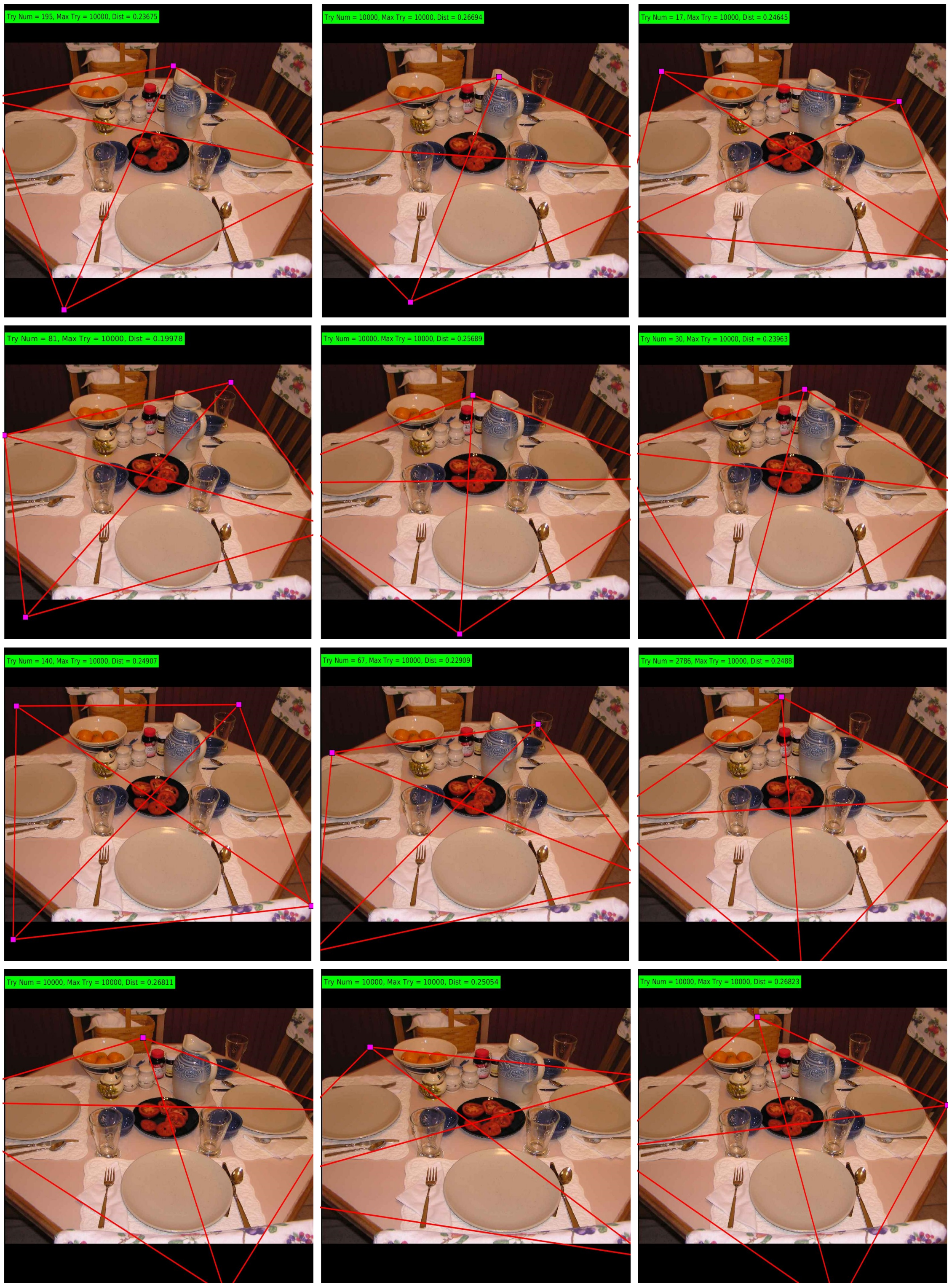

Try Num = 195, Max Try = 10000, Dist = 0.23675
Try Num = 10000, Max Try = 10000, Dist = 0.26694
Try Num = 17, Max Try = 10000, Dist = 0.24645
Try Num = 81, Max Try = 10000, Dist = 0.19978
Try Num = 10000, Max Try = 10000, Dist = 0.25689
Try Num = 30, Max Try = 10000, Dist = 0.23963
Try Num = 140, Max Try = 10000, Dist = 0.24907
Try Num = 67, Max Try = 10000, Dist = 0.22909
Try Num = 2786, Max Try = 10000, Dist = 0.2488
Try Num = 10000, Max Try = 10000, Dist = 0.26811
Try Num = 10000, Max Try = 10000, Dist = 0.25054
Try Num = 10000, Max Try = 10000, Dist = 0.26823

Figure 5.29: Consistent homography samples satisfy condition (5.4)(set 3).

5.2 Data Model

Let X^t denote the output of running the TableNet on patches from the input image I, namely $X^t = \{X^t_m, m \in \mathcal{A}_T(I) \subset \mathcal{A}\}$. Let $S = S(X^t)$ denote the detected table area in the image coordinate system. Let M be the subset of annocells at level-2 with $X^t_m = 1$; then, X^s_M is the set of ScaleNet outputs that are involved in estimating T *i.e.,* $\hat{T} = \hat{T}(X^s_M)$. Let $\mathbf{X} = \big\{(X^c_m, X^s_m),\ m \in \{1, ..., 1036\}\big\}$ denote the output of running CatNet and ScaleNet on some annocell patches from the input image. Finally, let $\mathbf{E_t}$ be the expanded evidence after t steps, *i.e.,* adjoining (X^t, X^s_M) to $\mathbf{e}_t$ composed of (X^c_m, X^s_m) pairs obtained from processing patches selected by EP so far.

We need to generate samples from $p(\mathbf{Y}^c|\mathbf{E}_t)$ for the EP query engine. To do so it is sufficient to draw $(\omega, \mathcal{W})$ samples conditional on $\mathbf{E}_t$. Having $(\omega, \mathcal{W})$ conditional samples one can project them to the image coordinate system and get the corresponding (c_V, ξ_V) and $\mathbf{Y}^c$ conditional samples. We have:

$$\begin{aligned} p(\omega, \mathcal{W}, T|\mathbf{E}_t) &= p(T|\mathbf{E}_t)p(\mathcal{W}|T, \mathbf{E}_t)p(\omega|T, \mathcal{W}, \mathbf{E}_t) \\ &\approx p(T|\hat{T}, \mathbf{e}_t)p(\mathcal{W}|T, S, \mathbf{e}_t)p(\omega|T, \mathcal{W}, \mathbf{e}_t). \qquad (5.5) \end{aligned}$$

The estimated table size $\hat{T}$ can limit the range of T to be close to the estimated size. We also have:

$$\begin{aligned}
p(\omega|T,\mathcal{W},\mathbf{e}_t) &\propto p(\omega,T,\mathcal{W},\mathbf{e}_t)\\
&= p(\omega,T,\mathcal{W}) \times p(\mathbf{e}_t|\omega,T,\mathcal{W})\\
&= p(\omega,T,\mathcal{W}) \times \prod_m p(X^c_m|\omega,\mathcal{W}) \prod_m p(X^s_m|\omega,\mathcal{W}) \qquad (5.6)
\end{aligned}$$

where the prior distribution $p(\omega, T, \mathcal{W})$ is given by (4.10). In (5.6), we have assumed that $\mathbf{e}_t$ depends only on the category and 2D pose of objects and that the outputs of CatNet and ScaleNet are conditionally independent given (c_V, ξ_V) calculated based on $(\omega, \mathcal{W})$. We have also assumed that the classifiers are conditionally independent of each other, a key assumption. Note that we trained CatNet and ScaleNet independent of the objects' scale and category, respectively. The CatNet was trained to respond to the existence of object instances from different categories in the input patch. Hence, we assume that the corresponding annoint is a sufficient statistic for the CatNet data model, namely:

$$p(X^c_i|c_V,\xi_V) = p(X^c_i|Y^c_i). \qquad (5.7)$$

We quantized the scale ratios to the 7 intervals listed in section 5.1.2 (see Figure 5.12). We denote the scale index of the i-th patch by $Y^s_i \in \{1, ..., 7\}$, that is equal to its case number computed based on the average scale of contained object instances. We assume that the scale index Y^s_i computed from (c_V, ξ_V) is a sufficient statistic for the ScaleNet data model, namely:

$$p(X_i^s|c_V, \xi_V) = p(X_i^s|Y_i^s). \tag{5.8}$$

We model both of the CatNet and ScaleNet data models using Dirichlet distributions, which are probability densities on probabilities or normalized proportions. Let $\mathbf{x}$ denote a random vector whose k–th element is denoted by x_k. The probability density of a Dirichlet model with parameter vector α at $\mathbf{x}$ is:

$$p(\mathbf{x}) \sim \text{Dir}(\alpha_1, ..., \alpha_K) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k x_k^{\alpha_k - 1}, \tag{5.9}$$

where $x_k > 0$, $\sum_k x_k = 1$, and $\Gamma(.)$ indicates the Gamma function.

We learned 16 CatNet data models for different annoint configurations indicating presence/absence of 4 object categories: “Plate”, “Bottle”, “Glass”, and “Utensil” in the input patch. We also learned 7 ScaleNet data models, one per scale index. These models are learned from data collected by running the CNNs on patches with matching configuration. Since there is no closed–form solution for the maximum–likelihood estimates (MLE) of the Dirichlet distribution we used a fixed-point (without projection) iterative schemes to perform MLE parameter estimation according to [62] as follows. The maximum-likelihood estimate of α given a training set of proportions $D = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ maximizes the log-likelihood:

$$\log p(D|\alpha) = N \log \Gamma(\sum_k \alpha_k) - N \sum_k \log \Gamma(\alpha_k) + N \sum_k (\alpha_k - 1) \log \tilde{x}_k, \tag{5.10}$$

where $\log \tilde{x}_k = \sum_i x_{ik}/N$. Since the Dirichlet distribution is from the exponential family the log-likelihood in (5.10) is convex in α [77] leading to straightforward optimization. The gradient of the log-likelihood with respect to α_k is:

$$g_k = \frac{d \log p(D|\alpha)}{d\alpha_k} = N\Psi(\alpha_k) + N \log \tilde{x}_k, \tag{5.11}$$

where $\Psi(z) = d \log \Gamma(z)/dz$ is known as the Digamma function. We used a fixed-point iteration for maximizing the likelihood according to [62] as follows. We start from a guess for parameters α and iteratively maximize a simple lower bound on the likelihood which is tight at α. The maximum of this bound can be computed in closed–form and it becomes the currect guess for the next iteration. Such an iteration is guaranteed to converge to a stationary point of the likelihood, and for the Dirichlet distribution, the maximum is the only stationary point. Using the bound:

$$\Gamma(z) \geq \Gamma(\hat{z}) \exp\big((z - \hat{z})\Psi(\hat{z})\big), \tag{5.12}$$

on $\Gamma(\sum_k \alpha_k)$ in (5.10) we get:

$$\frac{1}{N}\log p(D|\alpha) \geq \big(\sum_k \alpha_k\big)\Psi\big(\sum_k \alpha_k^{old}\big) - \sum_k \log\Gamma(\alpha_k) + \sum_k (\alpha_k - 1)\log\tilde{x}_k + \text{(const.)} \tag{5.13}$$

Differentiating the lower bound in (5.13) with respect to α_k and equating it to zero leads to the following fixed-point iteration:

$$\Psi(\alpha_k^{new}) = \Psi(\sum_k \alpha_k^{old}) + \log\tilde{x}_k. \tag{5.14}$$

For the exponential family distributions, when the gradient of log-likelihood is zero, the expected sufficient statistics and the observed sufficient statistics are equal [62]. In this case, the expected sufficient statistics are:

$$\mathbb{E}[\log x_k] = \Psi(\alpha_k) - \Psi(\sum_k \alpha_k), \tag{5.15}$$

and the observed sufficient statistics are $\log\tilde{x}_k$. Note the similarity of updates according to (5.14) and equating the expected sufficient statistics in (5.15) to the observed sufficient statistics $\log\tilde{x}_k$.

Computing the new set of parameters according to (5.14) requires inversion of the Ψ function. An efficient and highly accurate (fourteen digits of precision after five Newton iterations) way of inverting the Ψ function is as follows. The Ψ function inversion involves solving $\Psi(z) = y$ for z given y. Given a starting guess for z, we use the Newtons method to find the root of $\Psi(z) - y = 0$. The Newton update is:

$$z^{new} = z^{old} - \frac{\Psi(z) - y}{\Psi'(z)}. \tag{5.16}$$

To start the iteration, we use the following asymptotic formulas for $\Psi(z)$:

$$\Psi(z) \approx \begin{cases} \log(z - 1/2) & \text{if} \quad z \geq 0.6 \\ \\ -\frac{1}{z} - \gamma & \text{if} \quad z < 0.6 \end{cases} \tag{5.17}$$

where $\gamma = -\Psi(1)$ to get:

$$\Psi^{-1}(y) \approx \begin{cases} \exp(y) + 1/2 & \text{if} \quad y \geq -2.22 \\ \\ -\frac{1}{y+\gamma} & \text{if} \quad y < -2.22 \end{cases} \tag{5.18}$$

With the above initialization, five Newton iterations are sufficient to reach fourteen digits of precision [62].

A useful initialization for the model parameters before starting the iterates in (5.14) can be found by matching the moments based on the model versus the empirical data. The first two moments of the Dirichlet density are:

$$\mathbb{E}[x_k] = \frac{\alpha_k}{\sum_k \alpha_k}, \tag{5.19}$$

and:

$$\mathbb{E}[x_k^2] = \mathbb{E}[x_k] \frac{1 + \alpha_k}{1 + \sum_k \alpha_k}. \tag{5.20}$$

On the other hand we have:

$$\sum_k \alpha_k = \frac{\mathbb{E}[x_1] - \mathbb{E}[x_1^2]}{\mathbb{E}[x_1^2] - \mathbb{E}[x_1]^2}. \tag{5.21}$$

Multiplying (5.19) and (5.21) gives a formula for α_k in terms of moments. Equation (5.21) uses x_1, but any other x_k could also be used to estimate $\sum_k \alpha_k$. Ronning [77] suggests instead using all of the x_k's via:

$$\text{var}(x_k) = \frac{\mathbb{E}[x_k](1 - \mathbb{E}[x_k])}{1 + \sum_k \alpha_k}, \tag{5.22}$$

and:

$$\log \sum_k \alpha_k = \frac{1}{K-1} \sum_{k=1}^{K-1} \log\left(\frac{\mathbb{E}[x_k](1 - \mathbb{E}[x_k])}{\text{var}(x_k)} - 1 \right). \tag{5.23}$$

Figures 5.30, 5.31, 5.32, 5.33 show stacked bar visualization of 25 samples (per configuration) drawn randomly from data collected by running CatNet on patches (left column) and samples taken from the Dirichlet model learned from CatNet output data (right column) where each row corresponds to one of the 16 annoint configurations. The length of each colored bar represent the proportion of each category; therefore, the total length of each stacked bar is equal to 1. Two interesting observations are: (1) the length of bars corresponding to the present categories are comparable and usually considerably larger than the length of absent categories; (2) the color distribution of CatNet outputs and Dirichlet model samples are very similar for the same configuration. This supports the argument for using a the Dirichlet distribution in modeling the the data distribution $p(X^c|Y^c)$. Stacked bars are good means to visually inspect and compare the true empirical distribution versus the Dirichlet model.

Figures 5.34, 5.35, 5.36, 5.37 show bean plots illustrating the distribution of CatNet outputs versus the learned Dirichlet model, where each row corresponds to one of the 16 annoint configurations; the Dirichlet model closely mimics the distribution of the CatNe outputs which is used to train the Dirichlet data model. Figures 5.38, 5.39, 5.40, 5.41 illustrate the corresponding ScaleNet plots for the 7 scale configurations.

As a different perspective, Figures F.1, F.2, F.3, F.4 in in Appendix F show the normalized histogram of the CatNet outputs (framed in black boxes) versus 100,000 samples (framed in red boxes) from the Dirichlet data model for every annoint configuration. Figures F.5, F.6, F.7 show the corresponding normalized histograms for ScaleNet.

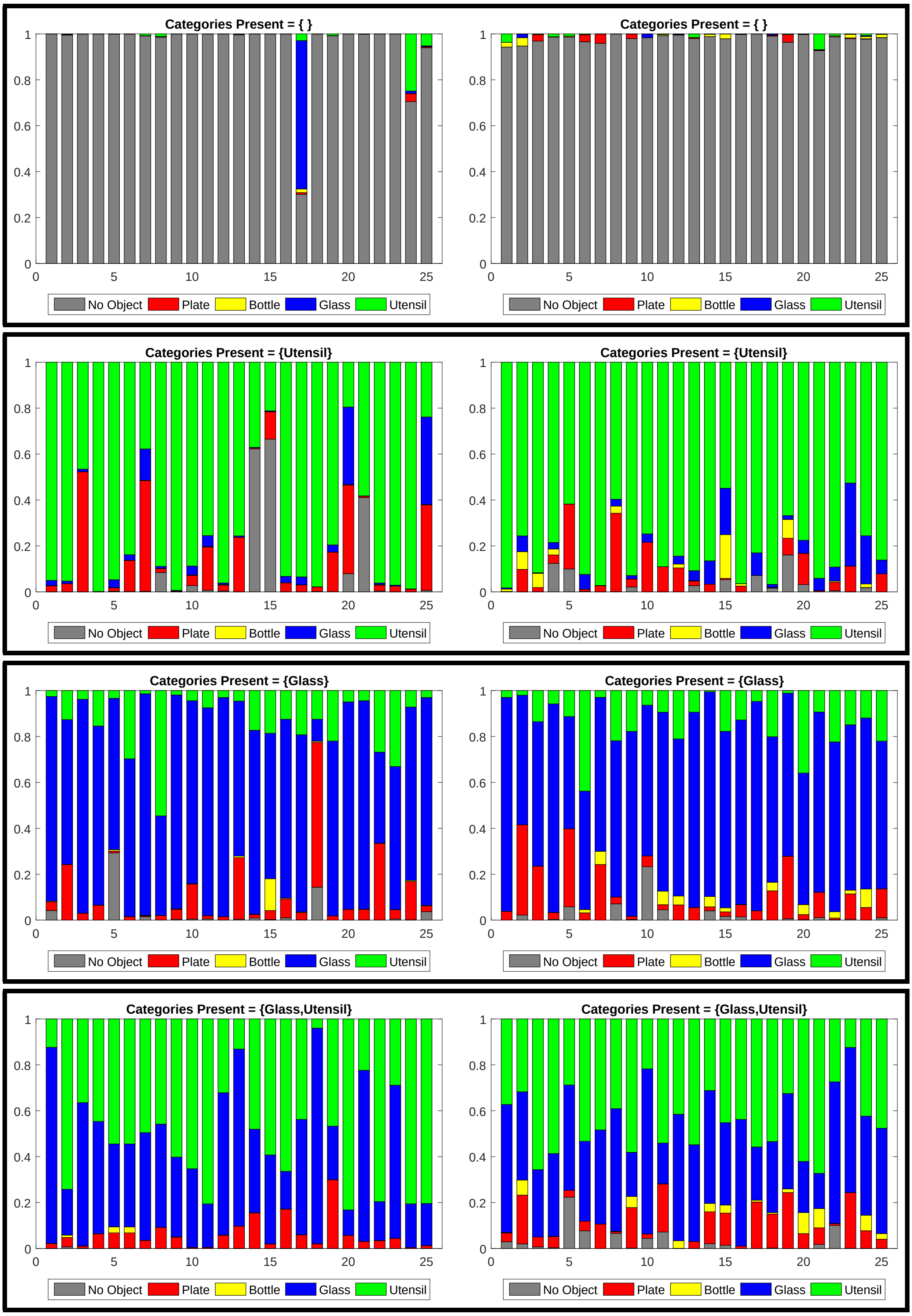


Figure 5.30: Stacked bar visualization of samples from CatNet output (left) and Dirichlet model (right).

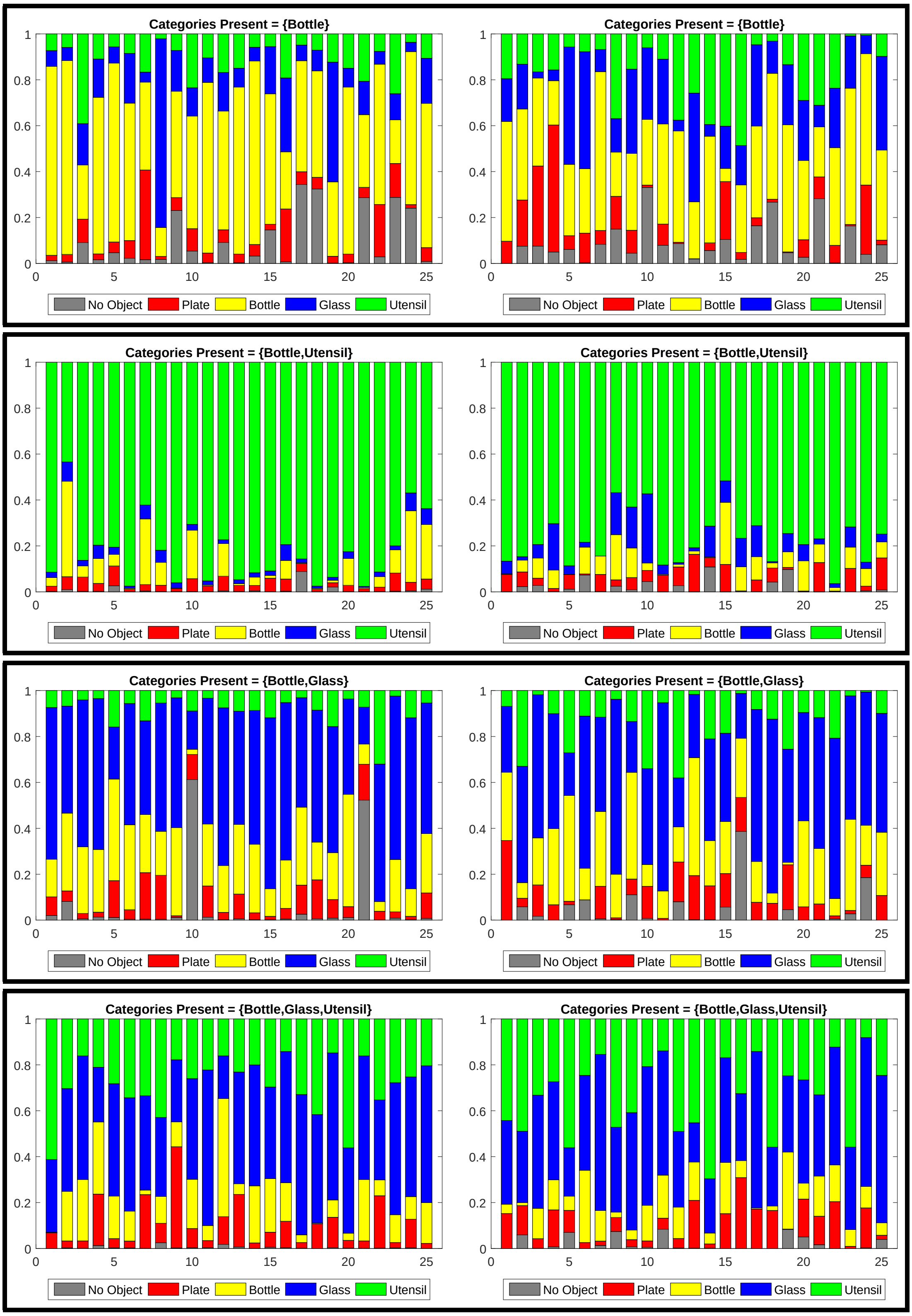


Figure 5.31: Stacked bar visualization of samples from CatNet output (left) and Dirichlet model (right).

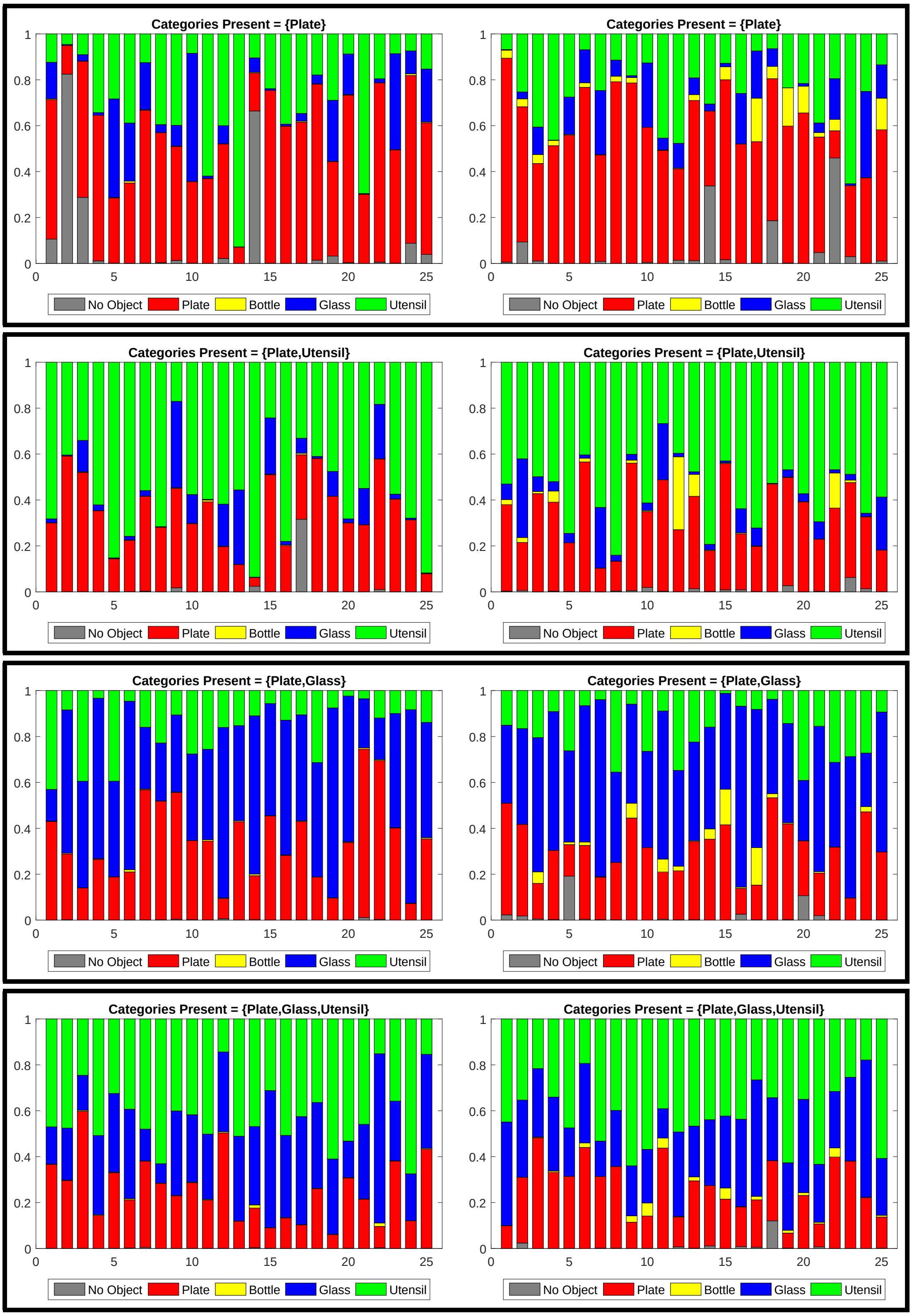


Figure 5.32: Stacked bar visualization of samples from CatNet output (left) and Dirichlet model (right).

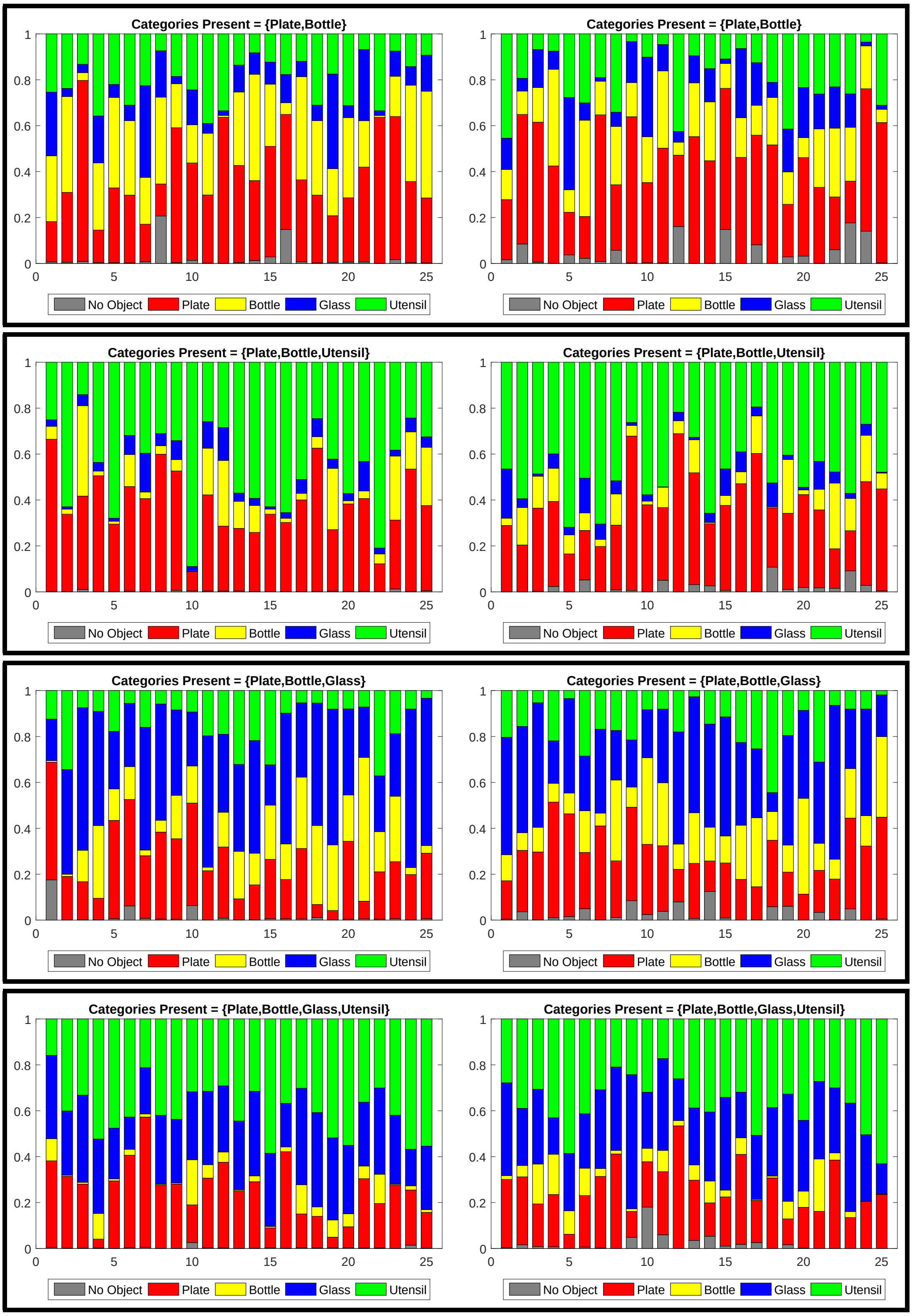


Figure 5.33: Stacked bar visualization of samples from CatNet output (left) and Dirichlet model (right).

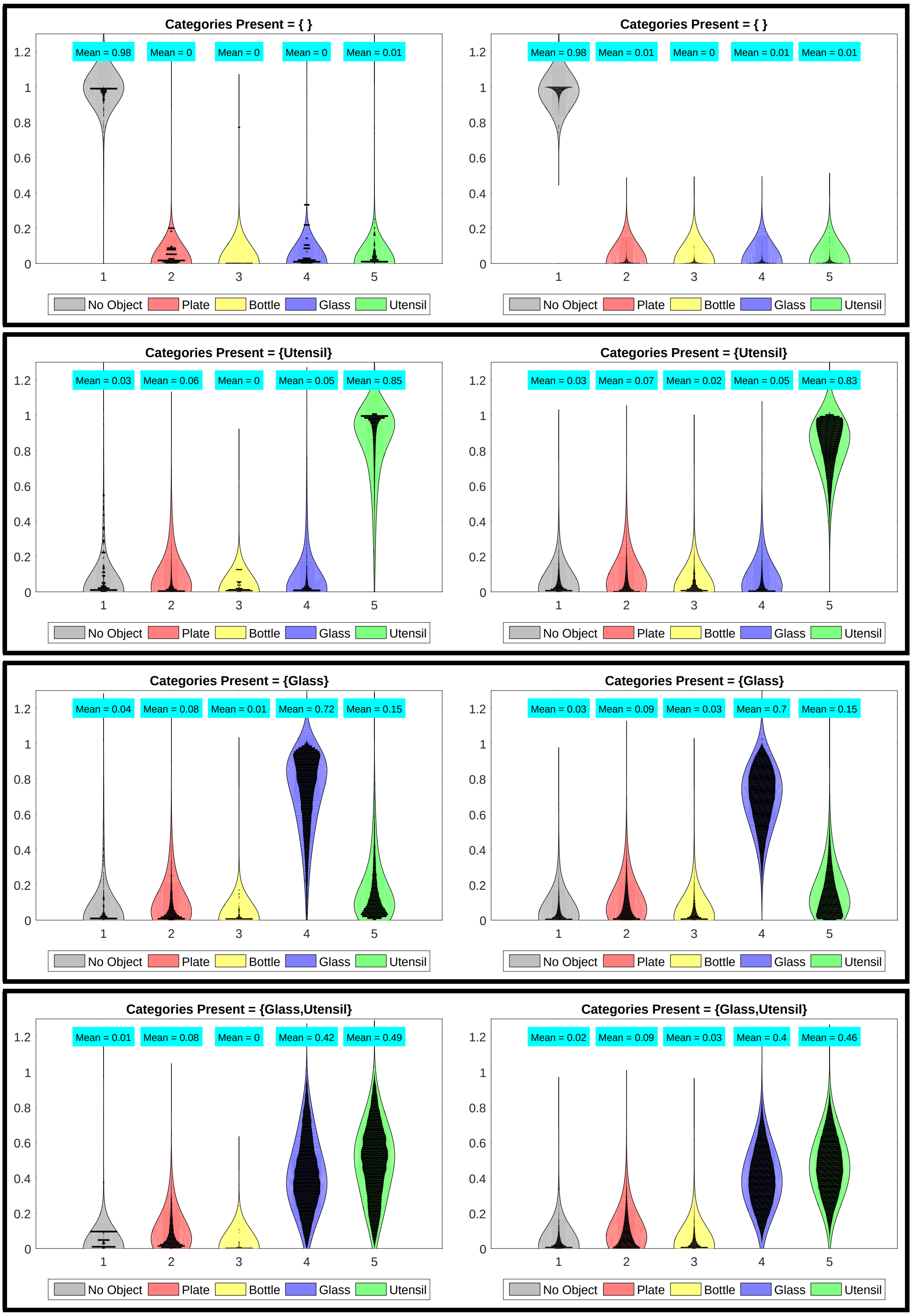


Figure 5.34: CatNet output and Dirichlet dist. bean plot.

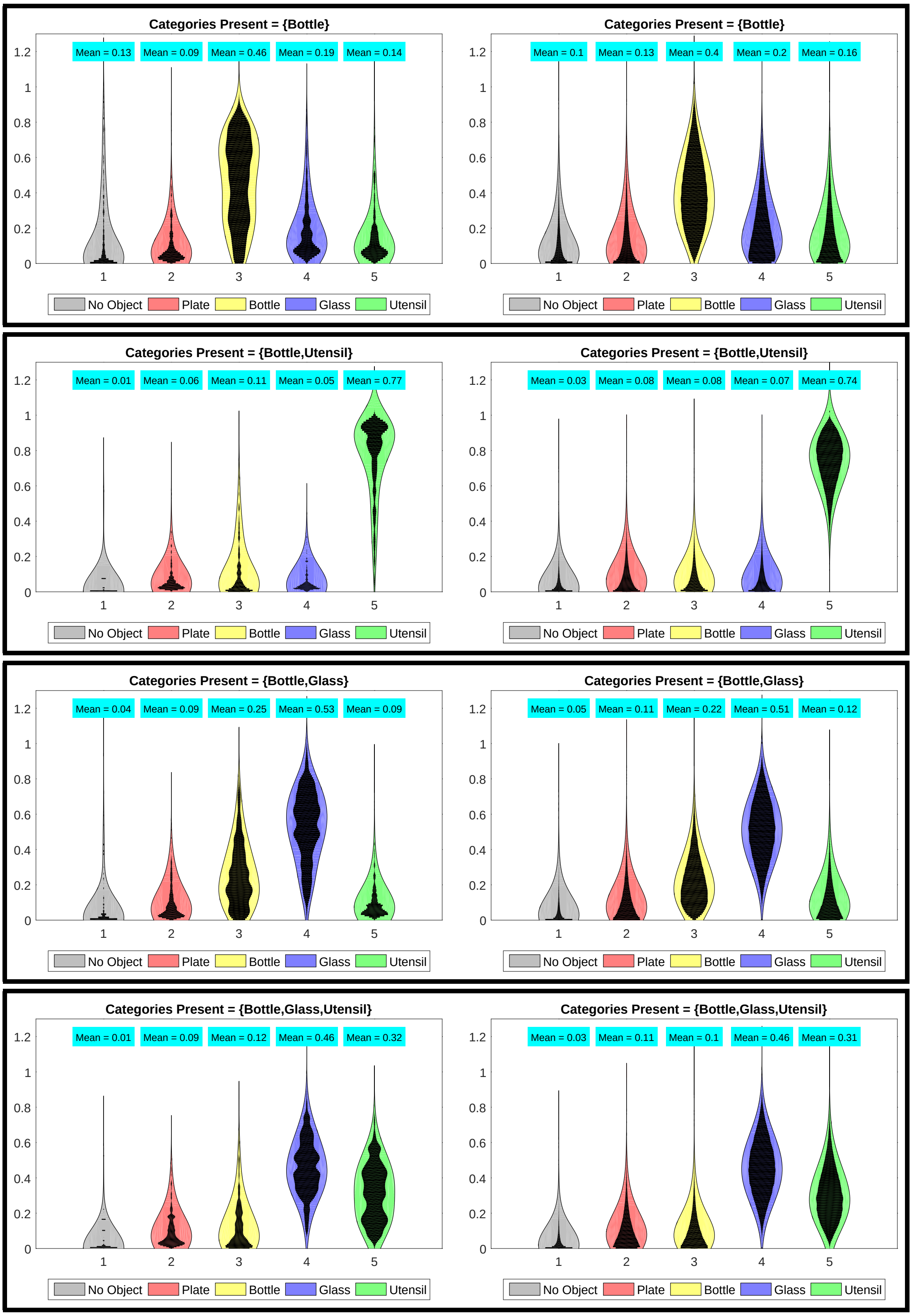


Figure 5.35: CatNet output and Dirichlet dist. bean plot.

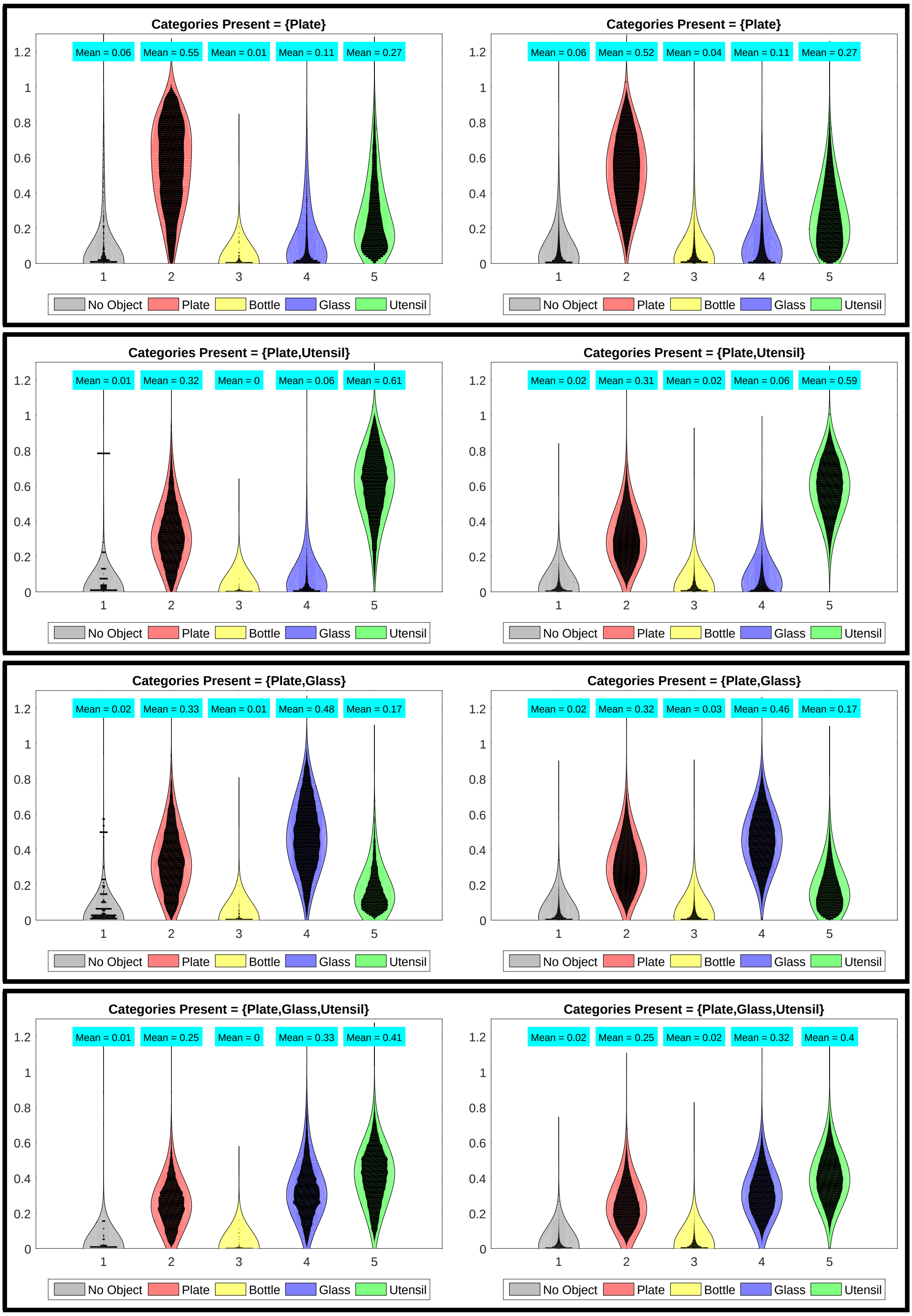


Figure 5.36: CatNet output and Dirichlet dist. bean plot.

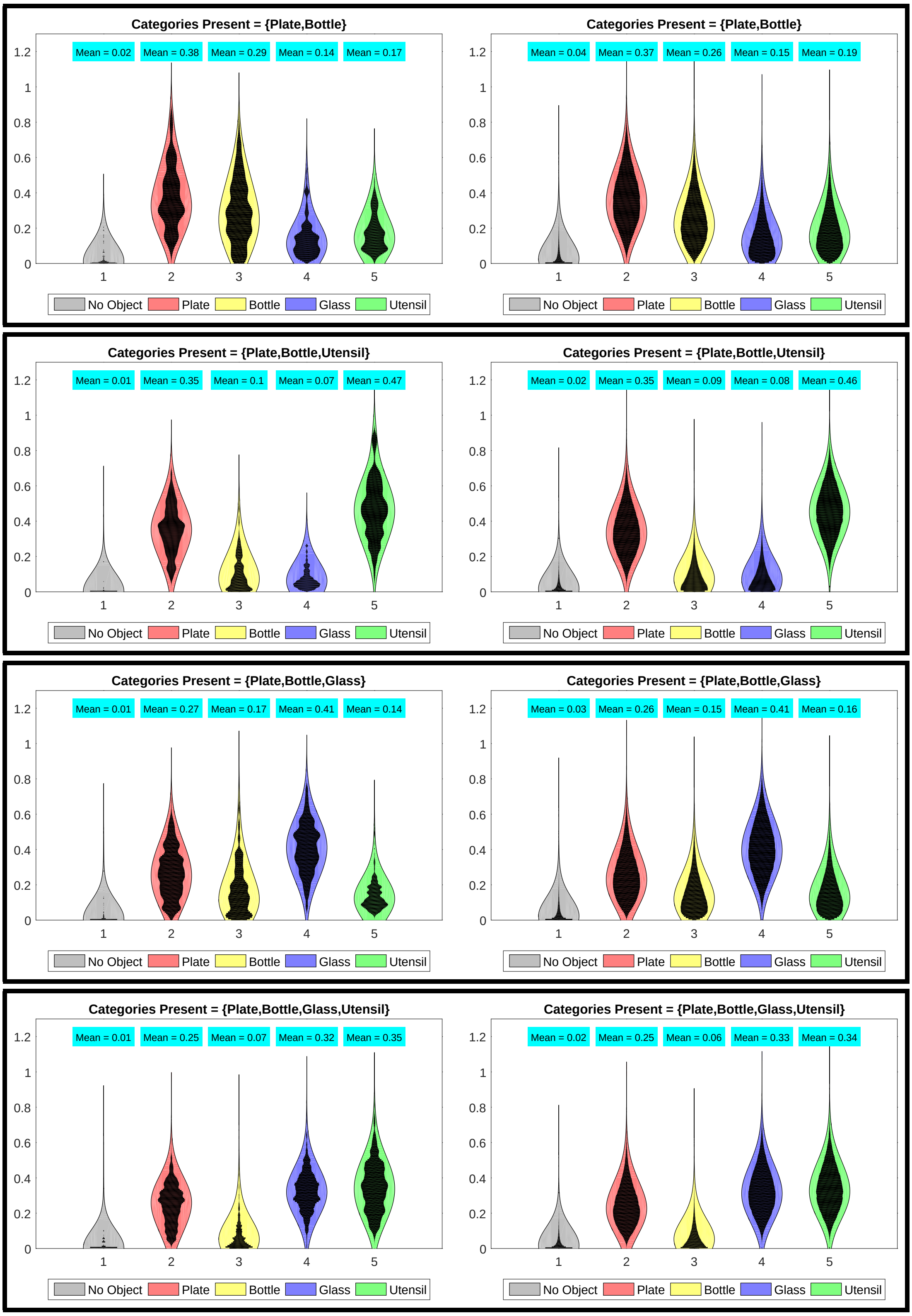


Figure 5.37: CatNet output and Dirichlet dist. bean plot.

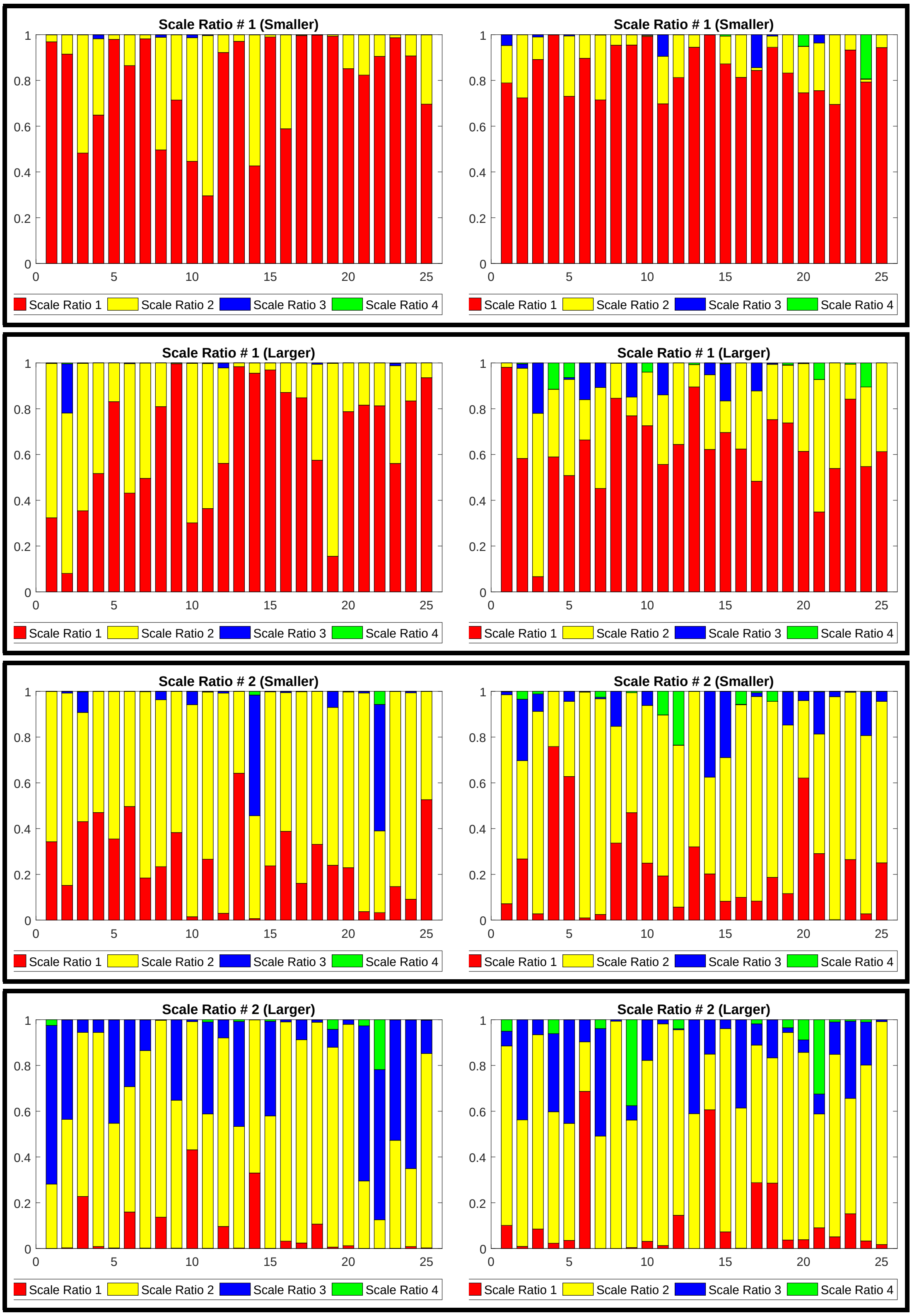


Figure 5.38: Stacked bar visualization of samples from ScaleNet output (left) and Dirichlet model (right).

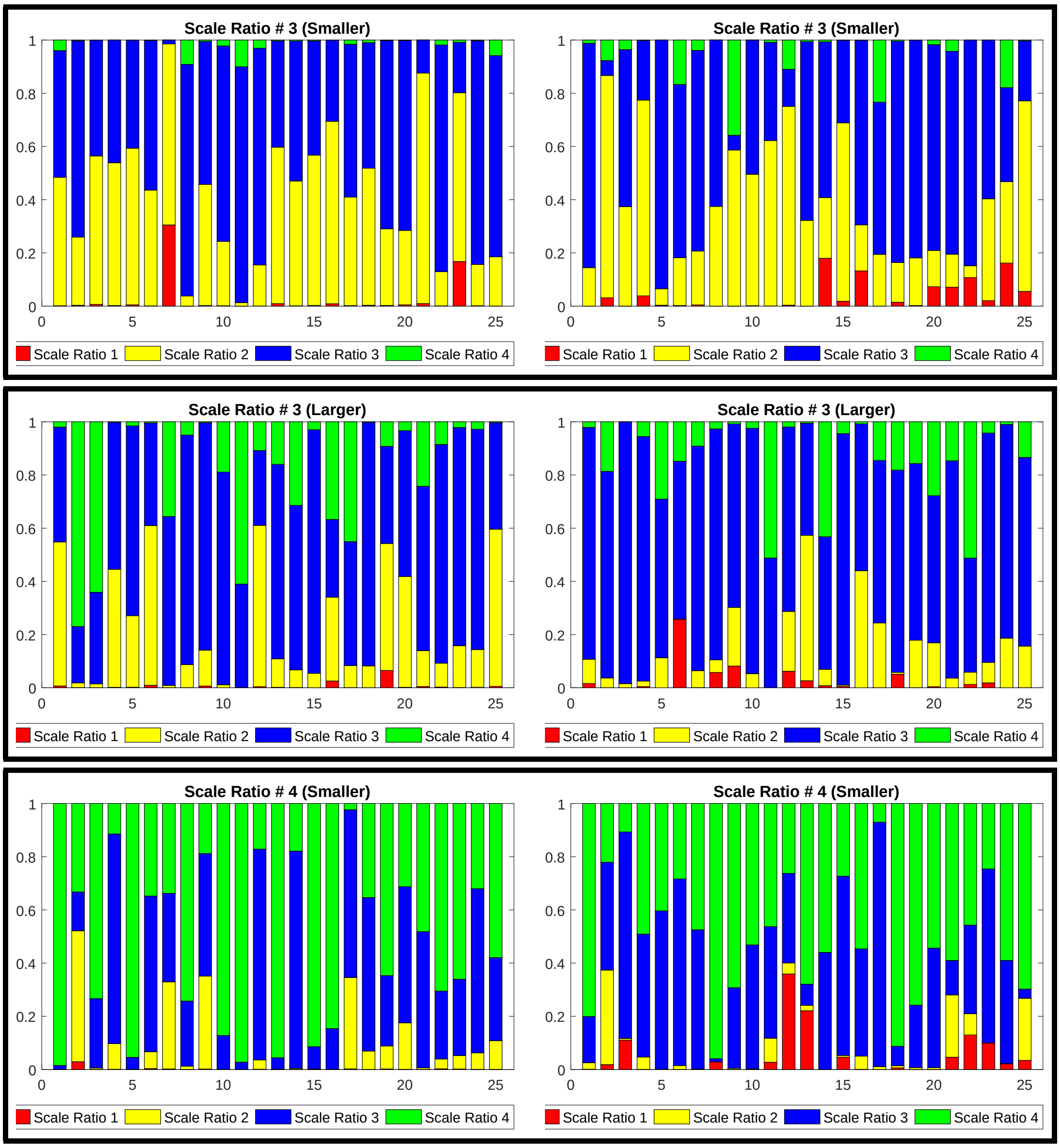


Figure 5.39: Stacked bar visualization of samples from ScaleNet output (left) and Dirichlet model (right).

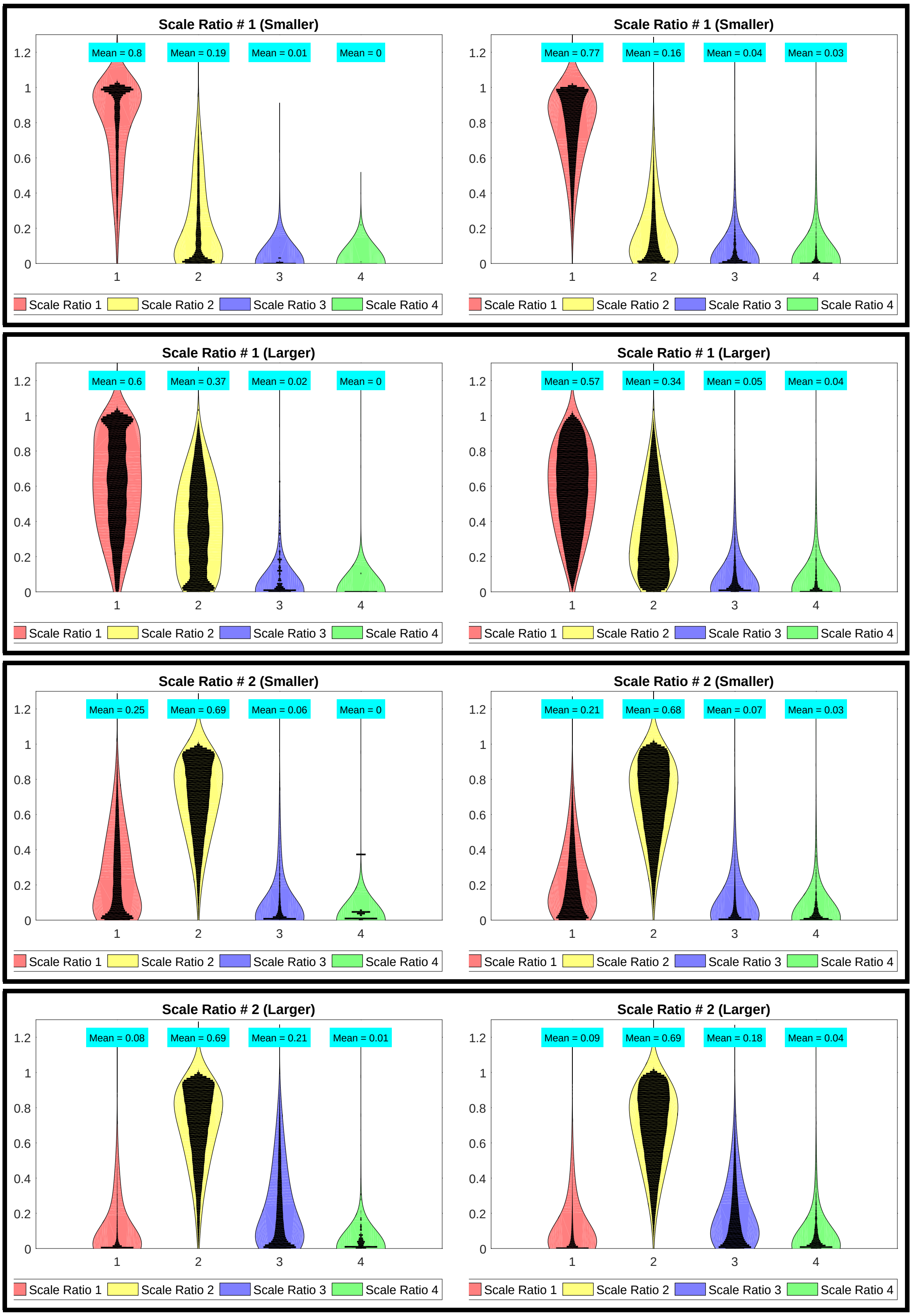


Figure 5.40: ScaleNet output and Dirichlet dist. bean plot.

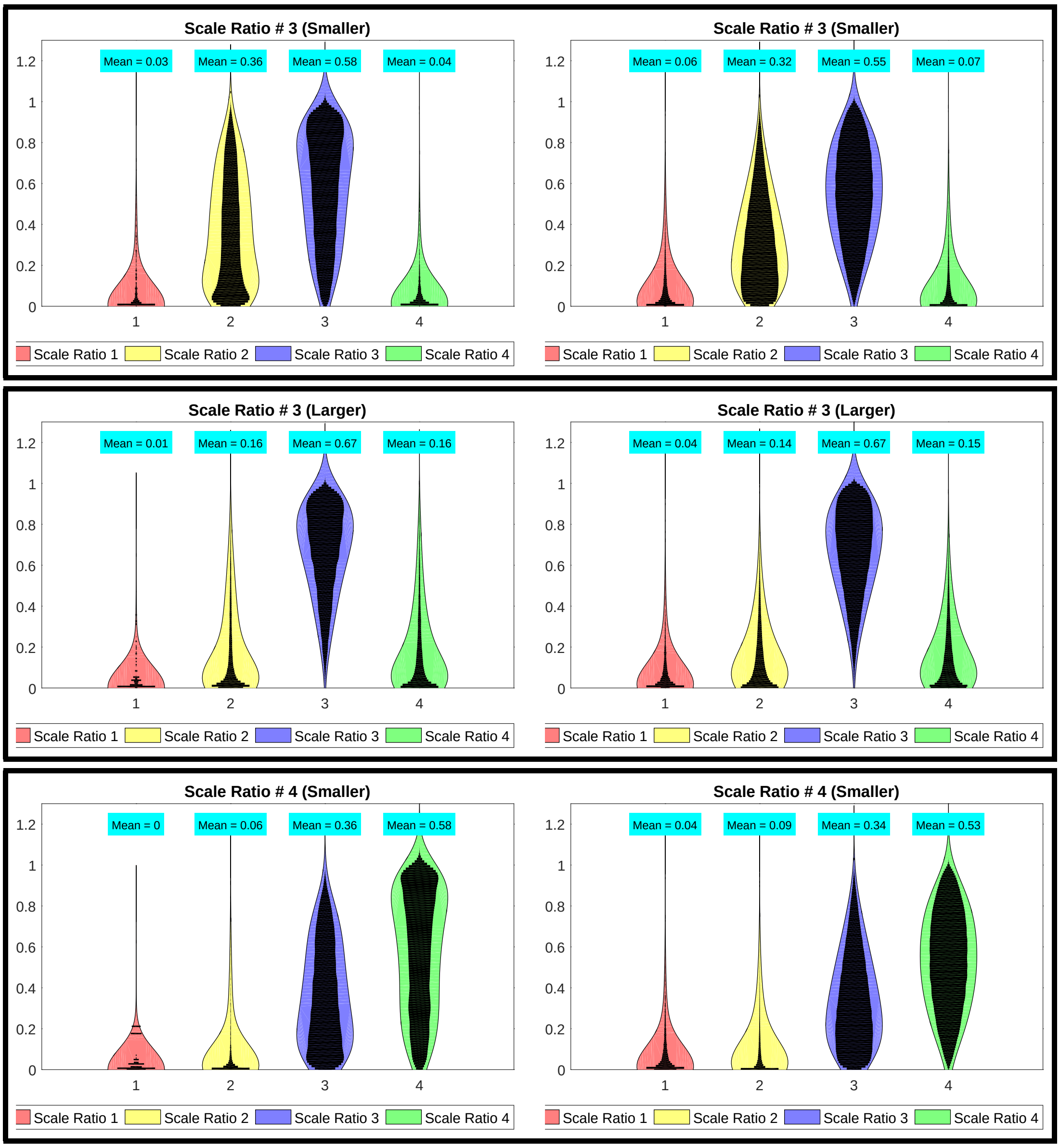


Figure 5.41: ScaleNet output and Dirichlet dist. bean plot.

Chapter 6

Entropy Pursuit Experiments

Entropy pursuit is an adaptive and sequential search strategy in the Bayesian framework that is inspired by the "Divide-and-Conquer" and "Coarse-to-Fine" search. We use the entropy pursuit (EP) approach for multi-category object recognition by collecting bits of evidence about the scene (encoded using some interpretation units) in an optimal order guided by the principle of uncertainty reduction. The idea is to sequentially select and run tests whose answer deliver the most amount of information (in an information-theoretic sense) given the history of collected evidence at each step. The EP search relies on a suitable prior model on the interpretation units sequentially fed with the collected evidence at each step to update the posteriors. Each question (test) is associated with one patch from the input image and is "What object categories exist in the patch?". The answer to this question is encoded by an "annoint" interpretation unit and predicted using the CNN classifiers.

We perform conditional inference (on the posterior distribution) given the accumulated evidence to select the most informative questions. To do this, we used an approach similar to the MCMC (Markov Chain Monte Carlo) sampling discussed in section 4.2 for inference, yet slightly modified for faster inference, and to address a problem in the calculation of acceptance ratios for a newly proposed table geometry. According to (4.12) the acceptance ratio requires computing the partition function (normalizing factor) for the MRF model since the factor $p_\lambda(\omega \mid T)$ appears in both the numerator and denominator of the ratio for different T and therefore does not cancel out. Hence, instead of using the Metropolis-Hastings table geometry sampler in the outer loop, we take advantage of the estimated table size $\hat{T}$ described in section 5.1.3 as follows. We learned 10 MRF models, for square tables whose sizes are 20cm apart; the sizes in meters are $T_S = \{0.9, 1.1, 1.3, 1.5, 1.7, 1.9, 2.1, 2.3, 2.5, 2.7\}$. We find the closest size from T_S to the estimated table size $\hat{T}$ and loop over this size and, if existed, one size smaller and one size larger from T_S during the outer loop integration of samples; note that for table sizes 0.9 and 2.7 there is no smaller and no larger table size, respectively. Given the relatively unbiased and Gaussian–shaped histogram of errors made by the table estimator shown in Figure 5.26, this procedure seemed promising. Then, given each table size, we sampled 10 homographies which are consistent with the detected table surface area (described in section 5.1.3); finally, we sampled from the MRF model whose 3D samples are then projected to the image coordinate system using the homography samples. Conditional inference for each step of EP took approximately 30 seconds $\big($3 table sizes (outer loop) × 10 homography samples (middle loop) × 1 (time took for MRF sampling in the inner loop)$\big)$. Note that we used the data model described in section 5.2 during the above inference based on MCMC sampling.

We ran EP on a dataset of 284 images. In each step of EP, two questions were asked, *i.e.,* two patches were processed by CNNs. The choice of two for the "batch" size was arbitrary. The questions selected at the early steps of EP are usually from the coarser levels of the annocell hierarchy since the larger annocells are more likely to include multiple instances from different object categories that can lead to a more uniform distribution across the 16 possible annoint configurations, *i.e.,* higher entropy. Figure 6.2 shows the annocells selected in the first four steps of EP for a given test image. As a reminder, the annocell hierarchy has 4 resolution levels whose patch size at the level-l is $D_{\text{max}}/2^l$ where $l \in \{0, 1, 2, 3\}$, and D_{max} is the longest side of the input image (see Figure 2.6). Note that the coarsest level annocells, basically the whole image, is not usually selected since its annoint distribution is expected to be concentrated on the few annoint configurations for which most of the categories are present. The general trend is that as EP proceeds finer annocells will be increasingly selected; but it is completely plausible, and actually happened during our experiments, to go back again to a coarser question after asking a sequence of finer questions; this is also observed in the human visual system (HVS) through the act of visual attention, where we usually start with investigating the scene coarsely to get a general idea, and then focus on finer details; but we may switch to a coarser (more general) investigation of the scene after collecting some finer details during the process. Coarse-to-fine search is one of the most interesting byproducts that comes naturally from the EP scene analysis; This results in a framework mimicking the visual attention property of the HVS. Figure 6.3, 6.4 show the selected annocells at later EP steps. We can see that the patches selected later are from the finer levels.

We never ask a question twice. Since the conditional entropy of a already asked test is zero, repeating a question would never occur if we used the original selection criterion $q_k(\mathbf{e}_{k-1}) = \operatorname{argmax}_{1 \leq i \leq K} H(X_i|\mathbf{e}_{k-1})$. However, we are using the the "oracle" approximation, which replaces $H(X_i|\mathbf{e}_{k-1})$ by $H(Y_i|\mathbf{e}_{k-1})$ in the EP criterion, consequently repetition of questions is not automatically avoided, and hence, should be imposed. Nonetheless, since we know the posterior will not change by repeating a question, we impose this constraint. Figure 6.5(top) shows the average conditional entropy of selected questions (averaged across the 284 processed images) during 100 EP steps; the ripples with period two in this figure are due to the EP batch size. Figure 6.5(bottom) shows the average conditional entropy of selected questions subsampled at rate two.

Turning to performance evaluation, an object instance may not necessarily fall completely inside any cell from our annocell hierarchy at a certain level even if there might exist a patch of the same size outside the hierarchy that completely includes that object instance. This is because the annocell hierarchy is constructed with 25% overlap between neighboring cells at the same level of resolution, and can therefore "miss" some object instances at a given level even if the cell size is large enough to include the object (*e.g.,* see Figure 6.1). The only way to avoid this is to make the hierarchy exhaustive at each resolution, *i.e.,* shifting patches by only one pixel at the time.

We define the "best-fitting annocells" for an object instance to be an annocell that meets the following three conditions: (1) it includes the object completely, (2) the ratio of the object's scale to the annocell size is maximum among all of the annocells satisfying the first condition, and (3) the distance between the center of the object instance and the center of the annocell is minimum among all of the annocells satisfying the second condition. Figure 6.6 shows example best-fitting annocells for instances from the three existing object categories plate (in red), glass (in blue), and utensil (in green). During performance evaluation we declare a detected annocell to be a true positive for an object instance if the instance's best-fitting annocell of the instance and the detected annocell are "close neighbors". Two annocells are defined as "close neighbors" if they are within one resolution level apart in the annocell hierarchy and also at least 50% of the smaller annocell is covered by the larger annocell.

Since the size of cells from level-i of the hierarchy is twice the size of cells from level-$(i+1)$, the best-fitting annocell for an object instance with scale S (in pixels) is normally expected to be from the level with cell size W (in pixels) that satisfies $0.5 < S/W \leq 1$. However, as noted before, due to the sparsity of the hierarchy, the condition $0.5 < S/W \leq 1$ might not be satisfied by any annocell in the hierarchy: the cells that satisfy $0.5 < S/W$ may not include the object instance and the coarser level cells may not satisfy $0.5 < S/W$. Therefore, if we cannot find an annocell satisfying $0.5 < S/W \leq 1$, we will check for annocells that satisfy a relaxation of the original condition: $0.375 < S/W \leq 1$. Figure 6.1 shows an example where a plate instance is not captured by annocells from a resolution

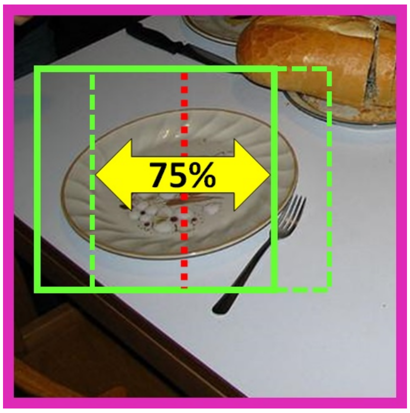


Figure 6.1: An example showing a plate instance not captured by annocells from the possibly best-fitting level (in green).

level whose cells $0.5 < S/W \leq 1$; two adjacent cells are shown by green cells. An object instance may not be captured by cells from the potentially best-fitting level if its scale is larger than 75% of the cells' size; it is more likely to be missed by the potentially best-fitting cells if the object is centered closer to the lines in the middle of adjacent cells (*e.g.,* the dashed red line in Figure 6.1). Note that the reason for choosing 0.375 in the relaxed condition is that if the scale of an object is larger than 75% of the cells from the potentially best-fitting level, its scale must be larger than 37.5% of one-level coarser cells ($0.75 \times 0.5 = 0.375$). If the object instance is missed at the potentially best fitting level of the annocell hierarchy it will certainly be captured by a cell from a one-step coarser level.

To make the best fitting model-based-detections given an accumulated evidence we first sample the posterior distribution using MCMC sampling described earlier in this chapter; each posterior sample gives the category label and 2D pose of object instances *i.e.,* a 2D scene. Then, for each posterior sample, we evaluate a new set of binary variables defined similar (in domain and range) to the annobits whose only difference with annobits is that they must also satisfy $0.375 < S/W \leq 1$ (if no annocell satisfying $0.5 < S/W \leq 1$ was found) to be turned on; then, these samples are aggregated to obtain a posterior distribution on the new set of variables; we finally run non–maximum suppression on this posterior distribution for each object category to obtain the detections. Note that the above process is just a trick to obtain the best-fitting cells considered as bounding boxes.

Figure 6.7 shows Precision–Recall curves for 13 different methods that we ran on the data set of 284 images for all object categories. As a reminder, the precision and recall are:

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}, \tag{6.1}$$

where TP, FP, and FN denote, respectively, the number of true positives, false positives, and false negatives (see Figure 6.8). Note that we would like to detect as many true instances as possible (high recall) for as few mistakes as possible (high precision or false detection rate) which invariably necessitates a trade-off. Here, we focus on achieving high recall in the following analysis of the results. According to Figure 6.7 the EP model-based detection performance improve as more classifiers are run and incorporated into the model until step 30 (= 60 questions), but after step 30 the performance starts to worsen *e.g.,* see the P-R curve for step 70 (= 140 questions). Figure 6.9 shows only three P-R curves, selected out of the 13 curves in Figure 6.7 for higher clarity, associated with detections based on the full posterior (including all questions/answers), entropy pursuit after step 30 (including 60 questions/answers), and prior model. The full posterior seems to have the worst performance which seems counterintuitive because we expect to achieve better performance by incorporating more evidence. However, achieving better performance should be expected only if we received accurate information. Assume that you are participating in a 20 questions game and after asking a certain number of questions you are almost sure about the final answer but you keep asking more questions and the host starts to give you wrong answers; this leads to doubts about your initial belief, which was perhaps correct, and to finally making a mistake. Consider Figure 6.12 where the 1st, 3rd, and 4th most confident plate detections are actually not a plate but the top or bottom of glasses; all of the annocells corresponding to these detections are from the finest level of the annocell hierarchy that are expected to be chosen later during the EP selection criterion and potentially degrade detection performance. Note that the model integrates the ScaleNet outputs in an attempt to suppress configurations with scale inconsistency (*e.g.,* the incorrect plate detections in Figure 6.12). However, since a multiplication of the outputs of CatNet and ScaleNet are incorporated into the model, the model may not be able to completely suppress such configuration if the output of one of the CatNet or ScaleNet networks is large enough to compensate for a smaller score from the other network.

Figure 6.13(top) shows the maximum conditional entropy of the marginal annoint distributions (averaged across the dataset of 284 processed images for each data point) versus EP step; as one can see, the maximum conditional entropy constantly decreases by integrating more evidence until step 20 (= 40 questions) which implies an increasing level of confidence in our detections. However, the maximum conditional entropy starts to increase at step 40 (= 80 questions) which implies a decreased level of certainty in our detections. Note that the difference between the entropy of selected questions in Figure 6.5 and the maximum conditional entropy in Figure 6.13(top) is the result of not repeating questions during EP and implies confusion in estimating the annoints that we have already investigated; the two figures start to differ considerably at EP step 20. Figure 6.13(bottom) shows the mean conditional entropy of the marginal annoint distributions (averaged across both the dataset of 284 processed images as well as across all of the annoints) versus EP step.

Figure 6.10 shows three curves selected out of the 13 P-R curves in Figure 6.7 which illustrates the result of model-based detection for two variations “Rand. 140 Q.” and “Rand. 30 Q.” with the same number of questions as in the two EP tests except that the questions are chosen at random; we have also included the result of CNN classifiers (no model) when 140 and 30 patches are randomly chosen and processed. One can see that the result with 140 randomly selected questions (the cyan curve in Figure 6.7) is almost the same as EP with only 10 questions asked (the yellow curve in Figure 6.7) which emphasizes the importance of efficient question selection in the Bayesian approach. The Bayesian approach provides a natural framework unifying the evidence collected from running tests and our prior knowledge encoding the contextual relations between different scene entities. This underlines the importance of the Bayesian approach. Our Bayesian approach to multi-category object recognition using the entropy pursuit search strategy selects the right set of patches which if processed provide the most information gain and uncertainty reduction on the interpretation units. Our tests shows that it makes significant difference to choose patches appropriately using our EP strategy versus randomly choosing them. In addition to saving the time it takes to process patches that do not provide much information, we can monitor the confidence of our detections (measured by conditional entropy) and stop processing more patches once the uncertainty starts to increase to achieve the best performance. The model-based approach with enough questions asked outperforms the CNN classifiers (higher precision at high recall area in the right); Figure 6.11 shows the P-R curves for detections based on the CNN classifiers, full posterior, and EP after 30 steps (60 questions); the results support the EP-based Bayesian approach. The result of running the CNN classifier on a small fraction of randomly selected annocells does not achieve high recall (see Figure 6.7).

Figures 6.14, 6.15, 6.16 show the detection results for full posterior, EP after 40 steps (80 questions), CNN classifiers, respectively. Figures 6.17, 6.18, 6.19, Figures 6.20, 6.21, 6.22, Figures 6.23, 6.24, 6.25, and Figures 6.26, 6.27, 6.28 show more examples. The threshold for positive detections using the model-based approaches was set to 0.4. One can find unlikely configurations that have been suppressed and likely configurations that have been encouraged using the model-based approach. For example, the top and bottom of glass instances in Figure 6.28 that were incorrectly classified as plate are not declared as plate as shown in Figures 6.26 and 6.27; or the salt/pepper shakers in Figure 6.19 that were incorrectly classified as bottle and glass using the CNN classifiers are not declared as bottle and glass using the model-based approaches as shown in Figures 6.17 and 6.18.

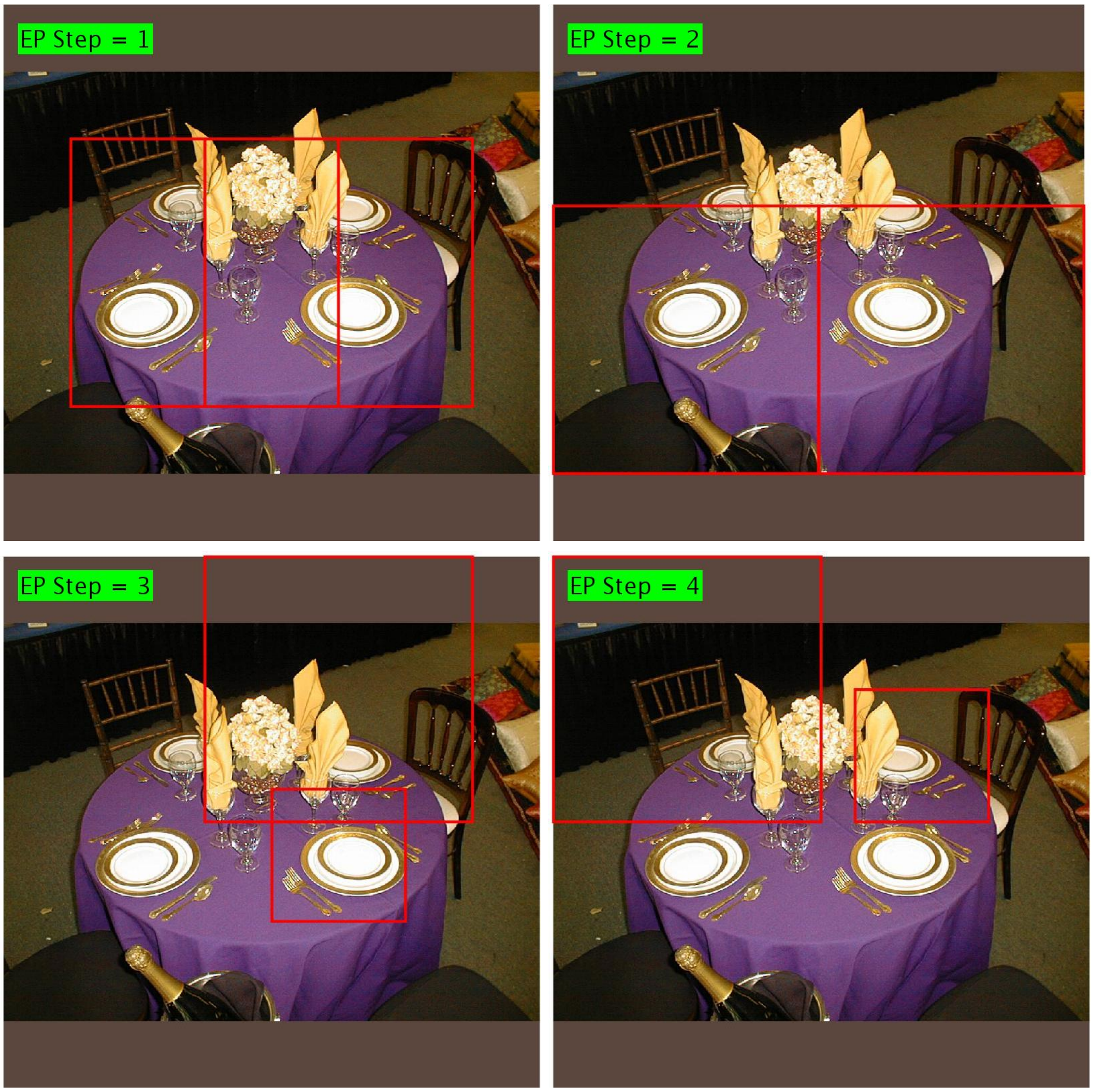


Figure 6.2: Questions at “early” EP steps. The annocells whose sizes are $1/2$ and $1/4$ the image’s size are from level-1 and level-2, respectively.

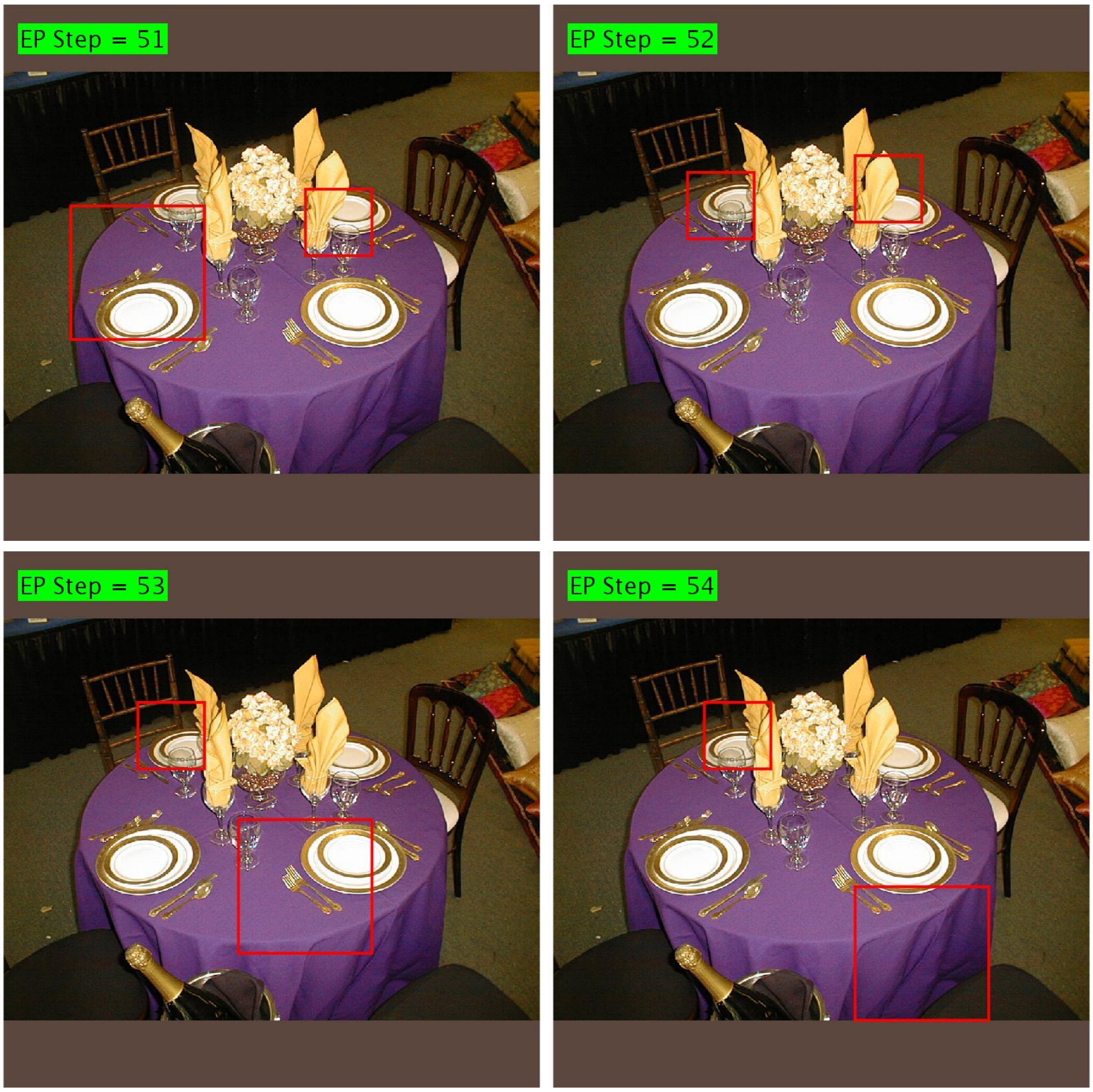


Figure 6.3: Questions at "middle" EP steps. The annocells whose sizes are $1/4$ and $1/8$ the image's size are from level-2 and level-3, respectively.

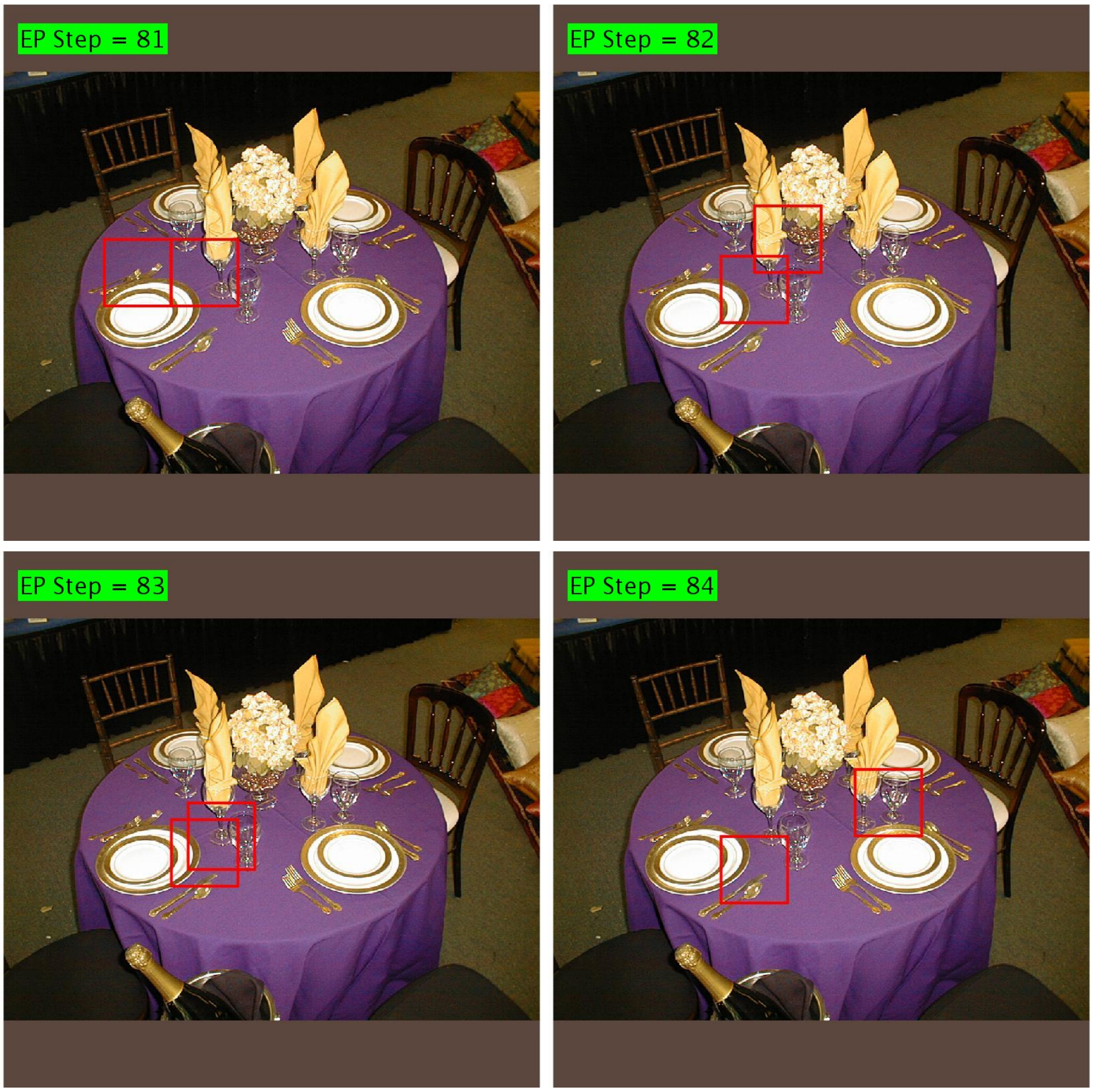
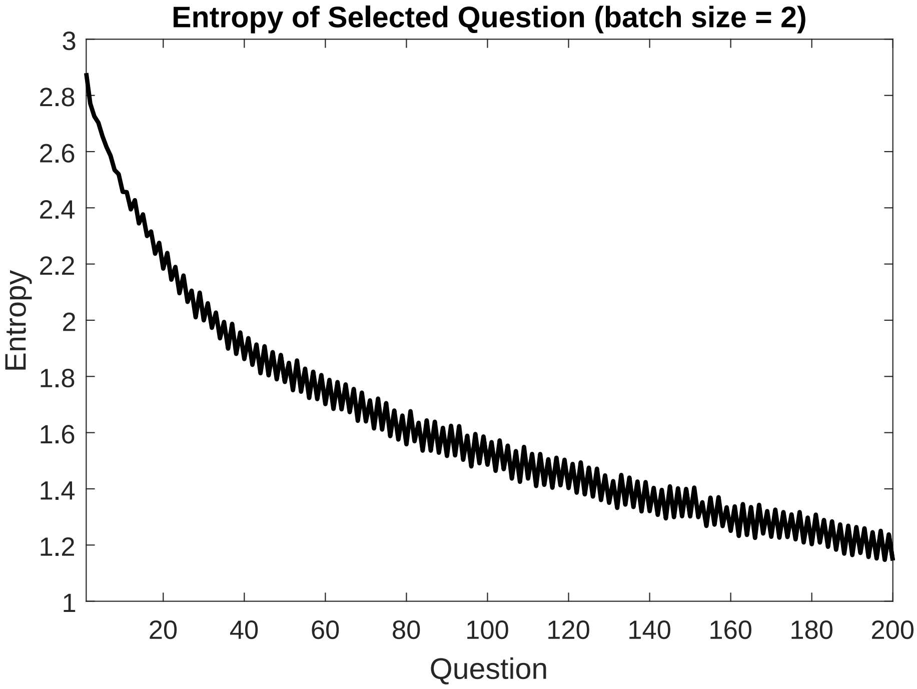


Figure 6.4: Questions at "later" EP steps. All of the annocells are from level-3 of the annocell hierarchy.



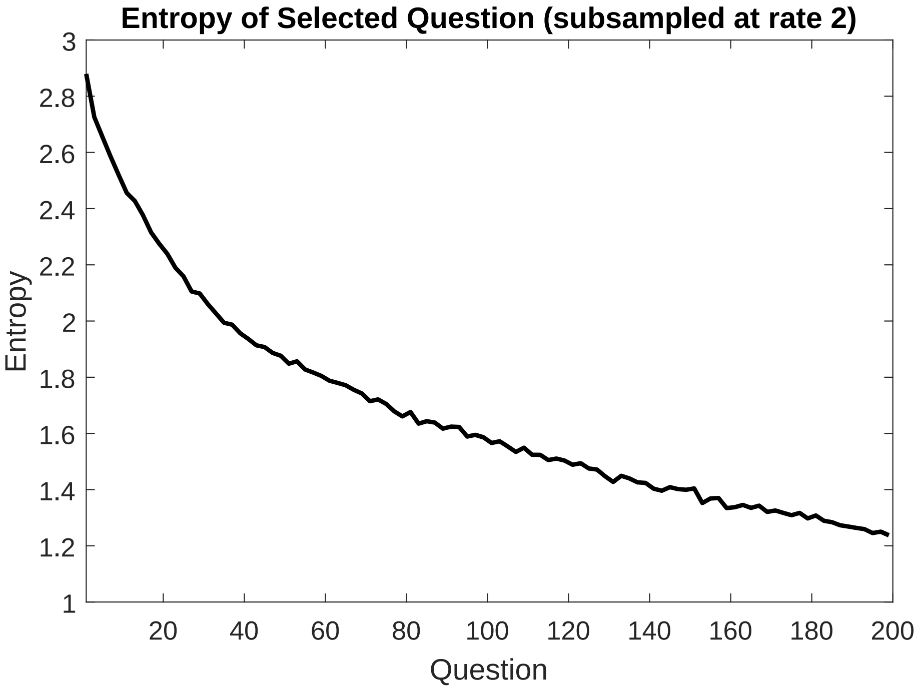


Figure 6.5: Entropy of EP Questions.

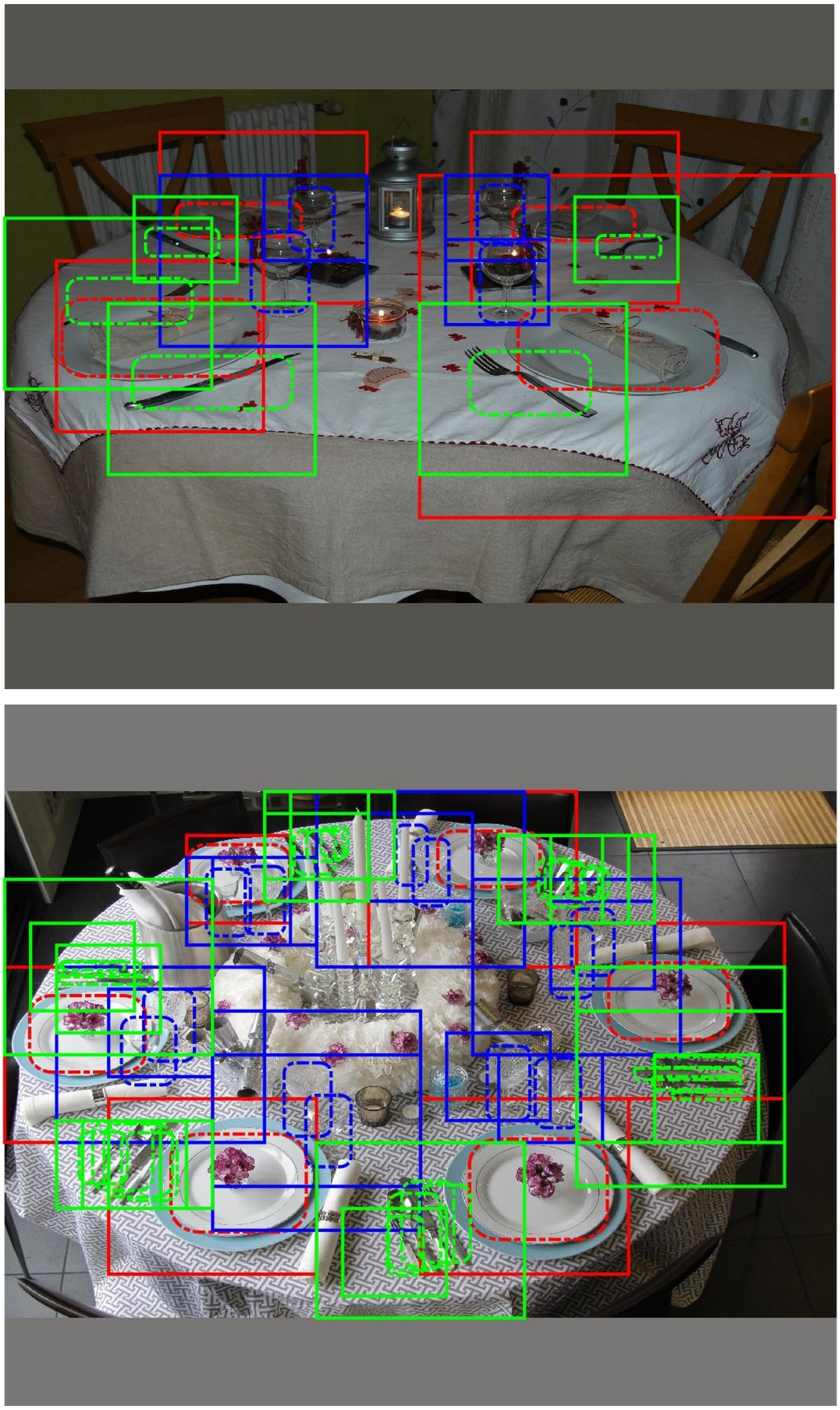


Figure 6.6: Some example fittest annocells shows by rectangles with solid lines.

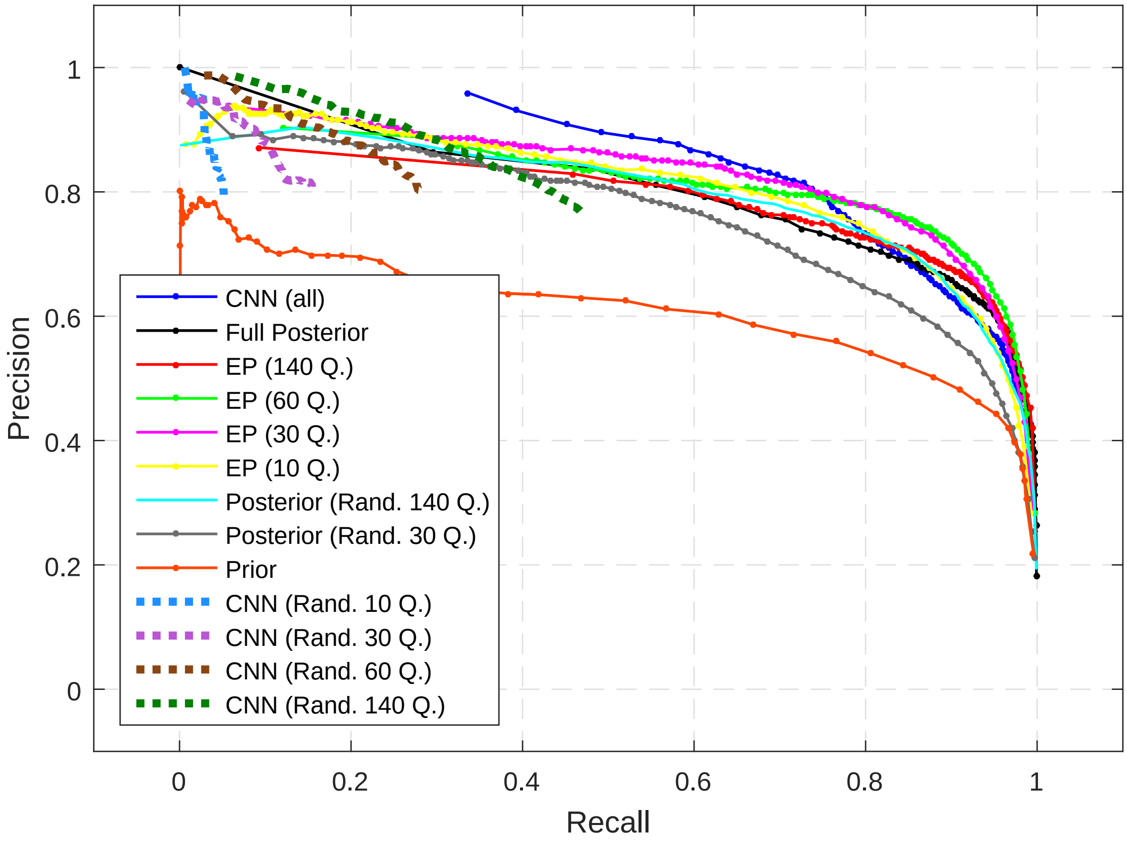


Figure 6.7: Precision-Recall Curves.

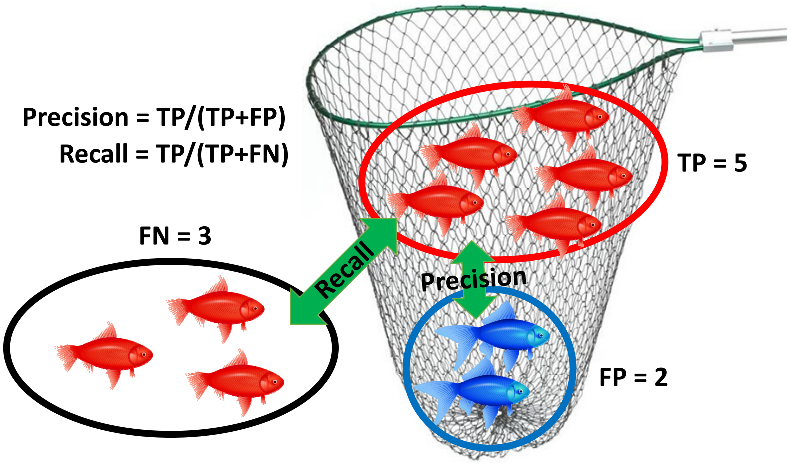


Figure 6.8: Precision-Recall Visualization. The net is supposed to catch all the red fishes (true class) but it catches some blue fishes (wrong class) and misses some red fishes (in the black circle).

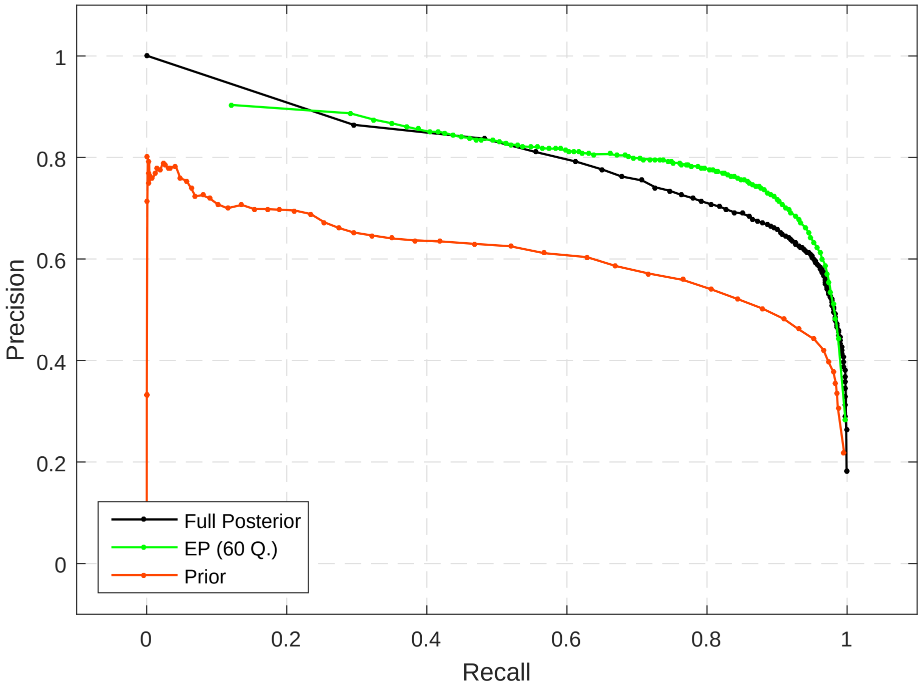


Figure 6.9: Precision-Recall Curves.

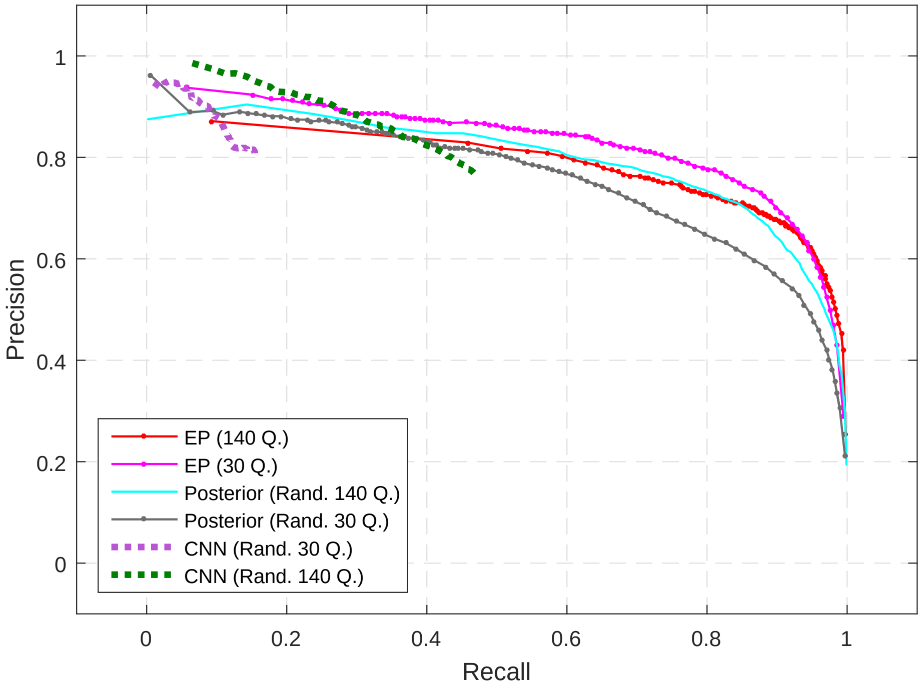


Figure 6.10: Precision-Recall Curves.

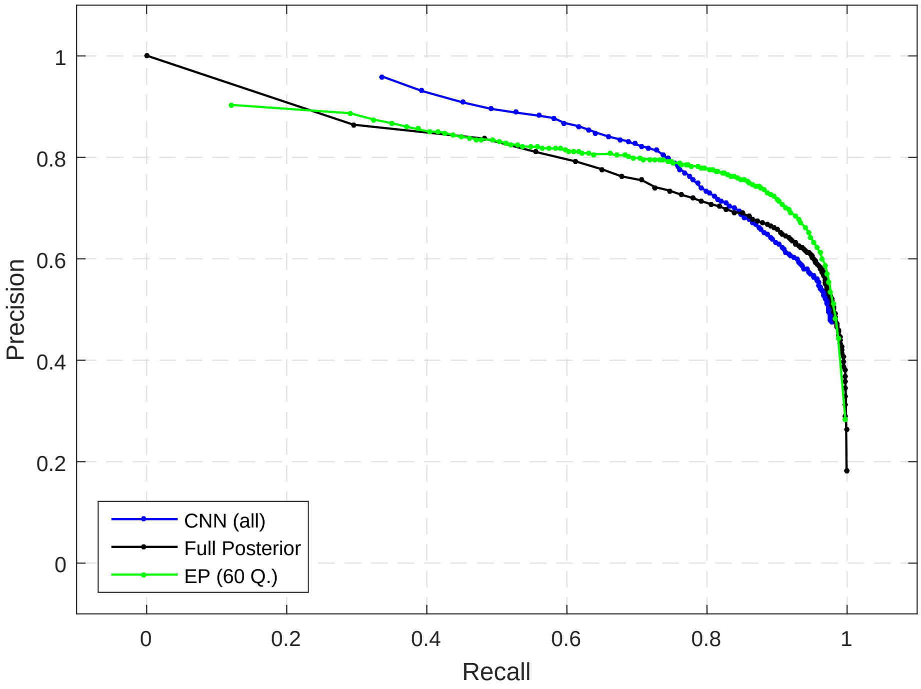


Figure 6.11: Precision-Recall Curves.

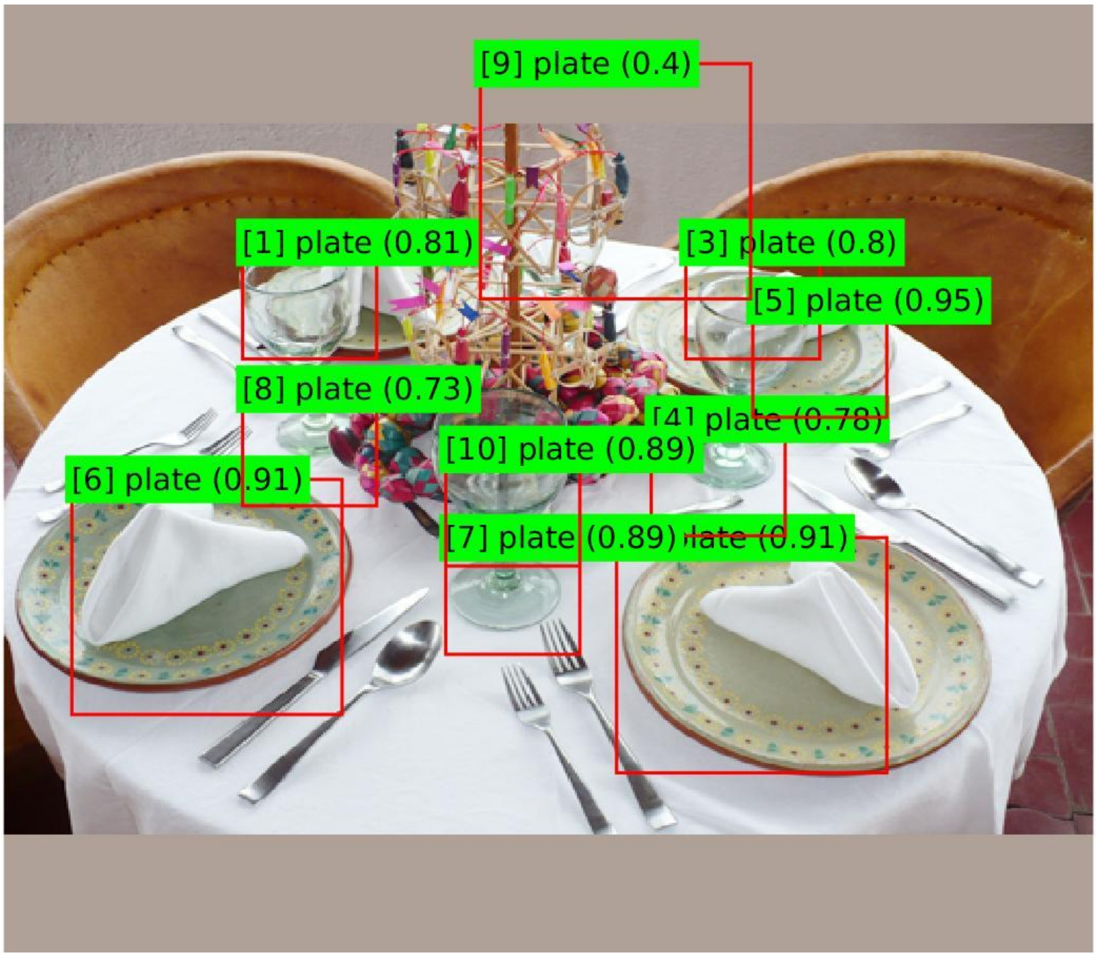
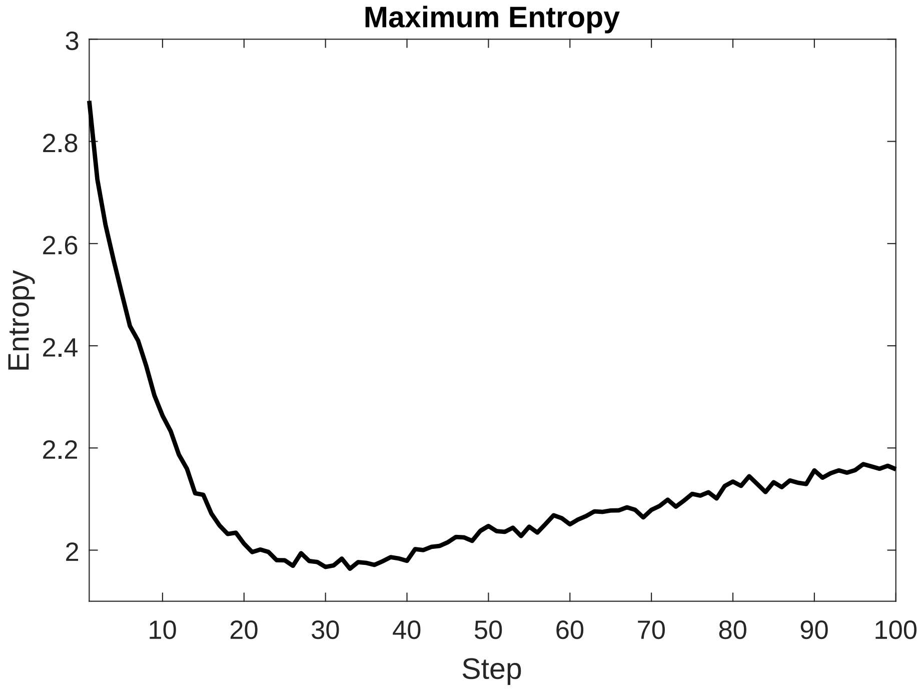


Figure 6.12: Confusing CNN detections example.



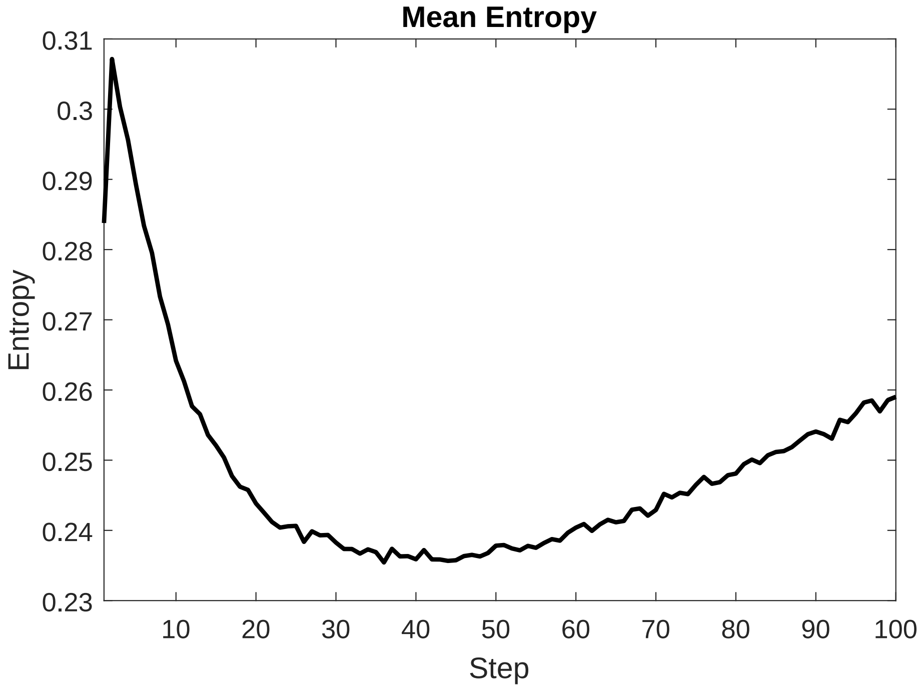


Figure 6.13: Maximum and mean entropy of the posterior distribution on annoints.

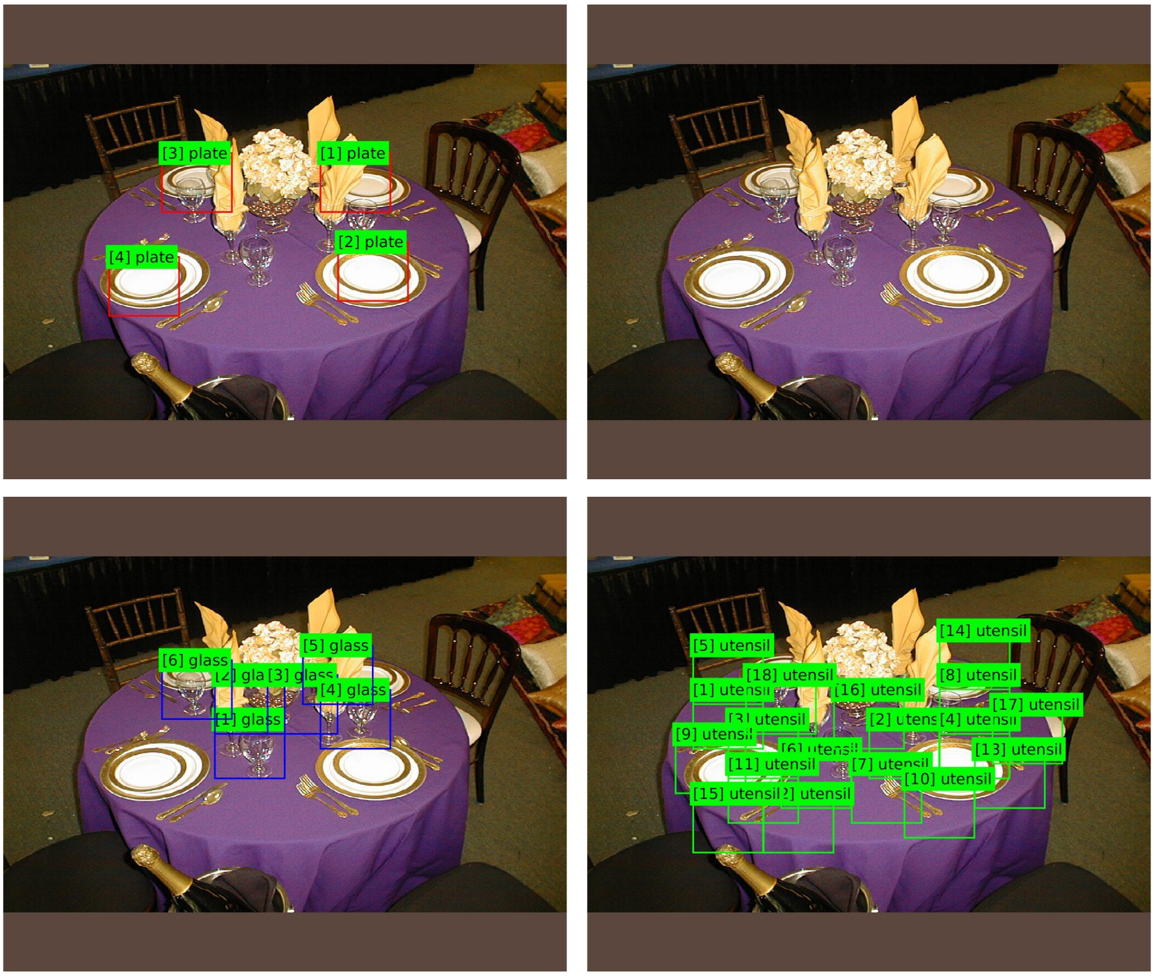


Figure 6.14: Model-based detections (all questions) for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

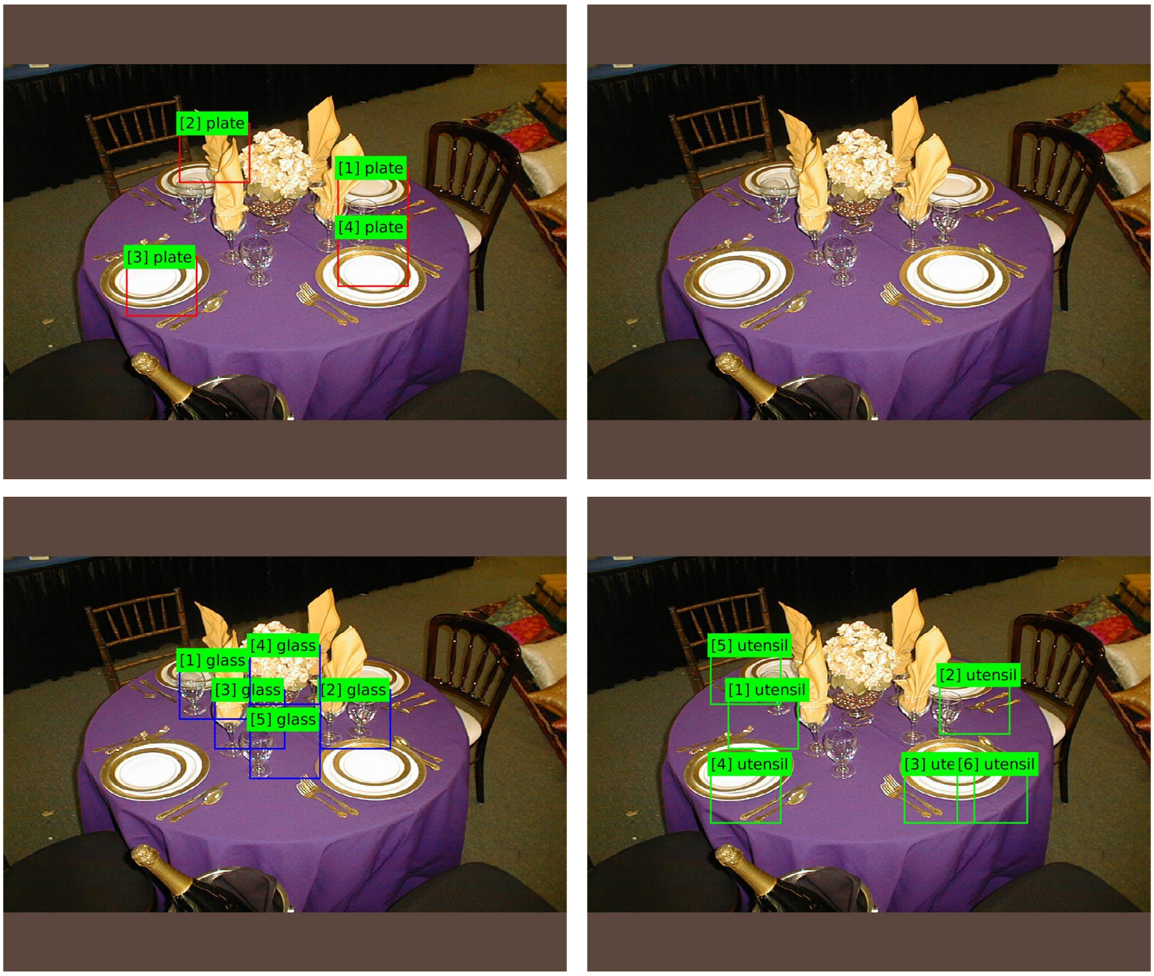


Figure 6.15: Model-based detections (EP 80-Questions) for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

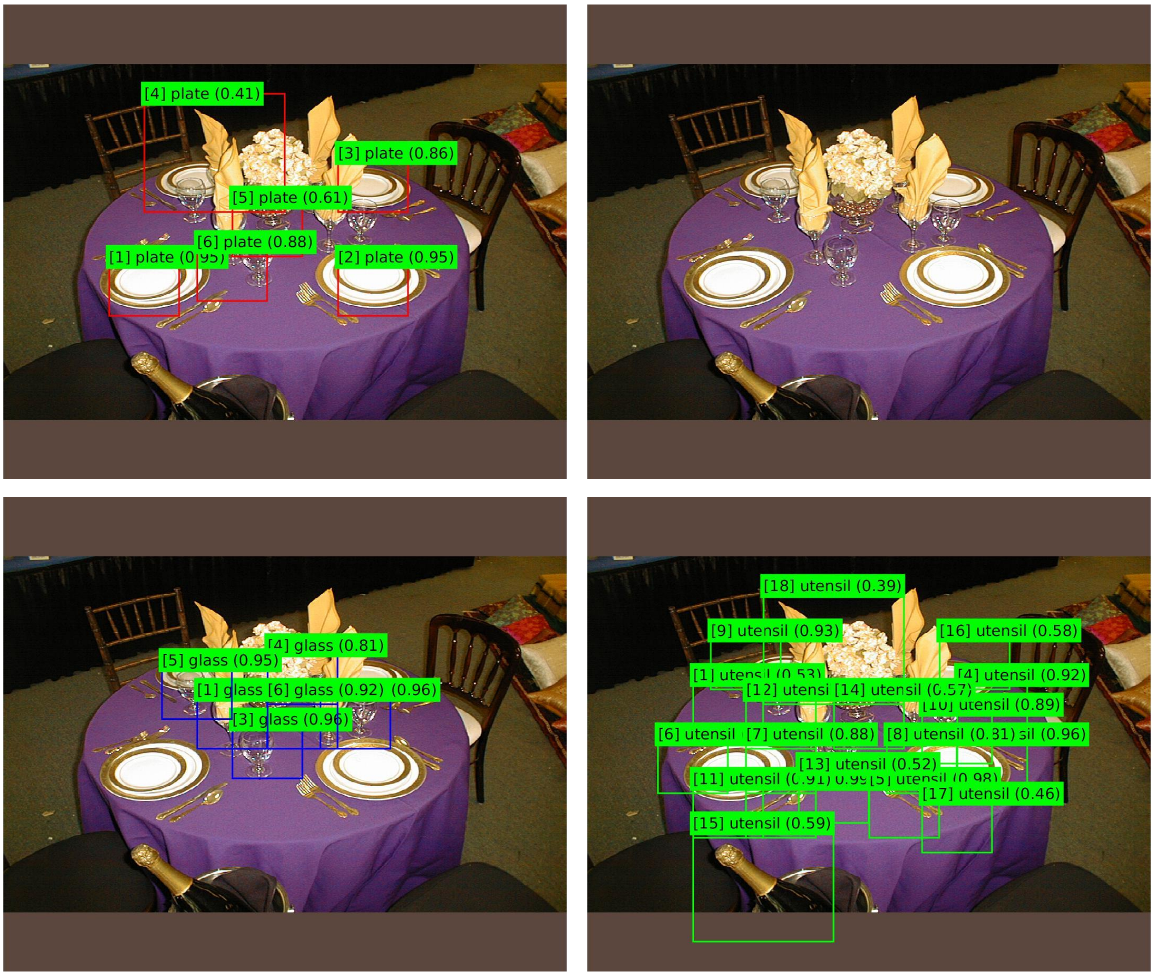


Figure 6.16: CNN detections for "plate" (top left), "bottle" (top right), and "glass" (bottom left), and "utensil" (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

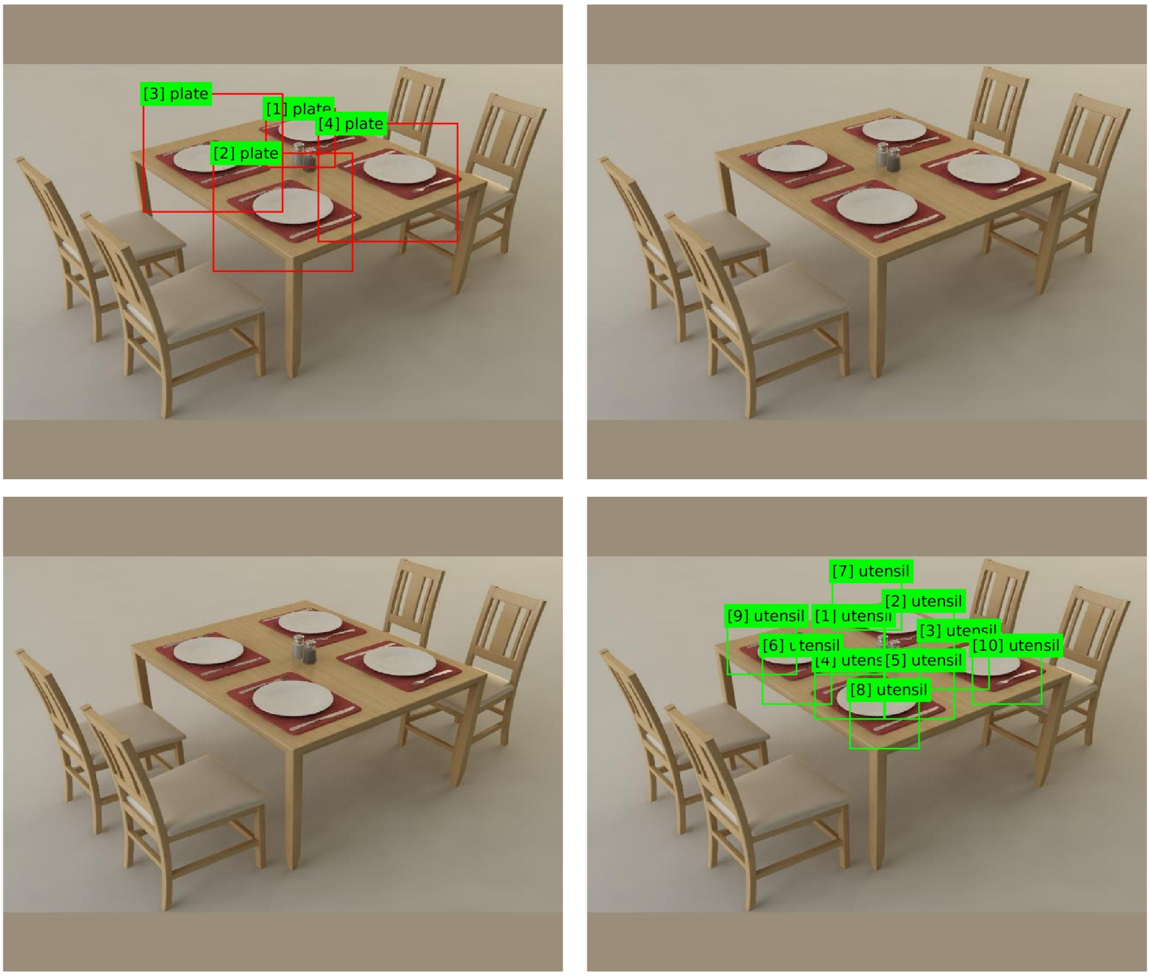


Figure 6.17: Model-based detections (all questions) for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

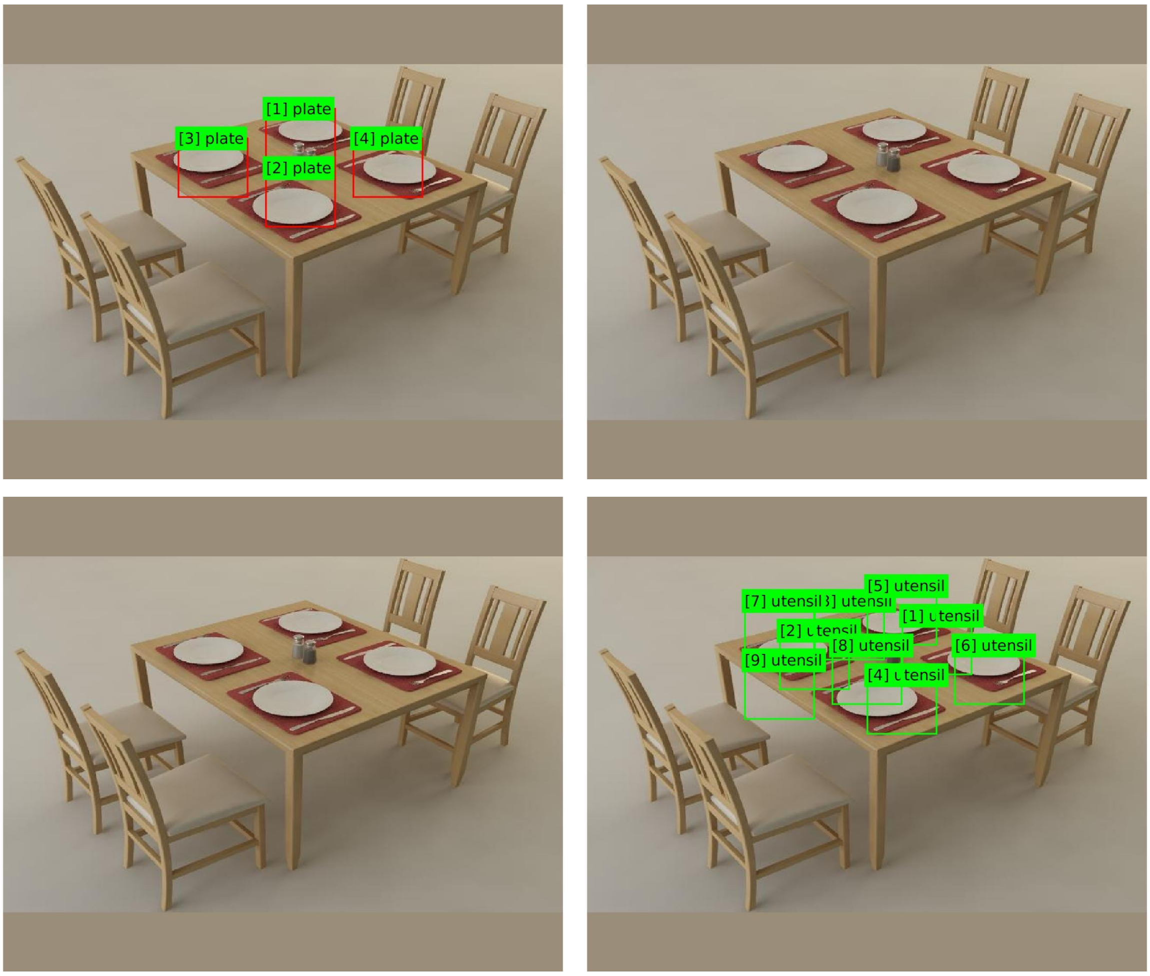


Figure 6.18: Model-based detections (EP 80-Questions) for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

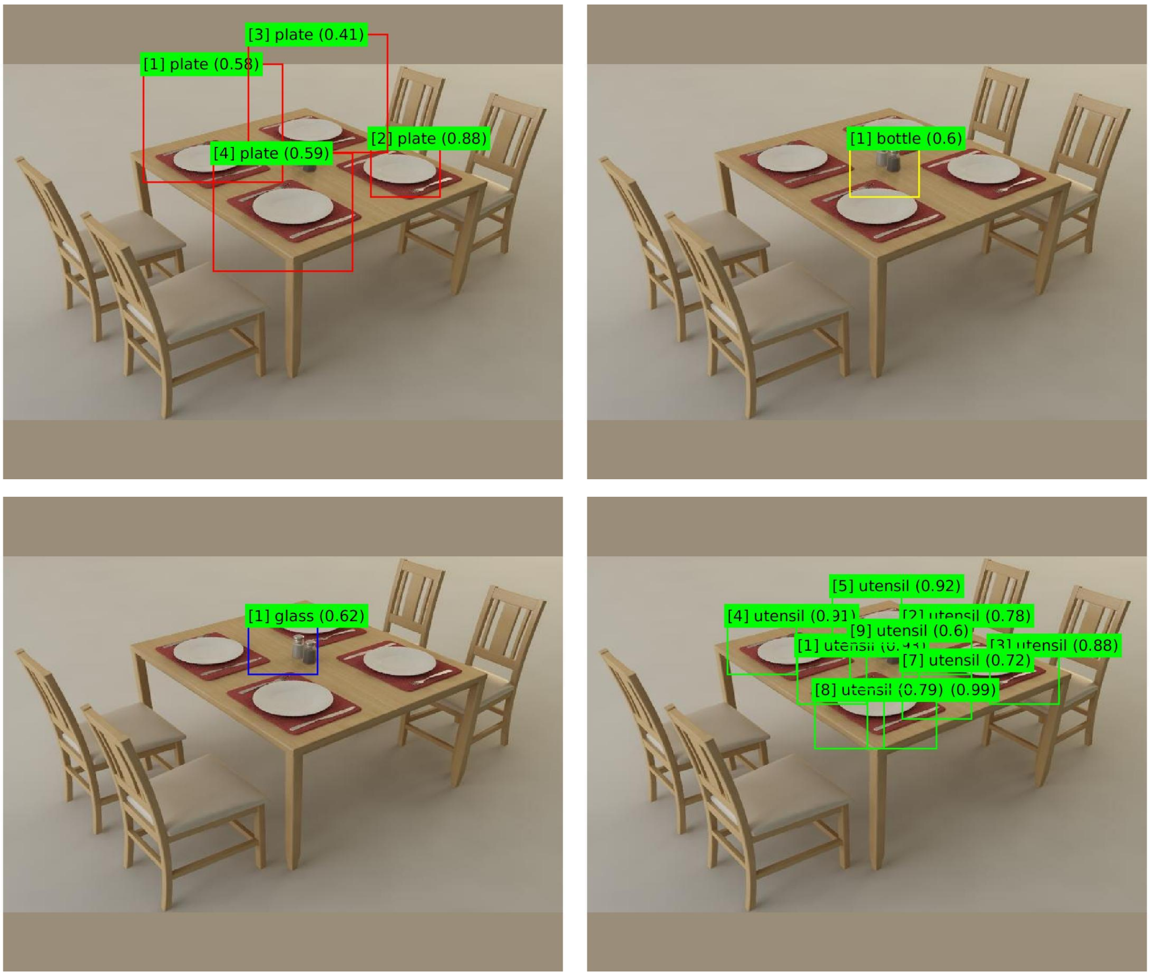


Figure 6.19: CNN detections for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

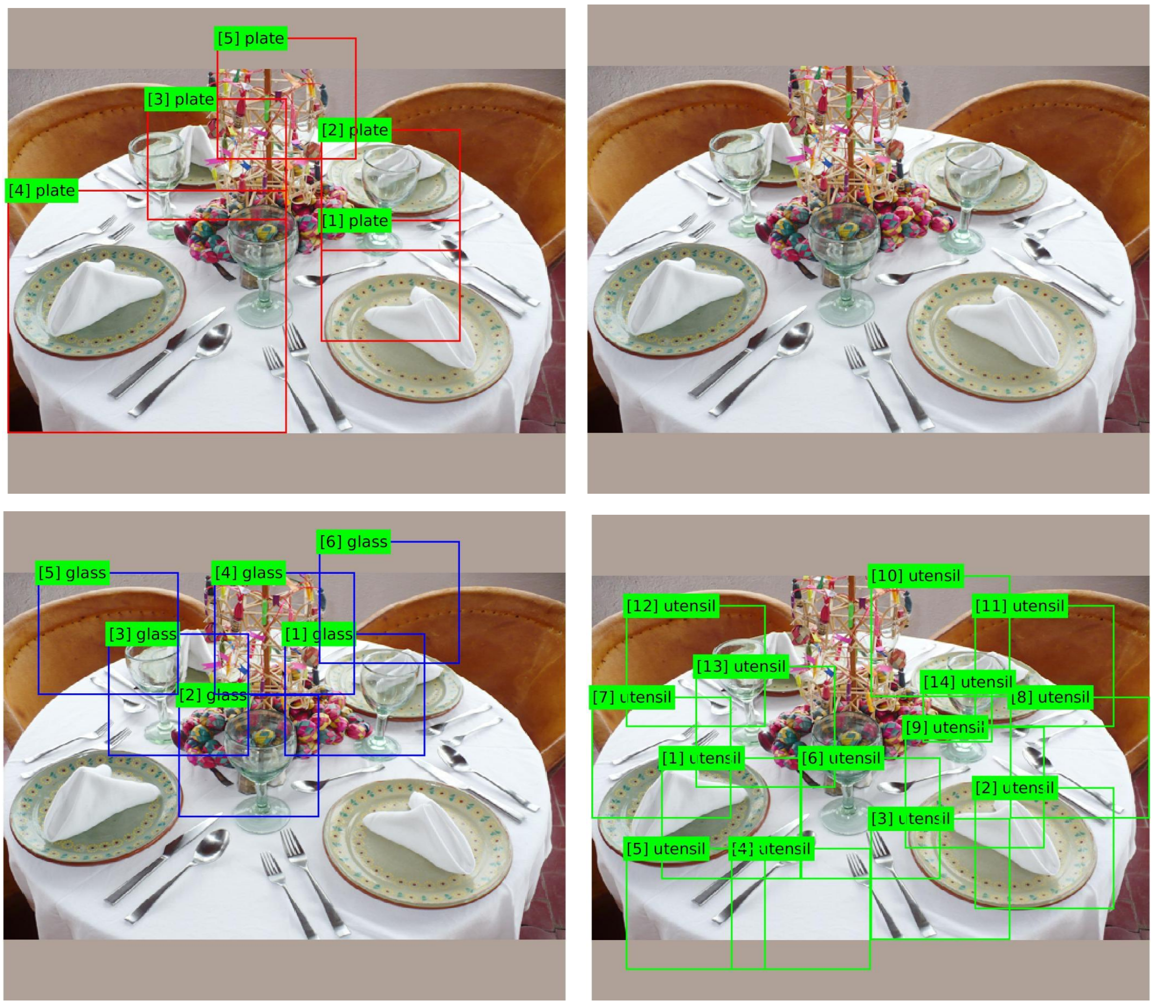


Figure 6.20: Model-based detections (all questions) for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

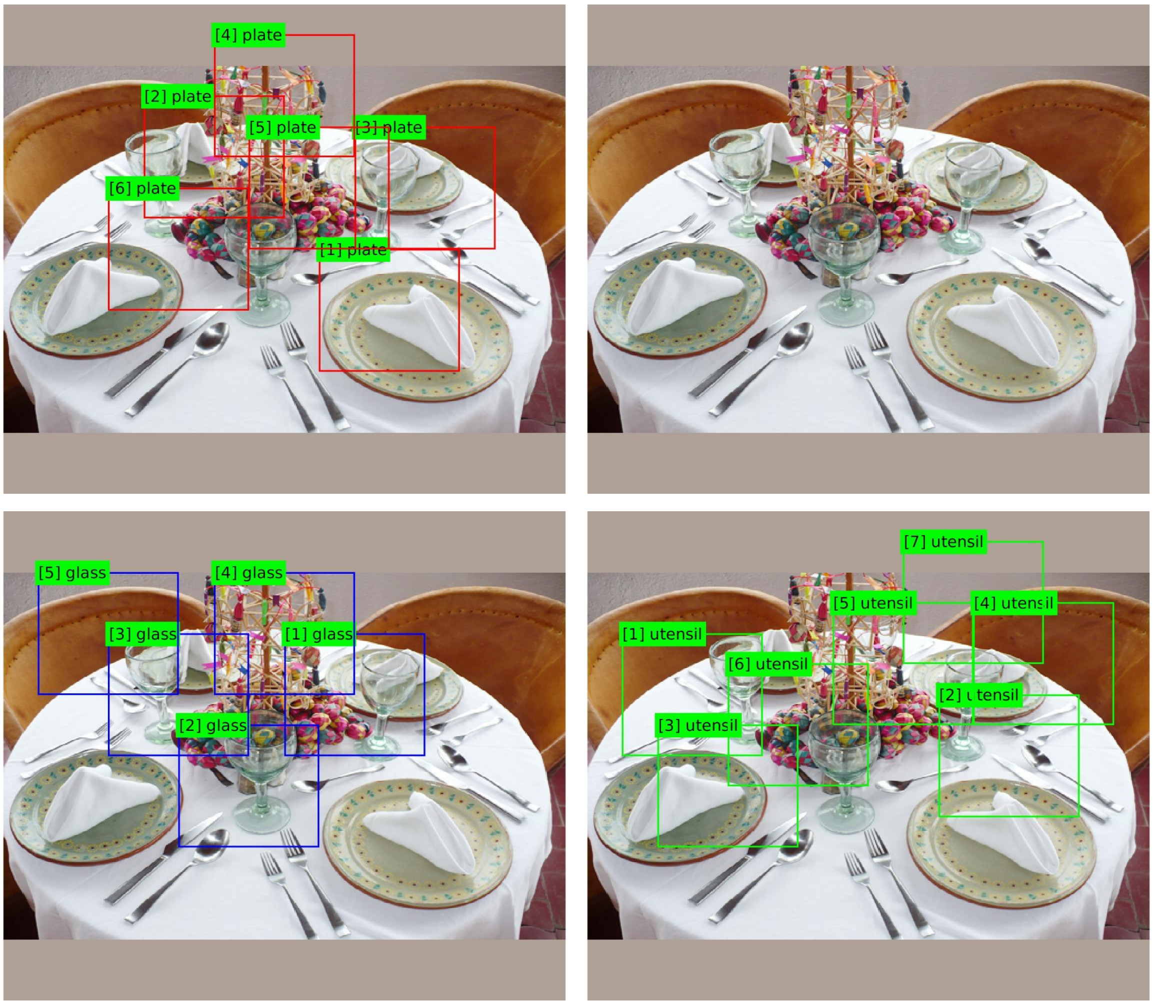


Figure 6.21: Model-based detections (EP 80-Questions) for "plate" (top left), "bottle" (top right), and "glass" (bottom left), and "utensil" (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

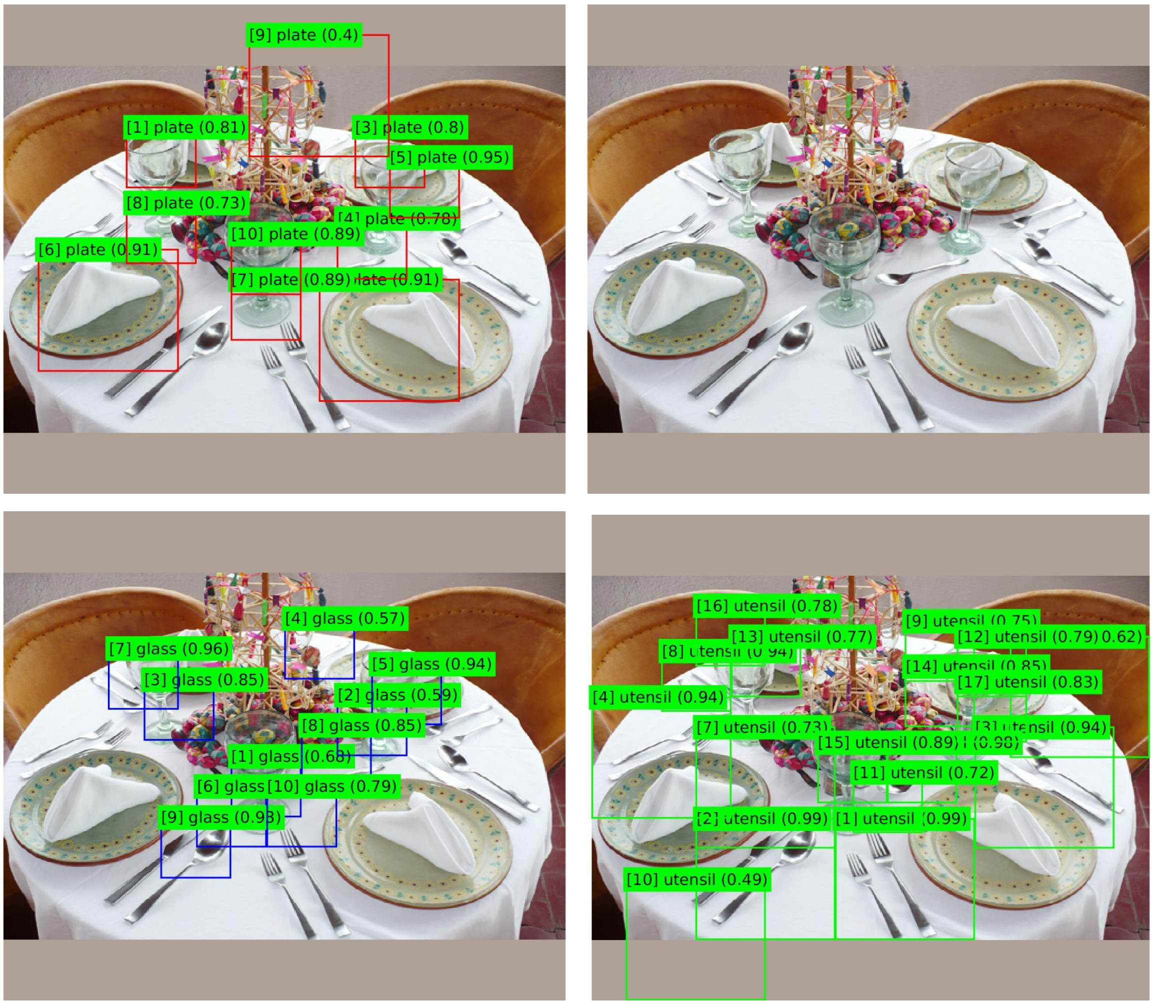


Figure 6.22: CNN detections for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

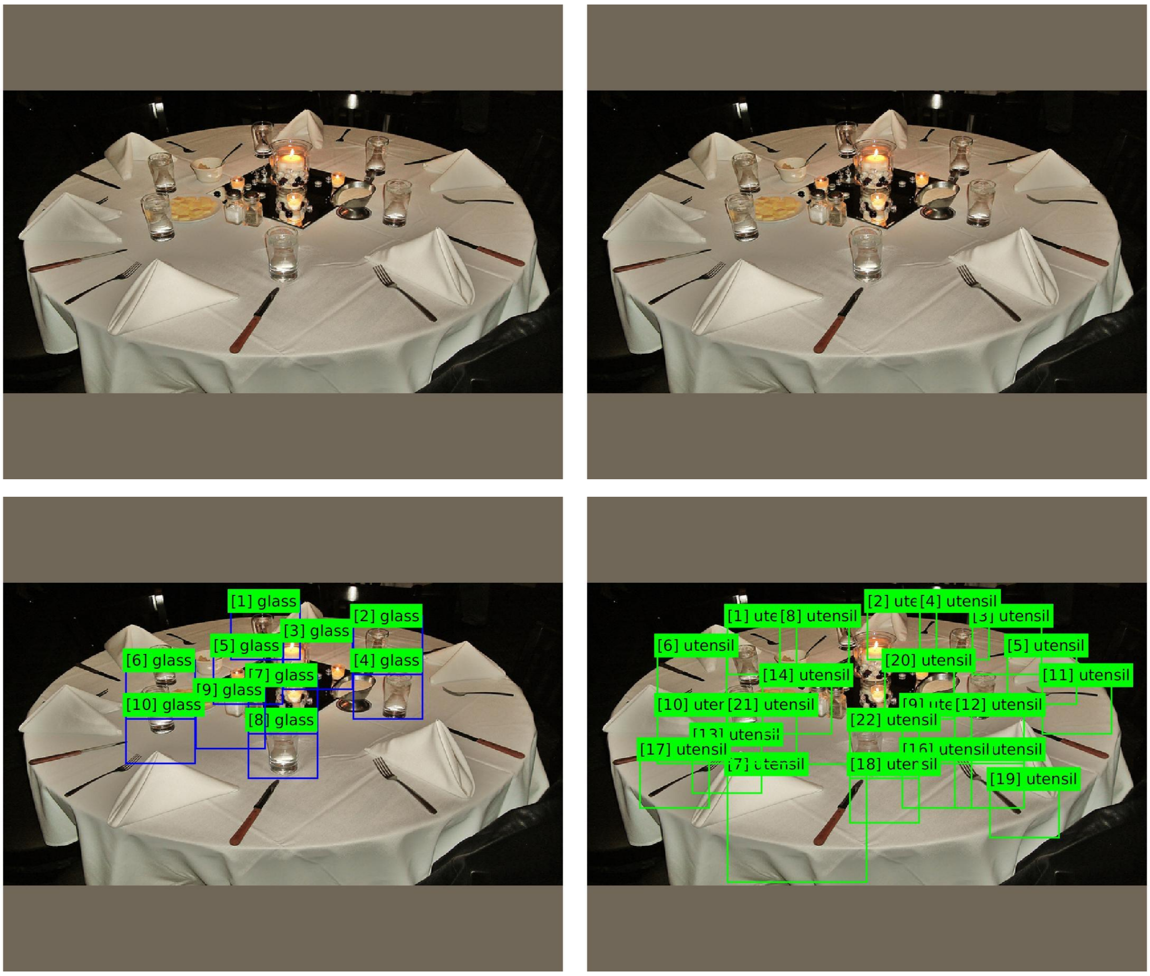


Figure 6.23: Model-based detections (all questions) for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

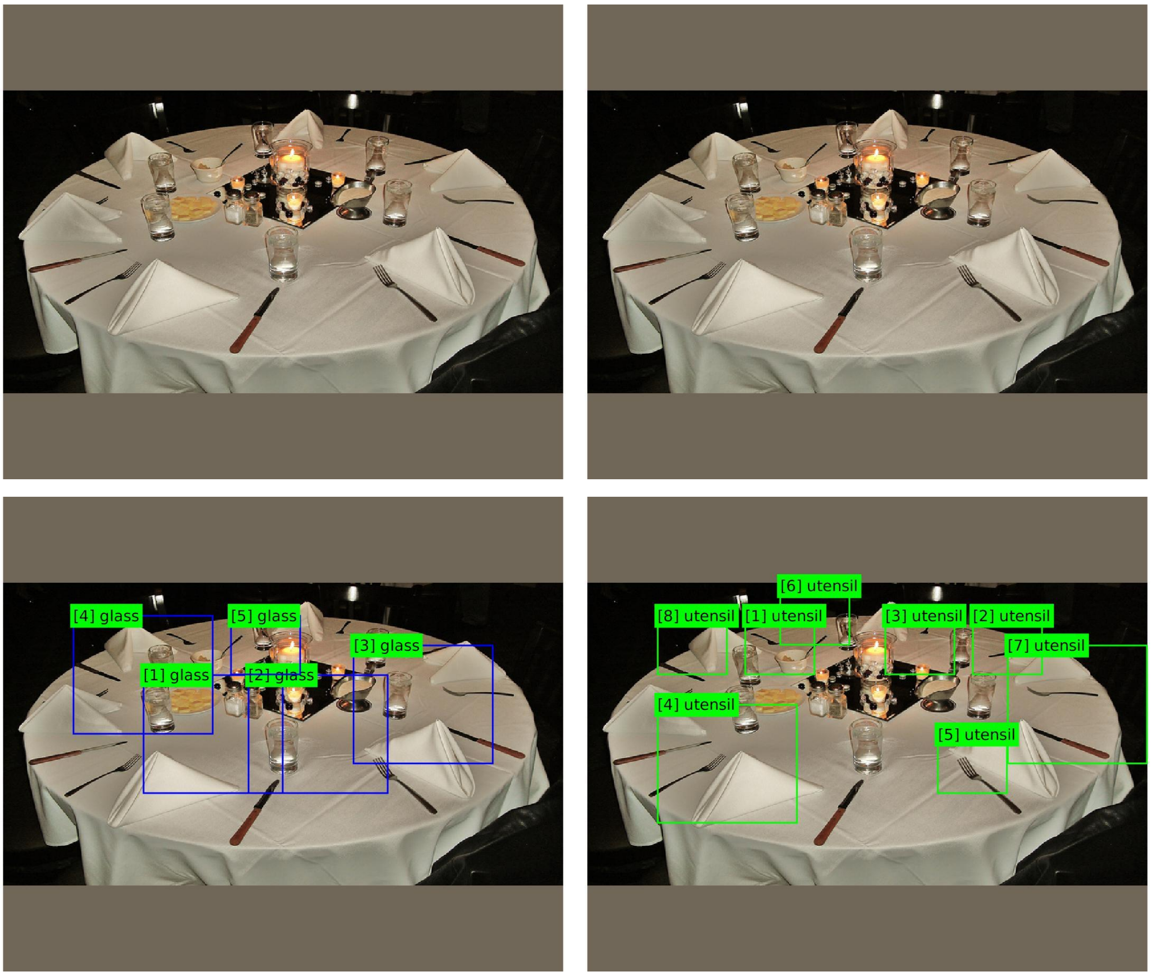


Figure 6.24: Model-based detections (EP 80-Questions) for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

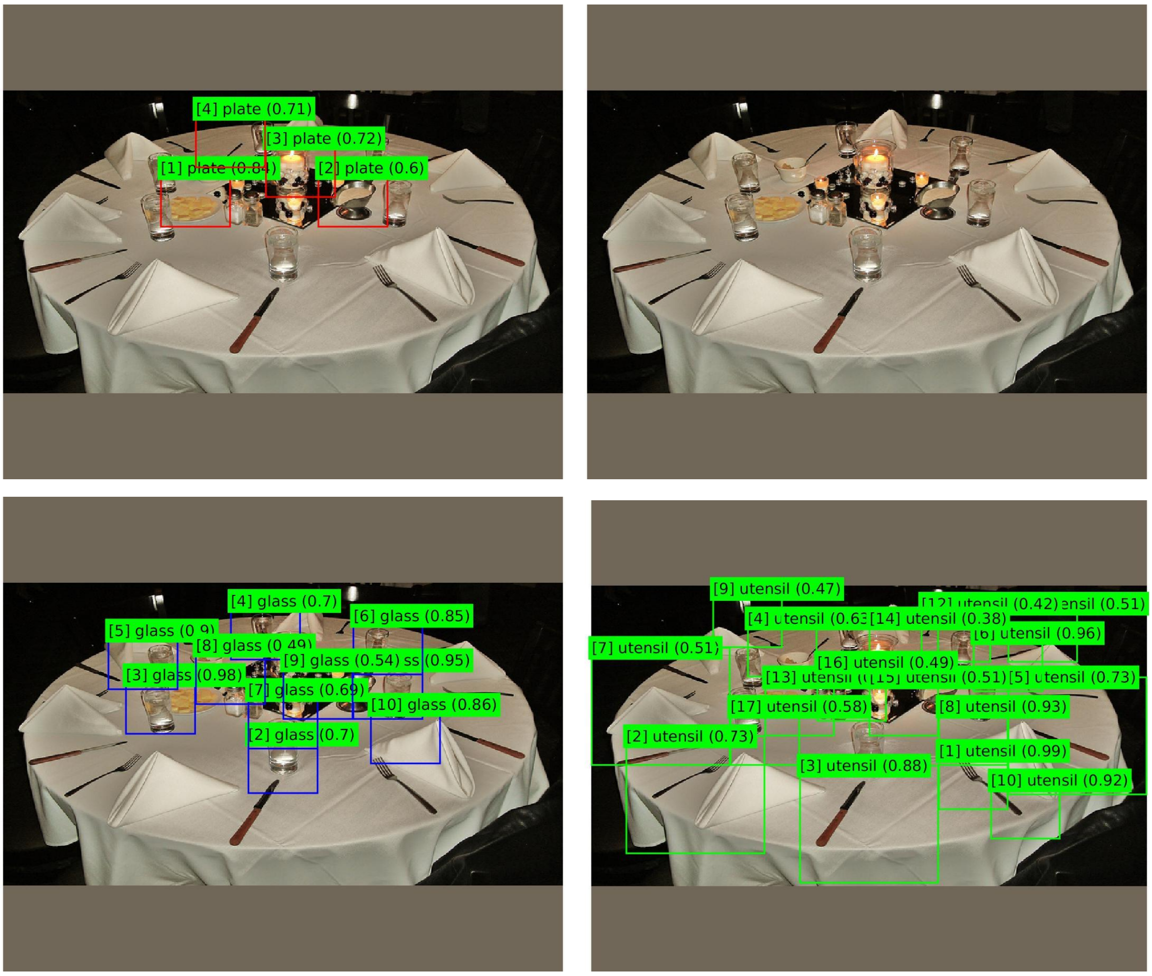


Figure 6.25: CNN detections for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

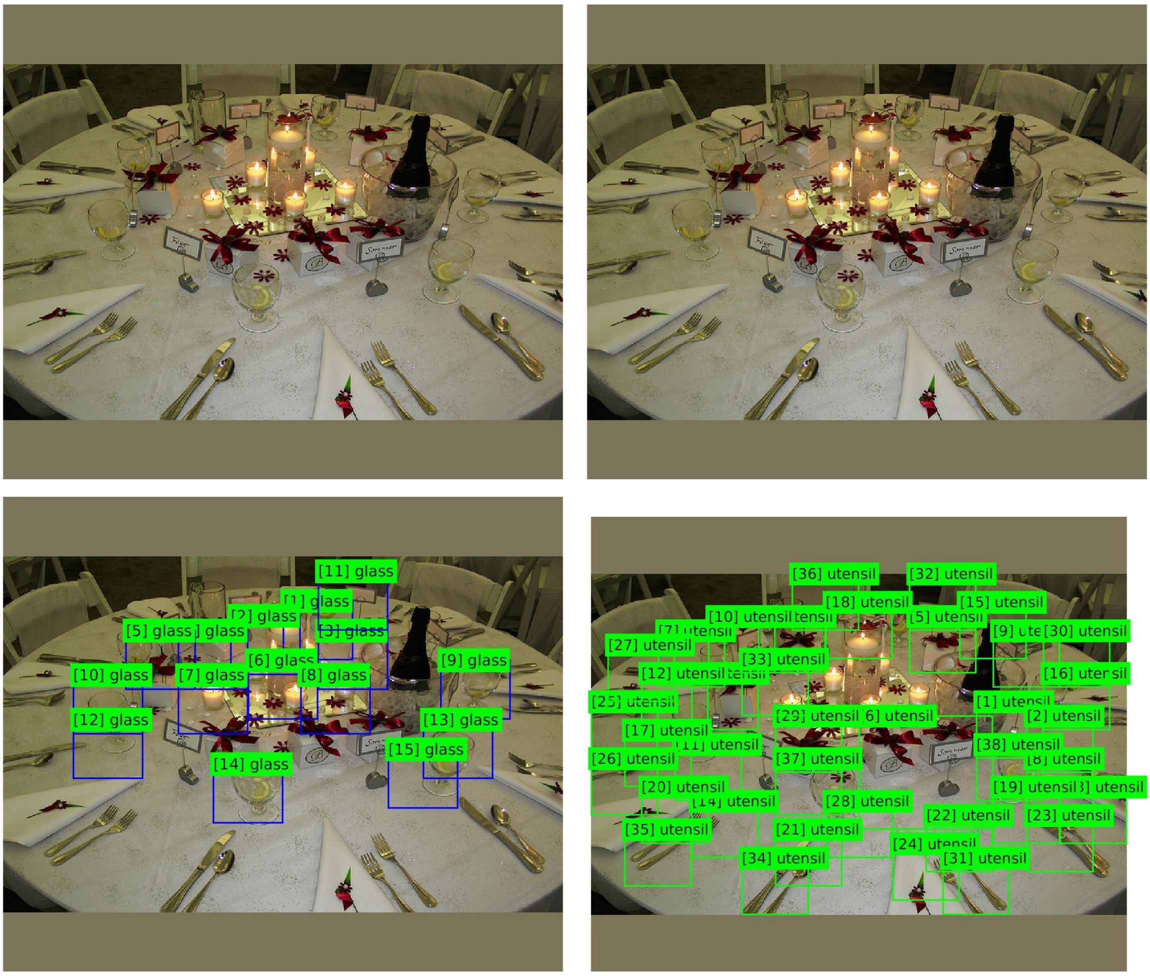


Figure 6.26: Model-based detections (all questions) for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

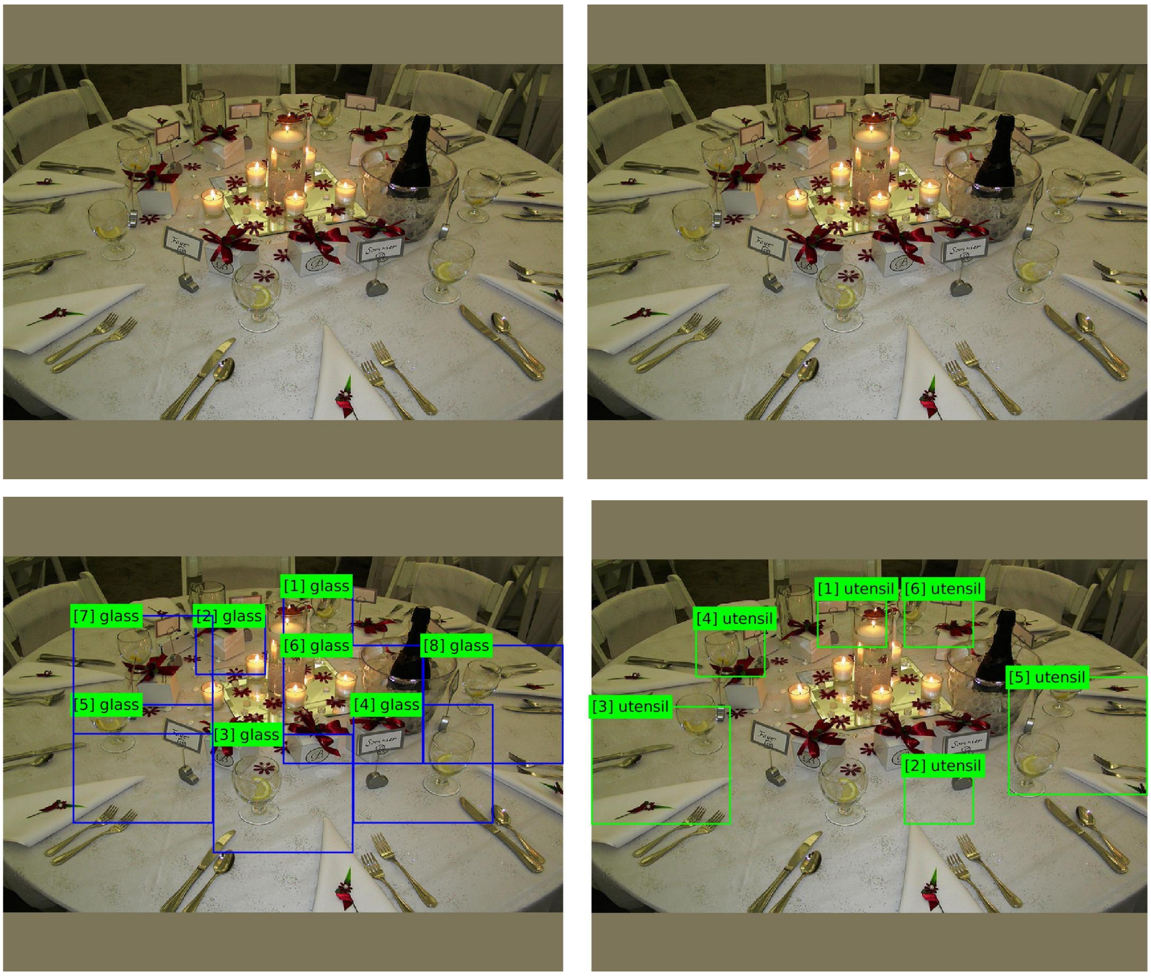


Figure 6.27: Model-based detections (EP 80-Questions) for "plate" (top left), "bottle" (top right), and "glass" (bottom left), and "utensil" (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

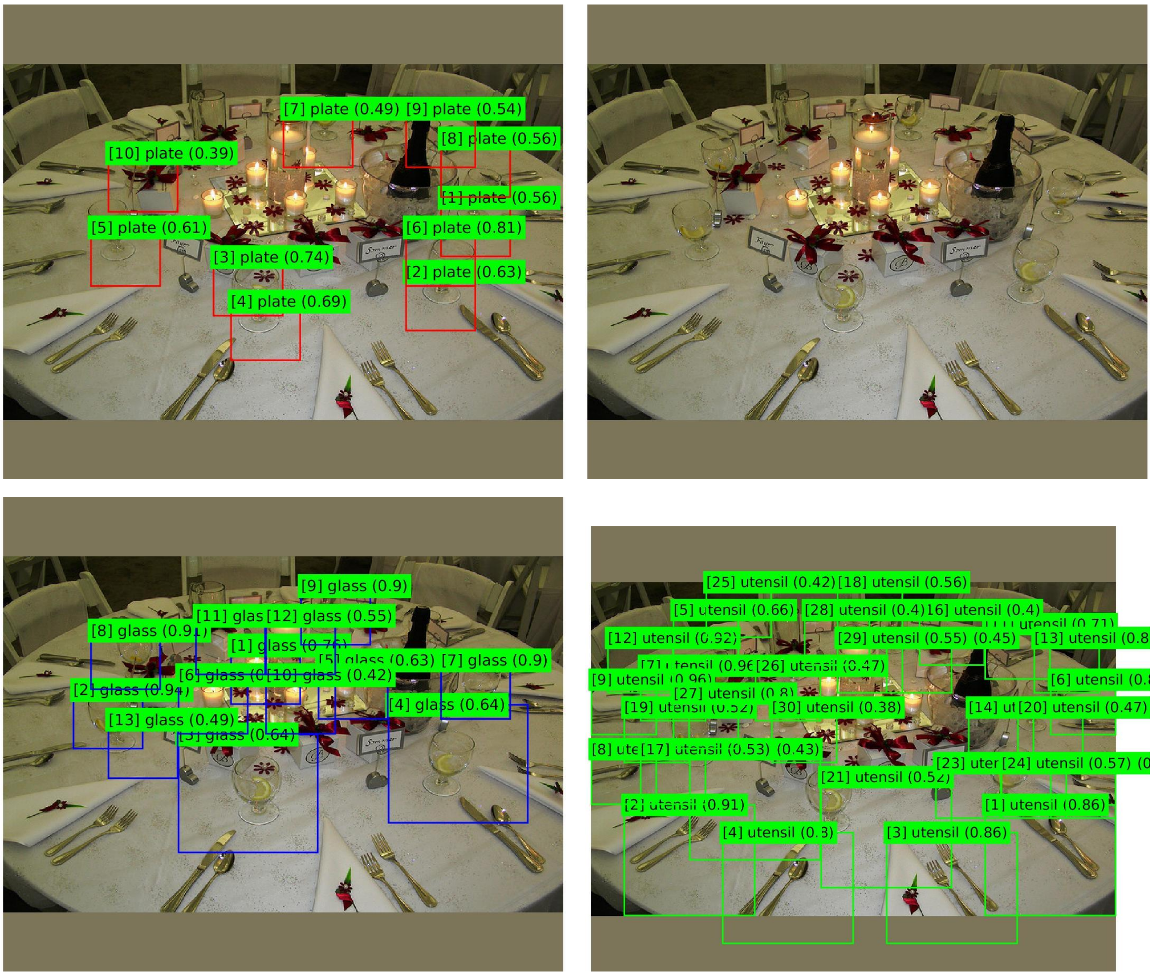


Figure 6.28: CNN detections for “plate” (top left), “bottle” (top right), and “glass” (bottom left), and “utensil” (bottom right) categories. The ordinal numbers in brackets represent the confidence rank of detections per category.

Chapter 7

Conclusion

In the past few years, deep learning has received a flurry of interest in the machine learning community due to its superior performance relative to rival methods. However, deep networks are computationally expensive and, without efficient implementation on high performance computing systems such as GPUs, not as practical as older methods. Therefore, achieving the same or better performance by processing only a small fraction of all patches is desirable, even necessary depending on available resources. This becomes even more notable as networks get deeper requiring systems with higher computational capability. On the other hand, there is a lot of contextual information that, if properly exploited, can distinguish among competing detections by ruling out unlikely scene configurations and encouraging likely ones.

We proposed a new approach for multi-category object recognition, called "Entropy Pursuit" (EP), that sequentially investigates patches from an input test image in order to come up with an accurate description by processing as few patches as possible. Our approach follows the Bayesian framework with a prior model that incorporates the contextual relations between different scene entities such as the spatial and semantic relations among object instances, consistency of scales, constraints imposed by coplanarity of objects, *etc*. As proof of concept we applied the EP approach to table-setting scenes. We designed a novel generative model on attributed graphs with flexible structure where each node in the graph corresponds to an object instance attributed by its category label and 3D pose. The GAG model was not directly used in our EP framework, but the statistics calculated from its samples were used to learn a Markov Random Field (MRF) model employed directly by EP. Whereas, the GAG model could be learned efficiently from the limited number of annotated images, the MRF model offered faster conditional inference. The entropy pursuit search strategy selects patches from the input image sequentially and investigates them to collect evidence about the scene. To investigate each patch we utilized state-of-the-art convolutional neural networks (CNNs). We used a new dataset of about 3000 fully annotated table-setting scenes to learn the Generative Attributed Graph (GAG) model, to train a battery of CNN classifiers, and to test the performance of the EP algorithm.

In summary, we studied the possibility of generating a scene interpretation by investigating only a fraction of all patches from an input image using the entropy pursuit Bayesian approach. The Bayesian framework is the natural approach for integrating contextual relations and the evidence collected using tests. We were able to show that by choosing the right patches in the right order we can identify an accurate interpretation by processing only a fraction of all patches from an input image.

Appendix A

Homography and Camera Parameters

Assuming a pinhole camera model, where the center of projection is the origin of camera coordinate system, we can relate a 3D point (X, Y, Z) in the world coordinate system to its projection point on the image plane in the camera coordinate system (x, y) as below:

$$\eta \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f/s_x & 0 & \dot{x}_0 \\ 0 & f/s_y & \dot{y}_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} R^\top & -R^\top \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{A.1}$$

where, $\eta > 0$ is the distance between the 3D point and the camera plane, $(R, \boldsymbol{t}) \in$ SE(3) [1] is the pose of camera (rotation and translation) in the world coordinate system, and $\mathbf{K} \in \mathbb{R}^{3\times3}$

[1] SE(3) denotes the special Euclidean group representing all rigid motions.

is the camera calibration matrix depending on the focal length f, the size of pixels in the x and y directions denoted respectively by s_x and s_y, and $(\dot{x}_0, \dot{y}_0)$ is the point on the image plane (in pixels) where the camera's principal axis meets the image plane [35]. In case that the image is not cropped we have $\dot{x}_0 = \frac{\text{\#of image columns}}{2}$ and $\dot{y}_0 = \frac{\text{\#of image rows}}{2}$. Note that we assumed a simplified calibration matrix with zero skew *i.e.,* $\mathbf{K}_{12} = \mathbf{K}_{21} = 0$.

Let $R = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_3 \end{bmatrix}$ and $R^\top = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix}$ denote, respectively, the columns and rows of R. Then, the projection of a point on the table, for which $Z = 0$, simplifies to:

$$\eta \mathbf{x} = \mathbf{K} \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & -R^\top \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H\boldsymbol{X}, \tag{A.2}$$

where, H is the *homography* matrix for projection from the world plane to the image plane, $\mathbf{x} = (x, y, 1)^\top$, and $\boldsymbol{X} = (X, Y, 1)^\top$.

Assuming that the camera coordinate system is a counter-clockwise rotation of the table coordinate system by ψ_x degrees about the x-axis[2], ψ_y degrees about the y-axis, and ψ_z degrees about the z-axis, the rotation matrix R can be written as below:

$$R(\psi_x, \psi_y, \psi_z) = R_z(\psi_z) R_y(\psi_y) R_x(\psi_x), \tag{A.3}$$

[2] The counter-clockwise rotation about an axis means the counter-clockwise rotation occurs when the axis points toward the observer.

where, $R_x(\psi_x)$, $R_y(\psi_y)$, and $R_z(\psi_z)$ denote, respectively, the rotation matrices about the x-axis, y-axis, and z-axis defined as below:

$$R_x(\psi_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_x) & -\sin(\psi_x) \\ 0 & \sin(\psi_x) & \cos(\psi_x) \end{bmatrix},$$

$$R_y(\psi_y) = \begin{bmatrix} \cos(\psi_y) & 0 & \sin(\psi_y) \\ 0 & 1 & 0 \\ -\sin(\psi_y) & 0 & \cos(\psi_y) \end{bmatrix},$$

$$R_z(\psi_z) = \begin{bmatrix} \cos(\psi_z) & -\sin(\psi_z) & 0 \\ \sin(\psi_z) & \cos(\psi_z) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

leading to:

$$R(\psi_x, \psi_y, \psi_z) = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_3 \end{bmatrix}, \tag{A.4}$$

where,

$$\boldsymbol{c}_1 = \begin{bmatrix} \cos(\psi_y)\cos(\psi_z) \\ \cos(\psi_y)\sin(\psi_z) \\ -\sin(\psi_y) \end{bmatrix},$$

$$\boldsymbol{c}_2 = \begin{bmatrix} -\cos(\psi_x)\sin(\psi_z) + \sin(\psi_x)\sin(\psi_y)\cos(\psi_z) \\ \cos(\psi_x)\cos(\psi_z) + \sin(\psi_x)\sin(\psi_y)\sin(\psi_z) \\ +\sin(\psi_x)\cos(\psi_y) \end{bmatrix},$$

$$\boldsymbol{c}_3 = \begin{bmatrix} \sin(\psi_x)\sin(\psi_z) + \cos(\psi_x)\sin(\psi_y)\cos(\psi_z) \\ -\sin(\psi_x)\cos(\psi_z) + \cos(\psi_x)\sin(\psi_y)\sin(\psi_z) \\ \cos(\psi_x)\cos(\psi_y) \end{bmatrix}.$$

Alternatively, if $\vec{u}_x$, $\vec{u}_y$, and $\vec{u}_z$ are the unit vectors of the camera coordinate system, and $\vec{u}_X$, $\vec{u}_Y$, and $\vec{u}_Z$ are the unit vectors of the table coordinate system, we have:

$$R = \begin{bmatrix} \vec{u}_x.\vec{u}_X & \vec{u}_y.\vec{u}_X & \vec{u}_z.\vec{u}_X \\ \vec{u}_x.\vec{u}_Y & \vec{u}_y.\vec{u}_Y & \vec{u}_z.\vec{u}_Y \\ \vec{u}_x.\vec{u}_Z & \vec{u}_y.\vec{u}_Z & \vec{u}_z.\vec{u}_Z \end{bmatrix}. \tag{A.5}$$

Appendix B

Perspective Projection of an Ellipse

Under a simplifying assumption that the height of objects is small relative to their distance from camera, we can assume that the objects are planar. Assuming that objects are planar, we can represent an object instance's pose θ by its enclosing 2D ellipse (as illustrated in Figure B.1) whose center, axes length, and orientation of the the main axis specify the location, size and orientation of the object, respectively. An ellipse can be formulated as:

$$AX^2 + BXY + CY^2 + DX + EY + F = 0, \quad B^2 < 4AC.$$

Matrix representation of the above ellipse in the homogeneous coordinate system is:

$$\boldsymbol{X}^\top \theta \boldsymbol{X} = 0,$$

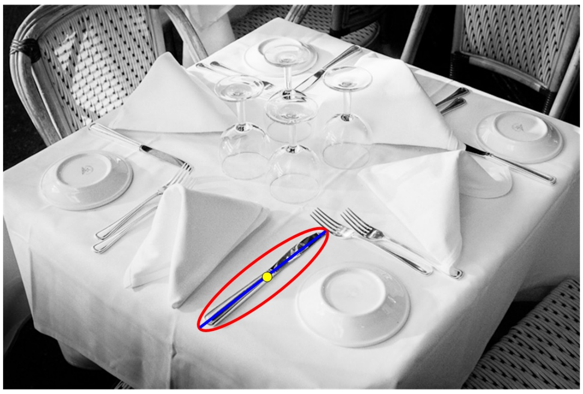


Figure B.1: Knife fitting ellipse.

where:

$$\theta = \begin{bmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \quad \boldsymbol{X}^\top = \begin{bmatrix} X & Y & 1 \end{bmatrix}.$$

For an ellipse centered at $\boldsymbol{x}_0 = (X_0, Y_0)^\top$, main axis orientation of ϕ radians with respect to the x-axis, major and minor radius of respectively a and b, we have:

$$A = cos^2(\phi)/a^2 \ + \ \sin^2(\phi)/b^2,$$

$$B = 2 \times \cos(\phi) \times \sin(\phi) \times (1/a^2 \ - \ 1/b^2),$$

$$C = \sin^2(\phi)/a^2 \ + \ \cos^2(\phi)/b^2,$$

$$D = -2 \times X_0 \times A \ - \ Y_0 \times B,$$

$$E = -2 \times Y_0 \times C \ - \ X_0 \times B,$$

$$F = X_0^2 \times A \ + \ Y_0^2 \times C \ + \ X_0 \times Y_0 \times B \ - \ 1.$$

Using the homography projection equation in (A.2), $\eta \mathbf{x} = H\boldsymbol{X}$, we can write:

$$\mathbf{x}^\top \xi \mathbf{x} = 0,$$

where, H is the *homography* projection matrix from the world plane to the image plane, and $\xi = H_{\eta(\boldsymbol{X})}^{-\top} \theta H_{\eta(\boldsymbol{X})}^{-1}$ for $H_{\eta(\boldsymbol{X})} = \frac{1}{\eta(\boldsymbol{X})} H$. The scale factor $\eta(\boldsymbol{X})$ depends on $\boldsymbol{X}$ because it is the distance of this point from the camera plane. For simplicity by assuming that the object's size is small relative to its distance from the camera we may fix the scale factor for all $\boldsymbol{X}$ to be the distance of the object's center from the camera plane $\eta = \eta(\boldsymbol{x}_0)$. To be more precise, we have:

$$\eta(\boldsymbol{x}_0) = \begin{bmatrix} H(3,1) & H(3,2) & H(3,3) \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ 1 \end{bmatrix},$$

where, $\begin{bmatrix} H(3,1) & H(3,2) & H(3,3) \end{bmatrix}$ represents the third row of the homography matrix H. In this case, the matrix $\xi_0 = H_{\eta(\boldsymbol{x}_0)}^{-\top} \theta H_{\eta(\boldsymbol{x}_0)}^{-1}$ encodes a 2D ellipse in the image coordinate system representing the projected pose. The center, major axis orientation, and the axes length of the projected ellipse which respectively represent the location, orientation and scale of the object's projected pose in the image coordinate system can be extracted from ξ_0 as below:

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = -S_p^{-1} \begin{pmatrix} \xi_0(3,1) \\ \xi_0(3,2) \end{pmatrix},$$

where:

$$S_p = \begin{pmatrix} \xi_0(1,1) & \xi_0(1,2) \\ \xi_0(2,1) & \xi_0(2,2) \end{pmatrix},$$

and $\xi_0(i, j)$ represents the element located at the i^{th} row and j^{th} column of the matrix ξ_0. The axes of the projected ellipse are basically the eigenvectors, $(\vec{u_1}, \vec{u_2})$, of S_p. The major, a_p, and minor, b_p, radii of the projected ellipse can be computed from the eigenvalues, (λ_1, λ_2), of S_p as below:

$$a_p = \max(\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}), \quad b_p = \min(\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}})$$

Note that an alternative approach was to project all of the perimeter points from the enclosing ellipse in the world coordinate system to the image coordinate system using the homography projection H and then find an enclosing ellipse for the projected points. However, the projection of an ellipse calculated in the above way is computationally more efficient if we already have computed an ellipse in the world coordinate system. This is because in addition to not computing the projection of every point we avoid solving an optimization problem that is involved in finding an enclosing ellipse in the image coordinate system.

Appendix C

$\mathcal{A}^+$ and $\mathcal{B}^-$ bounds

For a partitioning of the interval $[a, b]$ into N partitions via partition points $\{u_1, u_2, ..., u_{N+1}\}$ such that $a = u_1 < u_2 < ... < u_{N+1} = b$, we have:

$$\int_a^b f(x)dx = \sum_{n=1}^{N} \int_{u_n}^{u_{n+1}} f(x)dx. \tag{C.1}$$

Hence:

$$\sum_{n=1}^{N} \min_{x \in U_n}\{f(x)\}\ \Delta_n \leq \sum_{n=1}^{N} \int_{u_n}^{u_{n+1}} f(x)dx \leq \sum_{n=1}^{N} \max_{x \in U_n}\{f(x)\}\ \Delta_n, \tag{C.2}$$

where $U_n = [u_n, u_{n+1}]$ and $\Delta_n = u_{n+1} - u_n$. For a monotonically decreasing function $f(x)$ we have:

$$\min_{x \in U_n} f(x) = f(u_{n+1}), \quad \text{and} \quad \max_{x \in U_n} f(x) = f(u_n).$$

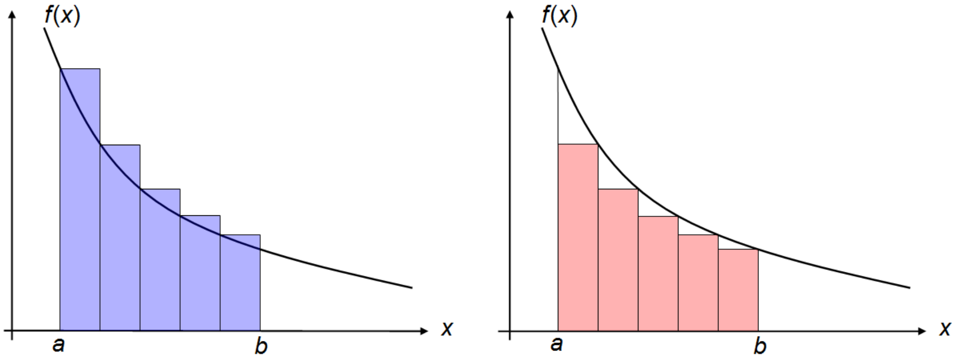


Figure C.1: The total area of rectangles in the left-hand side figure (blue rectangles) is an upper bound on the integral $\int_a^b f(x)dx$ whereas the total area of rectangles in the right-hand side figure (red rectangles) is a lower bound on this integral. These upper and lower bounds converge to $\int_a^b f(x)dx$ when the number of rectangles goes to infinity.

Therefore, for uniform partitioning with $\Delta_n = \Delta x = \frac{b-a}{N}$ we get:

$$\Delta x \sum_{n=1}^{N} f(a + n\Delta x) \leq \int_a^b f(x)dx \leq \Delta x \sum_{n=0}^{N-1} f(a + n\Delta x). \tag{C.3}$$

For example in Fig. C.1, the lower and upper bounds on the integral $\int_a^b f(x)dx$ amounts respectively to the total area of the red and blue rectangles.

Let $a = K\Delta x$ and $f(x) = \frac{1}{x^{2p}}$ which is a decreasing function for $1/2 \leq p \leq 1$. Then, we have $\Delta x = \frac{b}{N+K}$ and according to the left-hand side inequality in (C.3):

$$\begin{aligned}\sum_{n=1}^{N} \frac{1}{(K+n)^{2p}} \leq (\Delta x)^{2p-1} \int_{K\Delta x}^{b} \frac{1}{x^{2p}} dx &= \frac{1}{2p-1}\big(K^{1-2p} - (\frac{b}{\Delta x})^{1-2p}\big) \\ &= \frac{K^{1-2p} - (N+K)^{1-2p}}{2p-1}.\end{aligned}$$

Hence:

$$\sum_{n=K}^{N+K} \frac{1}{n^{2p}} \leq \frac{K^{1-2p} - (N+K)^{1-2p}}{2p-1} + \frac{1}{K^{2p}},$$

and consequently:

$$\mathcal{A}^+ = \frac{K^{1-2p} - Q^{1-2p}}{2p-1} + \frac{1}{K^{2p}}. \tag{C.4}$$

On the other hand, the right-hand side inequality in (C.3) for $a = K\Delta x$ (again leading to $\Delta x = \frac{b}{N+K}$) and $f(x) = \frac{1}{x^p}$ $(1/2 < p \leq 1)$ implies:

$$\sum_{n=0}^{N-1} \frac{1}{(K+n)^p} = \sum_{n=K}^{N+K-1} \frac{1}{n^p} \geq (\Delta x)^{p-1} \int_{K\Delta x}^{b} \frac{1}{x^p} dx = \frac{(N+K)^{1-p} - K^{1-p}}{1-p}.$$

Consequently:

$$\mathcal{B}^- = \frac{(Q+1)^{1-p} - K^{1-p}}{1-p}. \tag{C.5}$$

Note that as Q grows these lower and upper bounds become tighter and eventually $\lim_{Q\to\infty} \mathcal{A}^+ = \mathcal{A}$ and $\lim_{Q\to\infty} \mathcal{B}^- = \mathcal{B}$.

Appendix D

CNN Training

To estimate the model parameters during supervised training a loss function $L(\mathbf{W}, \mathcal{D})$ is minimized over the set of model parameters $\mathbf{W}$ given $|\mathcal{D}|$ training samples from the labeled dataset $\mathcal{D} = \{(I_i, l_i)\}_{i=1}^{N}$ where (I_i, l_i) denotes the i-th training patch and its label. The label l_i is the category of an object instance completely inside patch I_i. In the case that there are multiple object instances inside a patch, the same patch can appeared multiple times in $\mathcal{D}$, each time with the category label of one of the existing object instances. Neural networks with softmax output are often trained under the negative conditional log-likelihood (log loss or cross-entropy) regime where the loss is:

$$\begin{aligned} L(\mathbf{W}, \mathcal{D}) &= \mathbb{E}_{\tilde{p}}\big\{-\log\big(p(J=j|I)\big)\big\} + \lambda R(\mathbf{W}) \\ &= -\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\log\big(p(J=l_i|I_i)\big) + \lambda R(\mathbf{W}), \end{aligned} \tag{D.1}$$

where $R(\mathbf{W})$ is a regularization term with weight λ, $\mathbb{E}_{\tilde{p}}\{.\}$ denotes the expected value over the training dataset $\mathcal{D}$ with empirical distribution $\tilde{p}$, and $p(J = j|I)$ is estimated according to (5.1) where parameters are either seen directly in the equation as weight vectors $\mathbf{w}_j$ or hidden in the last layer's feature vector $\mathbf{x}$ computed based on the previous layers' weights. The number of training samples $|\mathcal{D}|$ is usually very large; therefore, in practice, the loss function in (D.1) is minimized iteratively using a stochastic approximation of the loss by drawing a mini-batch of instances $\mathcal{D}_{\text{mini}}$ with $|\mathcal{D}_{\text{mini}}| \ll |\mathcal{D}|$ in each iteration to compute an unbiased estimate of the gradients. The mini-batch typically contains 30–200 training images. The mini-batch loss is:

$$L_{\text{mini}}(\mathbf{W}, \mathcal{D}_{\text{mini}}) = -\frac{1}{|\mathcal{D}_{\text{mini}}|} \sum_{(I_i, l_i) \in \mathcal{D}_{\text{mini}}} \log\big(p(J = l_i|I_i)\big) + \lambda R(\mathbf{W}). \tag{D.2}$$

The model computes the loss function in the forward pass and the gradients in the backward pass. The gradients are calculated using the backpropagation algorithm [78, 79, 101] that uses the chain-rule to compute partial derivatives of the loss function with respect to the parameters starting from the last layer moving to the first layer. After calculating gradients $\nabla L_{\text{mini}}(\mathbf{W}, \mathcal{D}_{\text{mini}})$ the wights are updated according to the SGD with momentum method [70, 94] where the weights are updated by a linear combination of the negative gradient and the previous weight update value V_t according to:

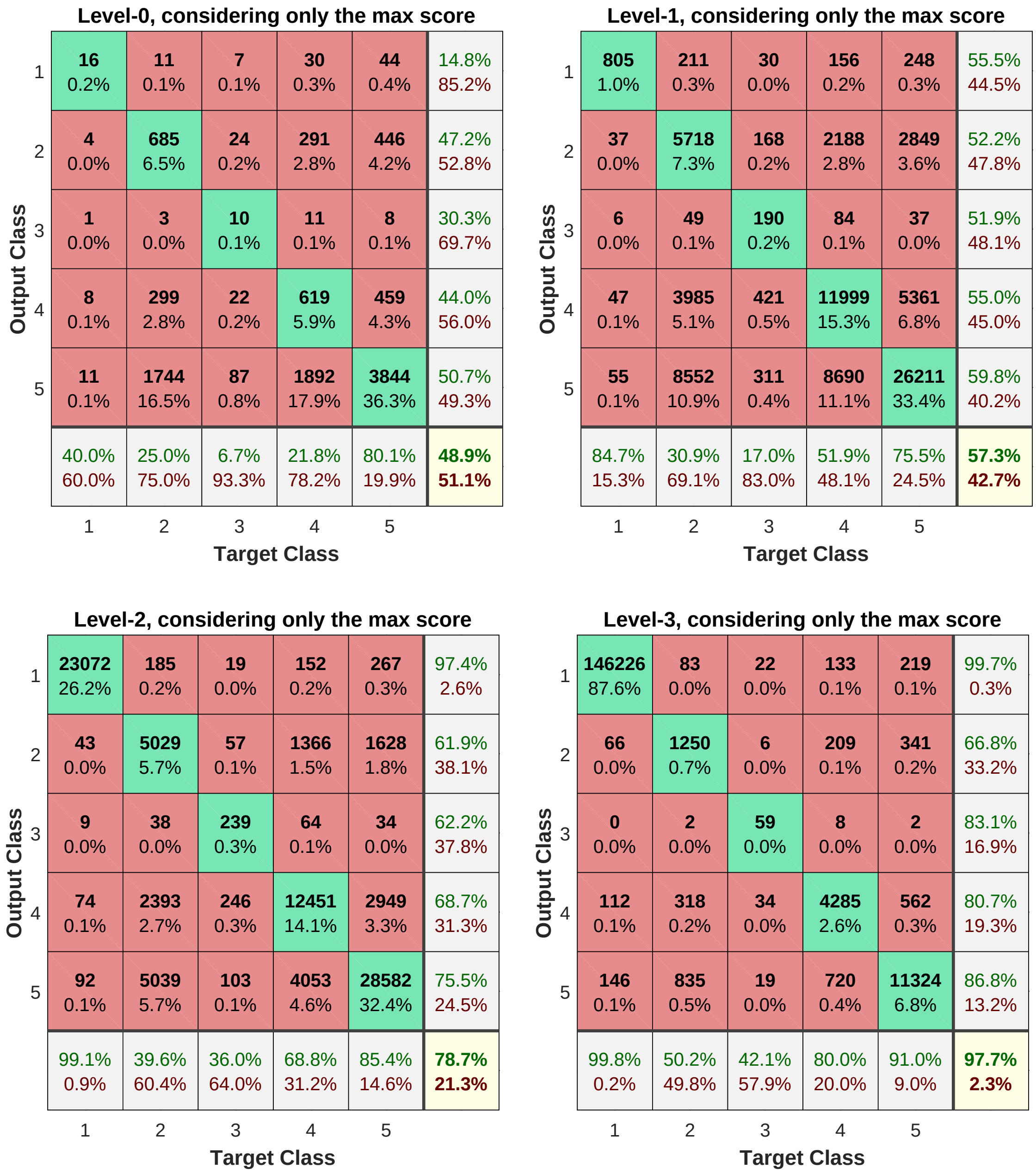
$$
\begin{aligned}
V_{t+1} &= \mu\, V_t - \alpha\, \nabla L_{\text{mini}}(\mathbf{W}_t, \mathcal{D}^t_{\text{mini}}), \\
\mathbf{W}_{t+1} &= \mathbf{W}_{t+1} + V_{t+1},
\end{aligned} \tag{D.3}
$$

where, $V_0 = 0$, and α and μ are two hyperparameters denoting, respectively, the learning rate (weight of the most recent gradient) and momentum (weight of the previous gradients in an exponential discounting function with discount factor μ). The hyperparameters might require some tuning for best results depending on the problem at hand. A “rule of thumb” is to set $\alpha = 0.01$ initially and dropping it by a constant factor (*e.g.,* 10) throughout training when the loss begins to reach a “plateau”, and $\mu = 0.9$ (see [9] for more information).

Appendix E

CatNet Confusion Matrices (Training Set)

Figures E.1, E.2, E.3, E.4 show the confusion matrices of CatNet for different resolution levels of the annocell hierarchy and different score gap values on the training set.


Output Class
Target Class

Level-0, considering only the max score

	1	2	3	4	5	
1	16 0.2%	11 0.1%	7 0.1%	30 0.3%	44 0.4%	14.8% 85.2%
2	4 0.0%	685 6.5%	24 0.2%	291 2.8%	446 4.2%	47.2% 52.8%
3	1 0.0%	3 0.0%	10 0.1%	11 0.1%	8 0.1%	30.3% 69.7%
4	8 0.1%	299 2.8%	22 0.2%	619 5.9%	459 4.3%	44.0% 56.0%
5	11 0.1%	1744 16.5%	87 0.8%	1892 17.9%	3844 36.3%	50.7% 49.3%
	40.0% 60.0%	25.0% 75.0%	6.7% 93.3%	21.8% 78.2%	80.1% 19.9%	**48.9%** **51.1%**

Level-1, considering only the max score

	1	2	3	4	5	
1	805 1.0%	211 0.3%	30 0.0%	156 0.2%	248 0.3%	55.5% 44.5%
2	37 0.0%	5718 7.3%	168 0.2%	2188 2.8%	2849 3.6%	52.2% 47.8%
3	6 0.0%	49 0.1%	190 0.2%	84 0.1%	37 0.0%	51.9% 48.1%
4	47 0.1%	3985 5.1%	421 0.5%	11999 15.3%	5361 6.8%	55.0% 45.0%
5	55 0.1%	8552 10.9%	311 0.4%	8690 11.1%	26211 33.4%	59.8% 40.2%
	84.7% 15.3%	30.9% 69.1%	17.0% 83.0%	51.9% 48.1%	75.5% 24.5%	**57.3%** **42.7%**

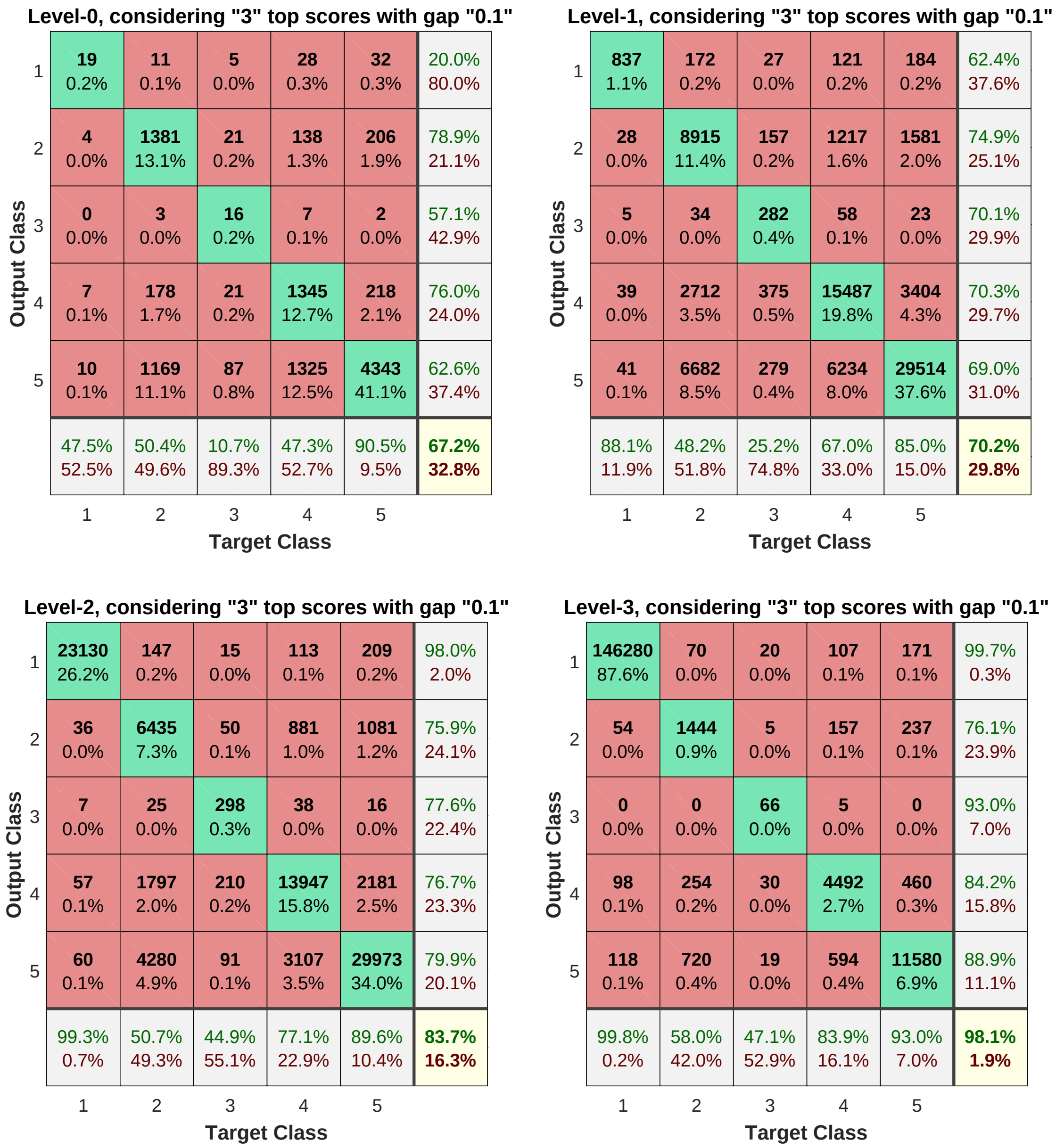
Level-2, considering only the max score

	1	2	3	4	5	
1	23072 26.2%	185 0.2%	19 0.0%	152 0.2%	267 0.3%	97.4% 2.6%
2	43 0.0%	5029 5.7%	57 0.1%	1366 1.5%	1628 1.8%	61.9% 38.1%
3	9 0.0%	38 0.0%	239 0.3%	64 0.1%	34 0.0%	62.2% 37.8%
4	74 0.1%	2393 2.7%	246 0.3%	12451 14.1%	2949 3.3%	68.7% 31.3%
5	92 0.1%	5039 5.7%	103 0.1%	4053 4.6%	28582 32.4%	75.5% 24.5%
	99.1% 0.9%	39.6% 60.4%	36.0% 64.0%	68.8% 31.2%	85.4% 14.6%	**78.7%** **21.3%**

Level-3, considering only the max score

	1	2	3	4	5	
1	146226 87.6%	83 0.0%	22 0.0%	133 0.1%	219 0.1%	99.7% 0.3%
2	66 0.0%	1250 0.7%	6 0.0%	209 0.1%	341 0.2%	66.8% 33.2%
3	0 0.0%	2 0.0%	59 0.0%	8 0.0%	2 0.0%	83.1% 16.9%
4	112 0.1%	318 0.2%	34 0.0%	4285 2.6%	562 0.3%	80.7% 19.3%
5	146 0.1%	835 0.5%	19 0.0%	720 0.4%	11324 6.8%	86.8% 13.2%
	99.8% 0.2%	50.2% 49.8%	42.1% 57.9%	80.0% 20.0%	91.0% 9.0%	**97.7%** **2.3%**

Figure E.1: CatNet confusion matrix on the “training” set broken down for different levels of resolution when classification is only based on the top score.



Level-0, considering "3" top scores with gap "0.1"

Output Class \ Target Class	1	2	3	4	5	
1	19 0.2%	11 0.1%	5 0.0%	28 0.3%	32 0.3%	20.0% 80.0%
2	4 0.0%	1381 13.1%	21 0.2%	138 1.3%	206 1.9%	78.9% 21.1%
3	0 0.0%	3 0.0%	16 0.2%	7 0.1%	2 0.0%	57.1% 42.9%
4	7 0.1%	178 1.7%	21 0.2%	1345 12.7%	218 2.1%	76.0% 24.0%
5	10 0.1%	1169 11.1%	87 0.8%	1325 12.5%	4343 41.1%	62.6% 37.4%
	47.5% 52.5%	50.4% 49.6%	10.7% 89.3%	47.3% 52.7%	90.5% 9.5%	**67.2%** **32.8%**

Level-1, considering "3" top scores with gap "0.1"

Output Class \ Target Class	1	2	3	4	5	
1	837 1.1%	172 0.2%	27 0.0%	121 0.2%	184 0.2%	62.4% 37.6%
2	28 0.0%	8915 11.4%	157 0.2%	1217 1.6%	1581 2.0%	74.9% 25.1%
3	5 0.0%	34 0.0%	282 0.4%	58 0.1%	23 0.0%	70.1% 29.9%
4	39 0.0%	2712 3.5%	375 0.5%	15487 19.8%	3404 4.3%	70.3% 29.7%
5	41 0.1%	6682 8.5%	279 0.4%	6234 8.0%	29514 37.6%	69.0% 31.0%
	88.1% 11.9%	48.2% 51.8%	25.2% 74.8%	67.0% 33.0%	85.0% 15.0%	**70.2%** **29.8%**

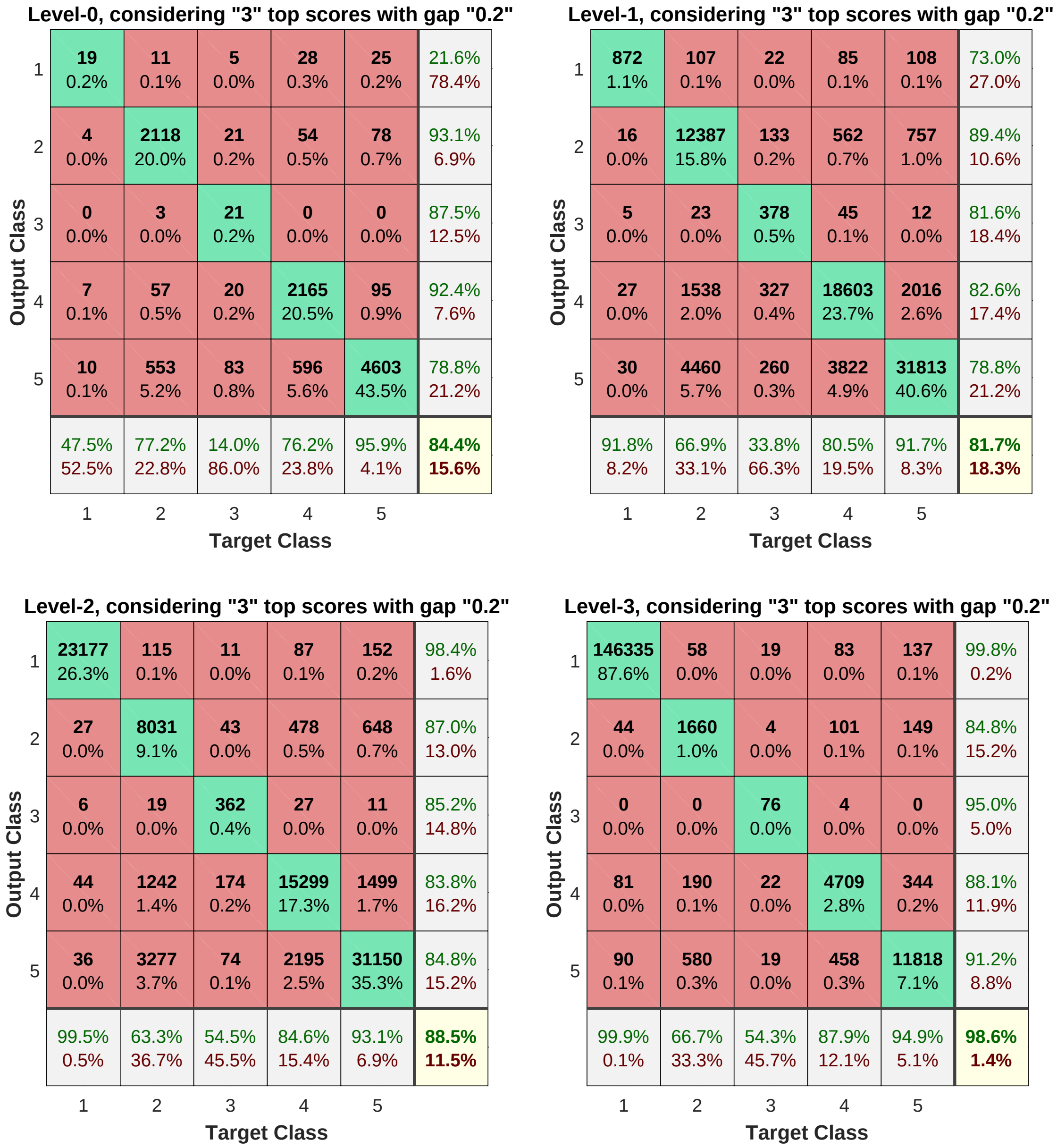
Level-2, considering "3" top scores with gap "0.1"

Output Class \ Target Class	1	2	3	4	5	
1	23130 26.2%	147 0.2%	15 0.0%	113 0.1%	209 0.2%	98.0% 2.0%
2	36 0.0%	6435 7.3%	50 0.1%	881 1.0%	1081 1.2%	75.9% 24.1%
3	7 0.0%	25 0.0%	298 0.3%	38 0.0%	16 0.0%	77.6% 22.4%
4	57 0.1%	1797 2.0%	210 0.2%	13947 15.8%	2181 2.5%	76.7% 23.3%
5	60 0.1%	4280 4.9%	91 0.1%	3107 3.5%	29973 34.0%	79.9% 20.1%
	99.3% 0.7%	50.7% 49.3%	44.9% 55.1%	77.1% 22.9%	89.6% 10.4%	**83.7%** **16.3%**

Level-3, considering "3" top scores with gap "0.1"

Output Class \ Target Class	1	2	3	4	5	
1	146280 87.6%	70 0.0%	20 0.0%	107 0.1%	171 0.1%	99.7% 0.3%
2	54 0.0%	1444 0.9%	5 0.0%	157 0.1%	237 0.1%	76.1% 23.9%
3	0 0.0%	0 0.0%	66 0.0%	5 0.0%	0 0.0%	93.0% 7.0%
4	98 0.1%	254 0.2%	30 0.0%	4492 2.7%	460 0.3%	84.2% 15.8%
5	118 0.1%	720 0.4%	19 0.0%	594 0.4%	11580 6.9%	88.9% 11.1%
	99.8% 0.2%	58.0% 42.0%	47.1% 52.9%	83.9% 16.1%	93.0% 7.0%	**98.1%** **1.9%**

Figure E.2: CatNet confusion matrix on the “training” set broken down for different levels of resolution when classification is based on the top-3 scores and score gap 0.1.



Level-0, considering "3" top scores with gap "0.2"

Output Class \ Target Class	1	2	3	4	5	
1	**19** 0.2%	**11** 0.1%	**5** 0.0%	**28** 0.3%	**25** 0.2%	21.6% 78.4%
2	**4** 0.0%	**2118** 20.0%	**21** 0.2%	**54** 0.5%	**78** 0.7%	93.1% 6.9%
3	**0** 0.0%	**3** 0.0%	**21** 0.2%	**0** 0.0%	**0** 0.0%	87.5% 12.5%
4	**7** 0.1%	**57** 0.5%	**20** 0.2%	**2165** 20.5%	**95** 0.9%	92.4% 7.6%
5	**10** 0.1%	**553** 5.2%	**83** 0.8%	**596** 5.6%	**4603** 43.5%	78.8% 21.2%
	47.5% 52.5%	77.2% 22.8%	14.0% 86.0%	76.2% 23.8%	95.9% 4.1%	**84.4%** **15.6%**

Level-1, considering "3" top scores with gap "0.2"

Output Class \ Target Class	1	2	3	4	5	
1	**872** 1.1%	**107** 0.1%	**22** 0.0%	**85** 0.1%	**108** 0.1%	73.0% 27.0%
2	**16** 0.0%	**12387** 15.8%	**133** 0.2%	**562** 0.7%	**757** 1.0%	89.4% 10.6%
3	**5** 0.0%	**23** 0.0%	**378** 0.5%	**45** 0.1%	**12** 0.0%	81.6% 18.4%
4	**27** 0.0%	**1538** 2.0%	**327** 0.4%	**18603** 23.7%	**2016** 2.6%	82.6% 17.4%
5	**30** 0.0%	**4460** 5.7%	**260** 0.3%	**3822** 4.9%	**31813** 40.6%	78.8% 21.2%
	91.8% 8.2%	66.9% 33.1%	33.8% 66.3%	80.5% 19.5%	91.7% 8.3%	**81.7%** **18.3%**

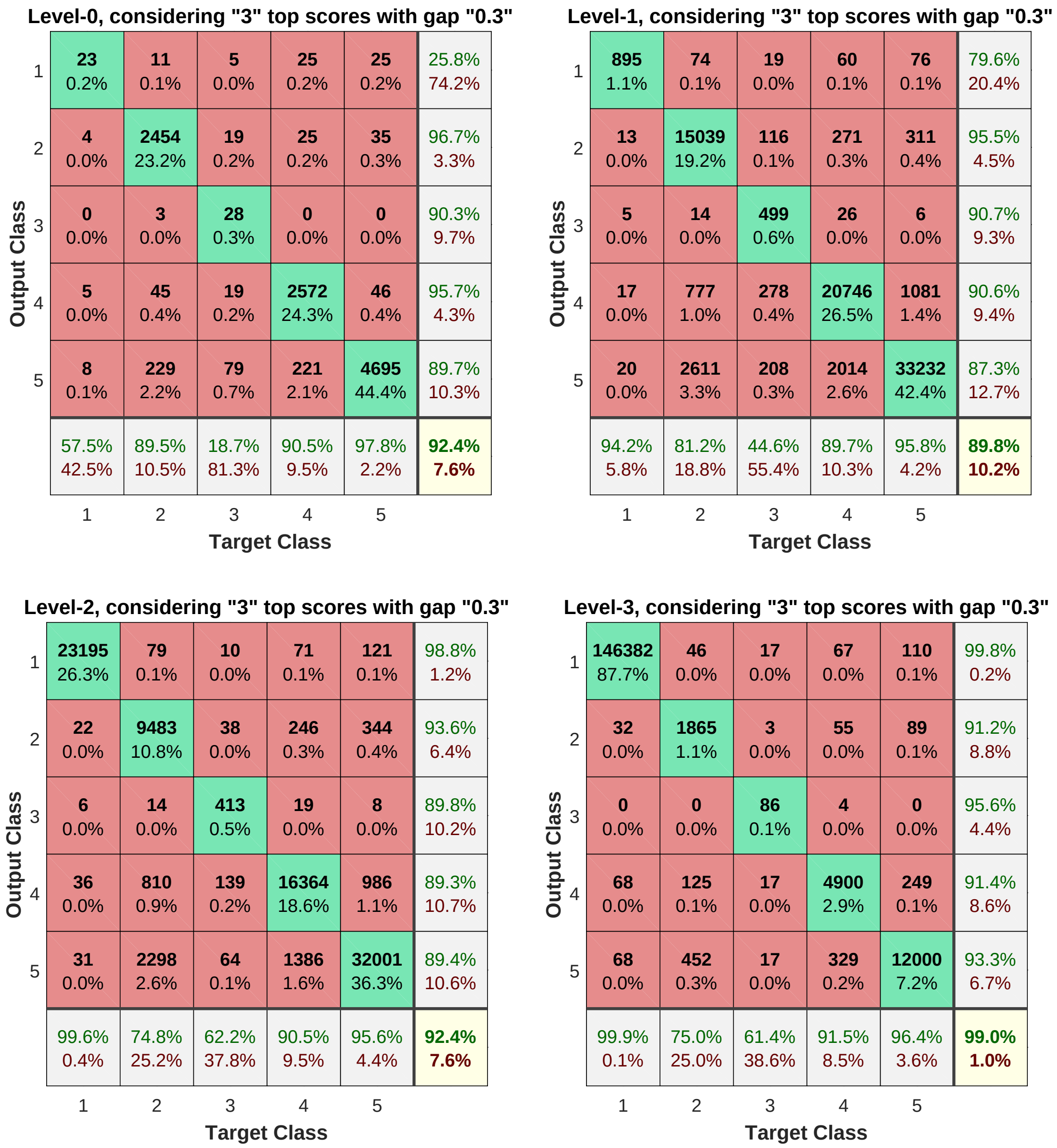
Level-2, considering "3" top scores with gap "0.2"

Output Class \ Target Class	1	2	3	4	5	
1	**23177** 26.3%	**115** 0.1%	**11** 0.0%	**87** 0.1%	**152** 0.2%	98.4% 1.6%
2	**27** 0.0%	**8031** 9.1%	**43** 0.0%	**478** 0.5%	**648** 0.7%	87.0% 13.0%
3	**6** 0.0%	**19** 0.0%	**362** 0.4%	**27** 0.0%	**11** 0.0%	85.2% 14.8%
4	**44** 0.0%	**1242** 1.4%	**174** 0.2%	**15299** 17.3%	**1499** 1.7%	83.8% 16.2%
5	**36** 0.0%	**3277** 3.7%	**74** 0.1%	**2195** 2.5%	**31150** 35.3%	84.8% 15.2%
	99.5% 0.5%	63.3% 36.7%	54.5% 45.5%	84.6% 15.4%	93.1% 6.9%	**88.5%** **11.5%**

Level-3, considering "3" top scores with gap "0.2"

Output Class \ Target Class	1	2	3	4	5	
1	**146335** 87.6%	**58** 0.0%	**19** 0.0%	**83** 0.0%	**137** 0.1%	99.8% 0.2%
2	**44** 0.0%	**1660** 1.0%	**4** 0.0%	**101** 0.1%	**149** 0.1%	84.8% 15.2%
3	**0** 0.0%	**0** 0.0%	**76** 0.0%	**4** 0.0%	**0** 0.0%	95.0% 5.0%
4	**81** 0.0%	**190** 0.1%	**22** 0.0%	**4709** 2.8%	**344** 0.2%	88.1% 11.9%
5	**90** 0.1%	**580** 0.3%	**19** 0.0%	**458** 0.3%	**11818** 7.1%	91.2% 8.8%
	99.9% 0.1%	66.7% 33.3%	54.3% 45.7%	87.9% 12.1%	94.9% 5.1%	**98.6%** **1.4%**

Figure E.3: CatNet confusion matrix on the “training” set broken down for different levels of resolution when classification is based on the top-3 scores and score gap 0.2.



Level-0, considering "3" top scores with gap "0.3"

Output Class						
1	23 0.2%	11 0.1%	5 0.0%	25 0.2%	25 0.2%	25.8% 74.2%
2	4 0.0%	2454 23.2%	19 0.2%	25 0.2%	35 0.3%	96.7% 3.3%
3	0 0.0%	3 0.0%	28 0.3%	0 0.0%	0 0.0%	90.3% 9.7%
4	5 0.0%	45 0.4%	19 0.2%	2572 24.3%	46 0.4%	95.7% 4.3%
5	8 0.1%	229 2.2%	79 0.7%	221 2.1%	4695 44.4%	89.7% 10.3%
	57.5% 42.5%	89.5% 10.5%	18.7% 81.3%	90.5% 9.5%	97.8% 2.2%	**92.4%** **7.6%**
Target Class	1	2	3	4	5	

Level-1, considering "3" top scores with gap "0.3"

Output Class						
1	895 1.1%	74 0.1%	19 0.0%	60 0.1%	76 0.1%	79.6% 20.4%
2	13 0.0%	15039 19.2%	116 0.1%	271 0.3%	311 0.4%	95.5% 4.5%
3	5 0.0%	14 0.0%	499 0.6%	26 0.0%	6 0.0%	90.7% 9.3%
4	17 0.0%	777 1.0%	278 0.4%	20746 26.5%	1081 1.4%	90.6% 9.4%
5	20 0.0%	2611 3.3%	208 0.3%	2014 2.6%	33232 42.4%	87.3% 12.7%
	94.2% 5.8%	81.2% 18.8%	44.6% 55.4%	89.7% 10.3%	95.8% 4.2%	**89.8%** **10.2%**
Target Class	1	2	3	4	5	

Level-2, considering "3" top scores with gap "0.3"

Output Class						
1	23195 26.3%	79 0.1%	10 0.0%	71 0.1%	121 0.1%	98.8% 1.2%
2	22 0.0%	9483 10.8%	38 0.0%	246 0.3%	344 0.4%	93.6% 6.4%
3	6 0.0%	14 0.0%	413 0.5%	19 0.0%	8 0.0%	89.8% 10.2%
4	36 0.0%	810 0.9%	139 0.2%	16364 18.6%	986 1.1%	89.3% 10.7%
5	31 0.0%	2298 2.6%	64 0.1%	1386 1.6%	32001 36.3%	89.4% 10.6%
	99.6% 0.4%	74.8% 25.2%	62.2% 37.8%	90.5% 9.5%	95.6% 4.4%	**92.4%** **7.6%**
Target Class	1	2	3	4	5	

Level-3, considering "3" top scores with gap "0.3"

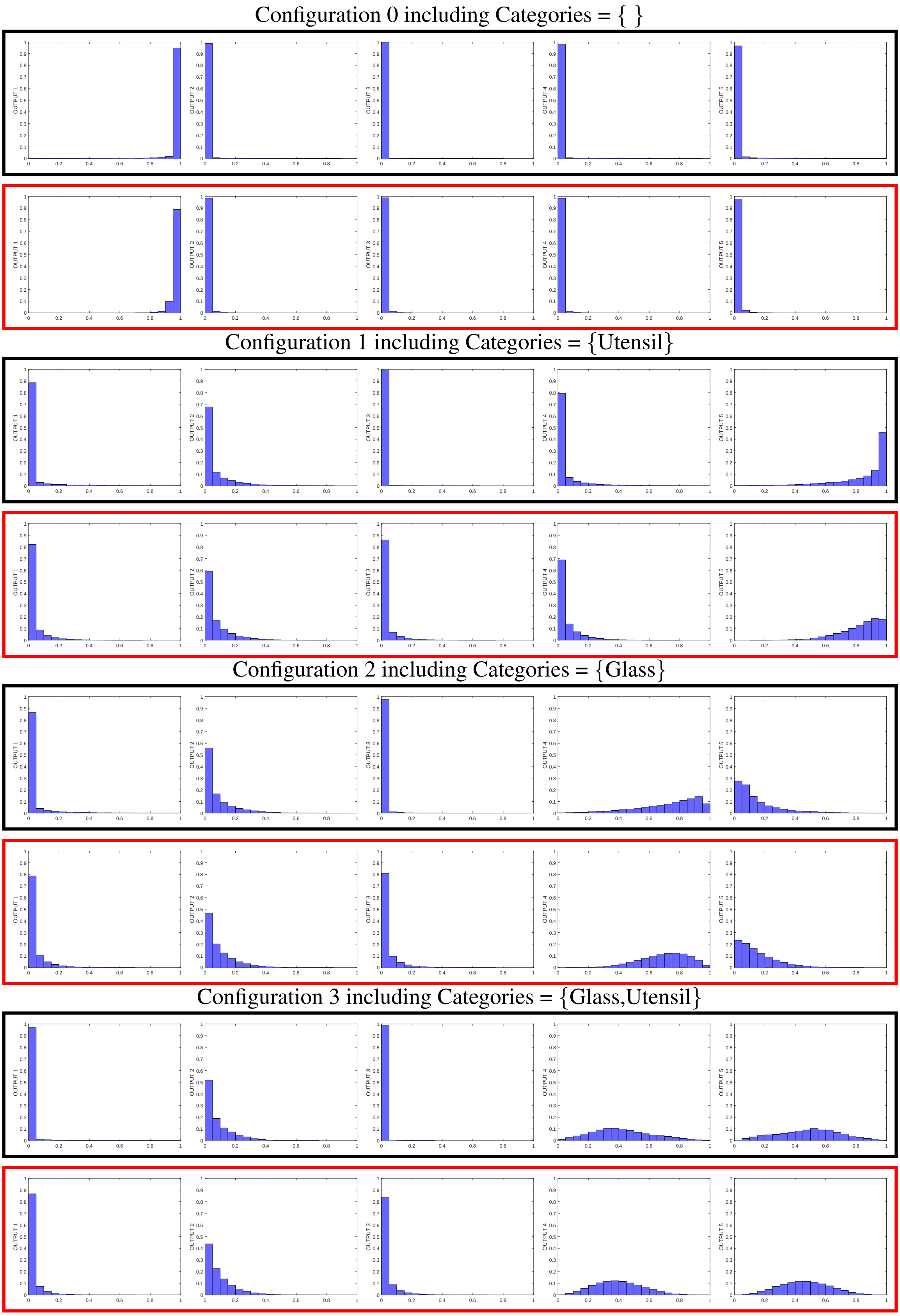
Output Class						
1	146382 87.7%	46 0.0%	17 0.0%	67 0.0%	110 0.1%	99.8% 0.2%
2	32 0.0%	1865 1.1%	3 0.0%	55 0.0%	89 0.1%	91.2% 8.8%
3	0 0.0%	0 0.0%	86 0.1%	4 0.0%	0 0.0%	95.6% 4.4%
4	68 0.0%	125 0.1%	17 0.0%	4900 2.9%	249 0.1%	91.4% 8.6%
5	68 0.0%	452 0.3%	17 0.0%	329 0.2%	12000 7.2%	93.3% 6.7%
	99.9% 0.1%	75.0% 25.0%	61.4% 38.6%	91.5% 8.5%	96.4% 3.6%	**99.0%** **1.0%**
Target Class	1	2	3	4	5	

Figure E.4: CatNet confusion matrix on the “training” set broken down for different levels of resolution when classification is based on the top-3 scores and score gap 0.3.

Appendix F

Dirichlet Data Model Histograms

Figures F.1, F.2, F.3, F.4 show the normalized histogram of the CatNet outputs (framed in black boxes) versus 100,000 samples (framed in red boxes) from the Dirichlet data model for every annoint configuration. Figures F.5, F.6, F.7 show the corresponding normalized histograms for ScaleNet.



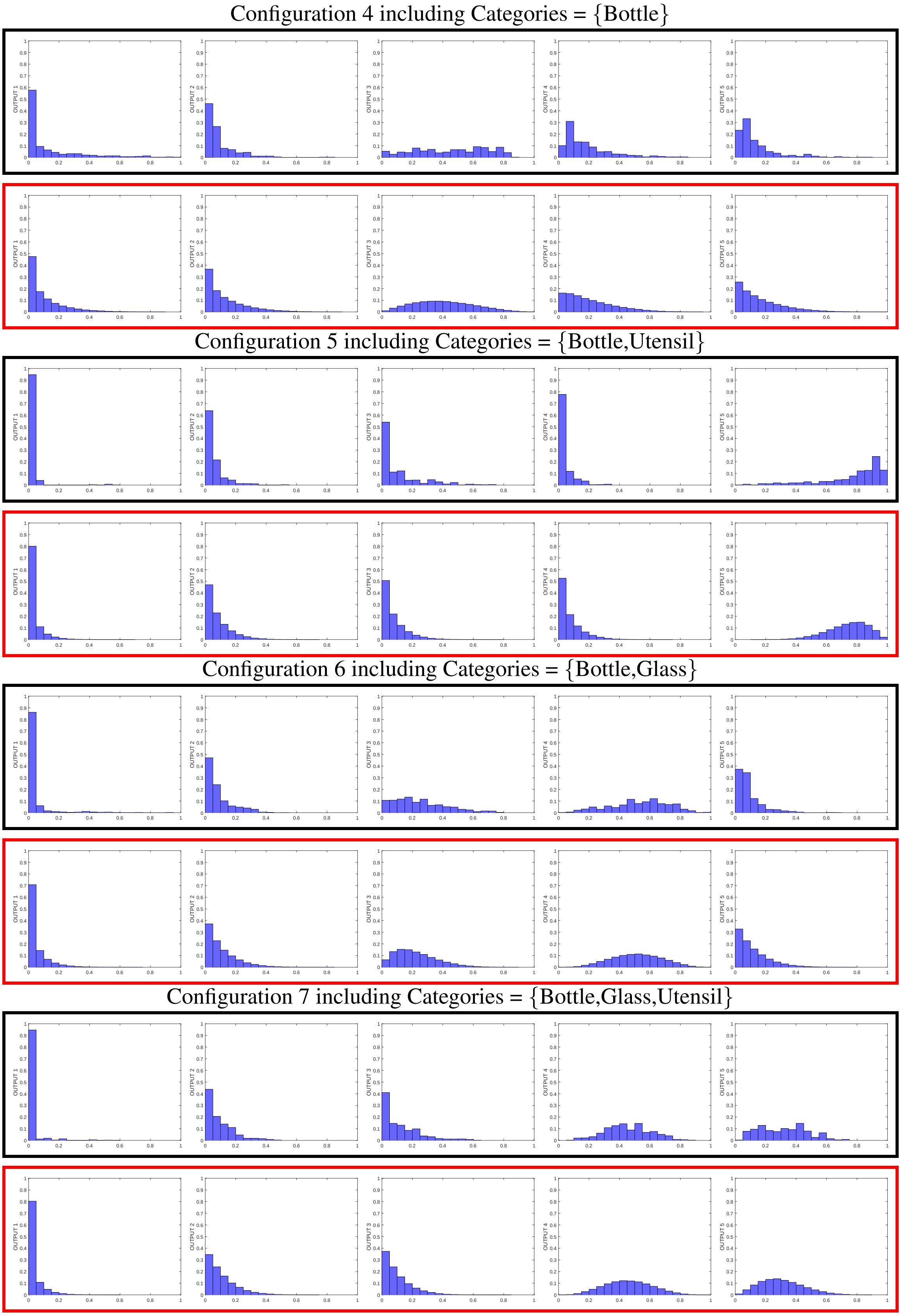
Configuration 0 including Categories = $\{ \}$

Configuration 1 including Categories = $\{$Utensil$\}$

Configuration 2 including Categories = $\{$Glass$\}$

Configuration 3 including Categories = $\{$Glass,Utensil$\}$

Figure F.1: Normalized histograms of CatNet outputs (framed in black boxes) and Dirichlet Dist. (framed in red boxes).



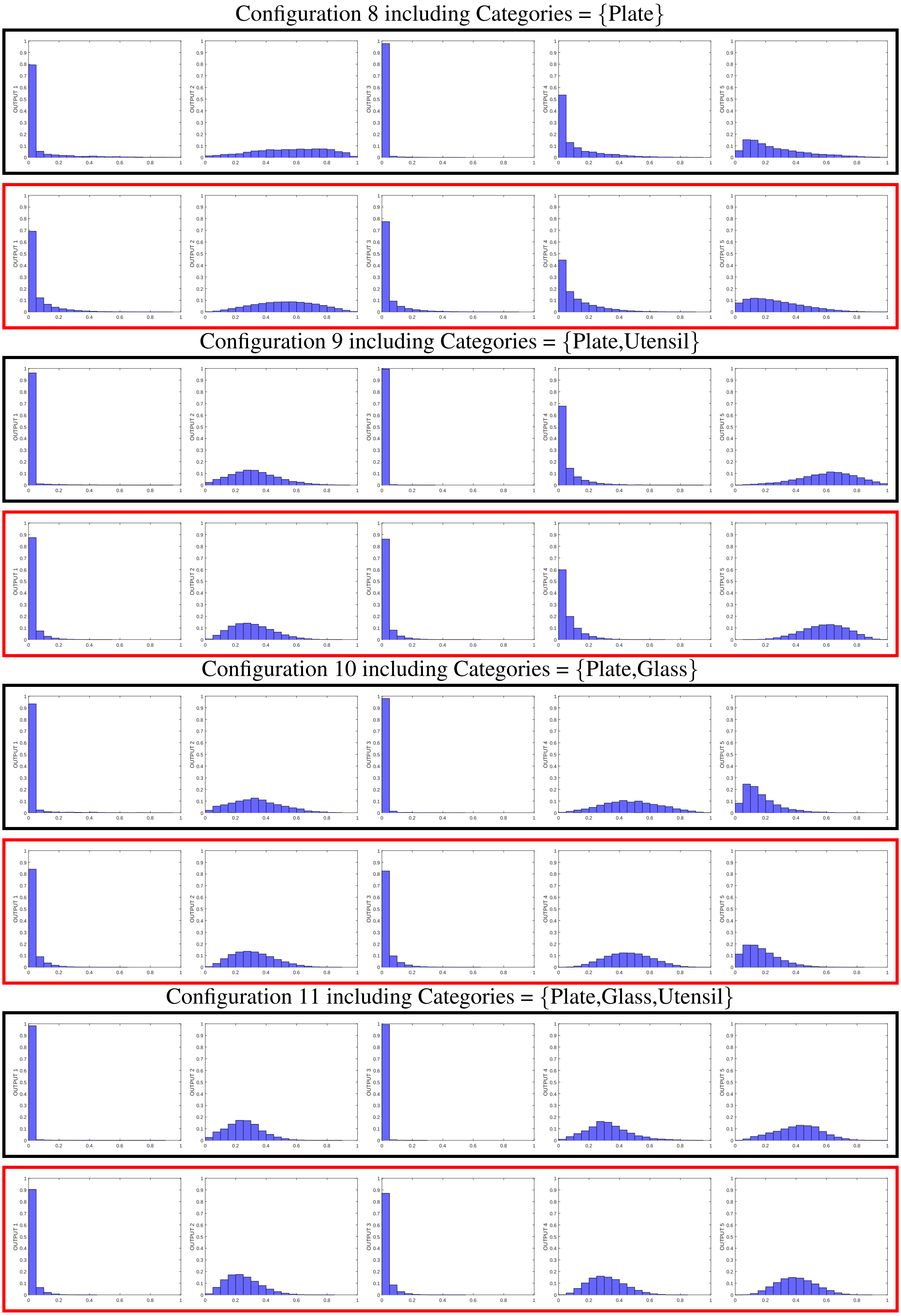
Configuration 4 including Categories = {Bottle}

Configuration 5 including Categories = {Bottle,Utensil}

Configuration 6 including Categories = {Bottle,Glass}

Configuration 7 including Categories = {Bottle,Glass,Utensil}

Figure F.2: Normalized histograms of CatNet outputs (framed in black boxes) and Dirichlet Dist. (framed in red boxes).



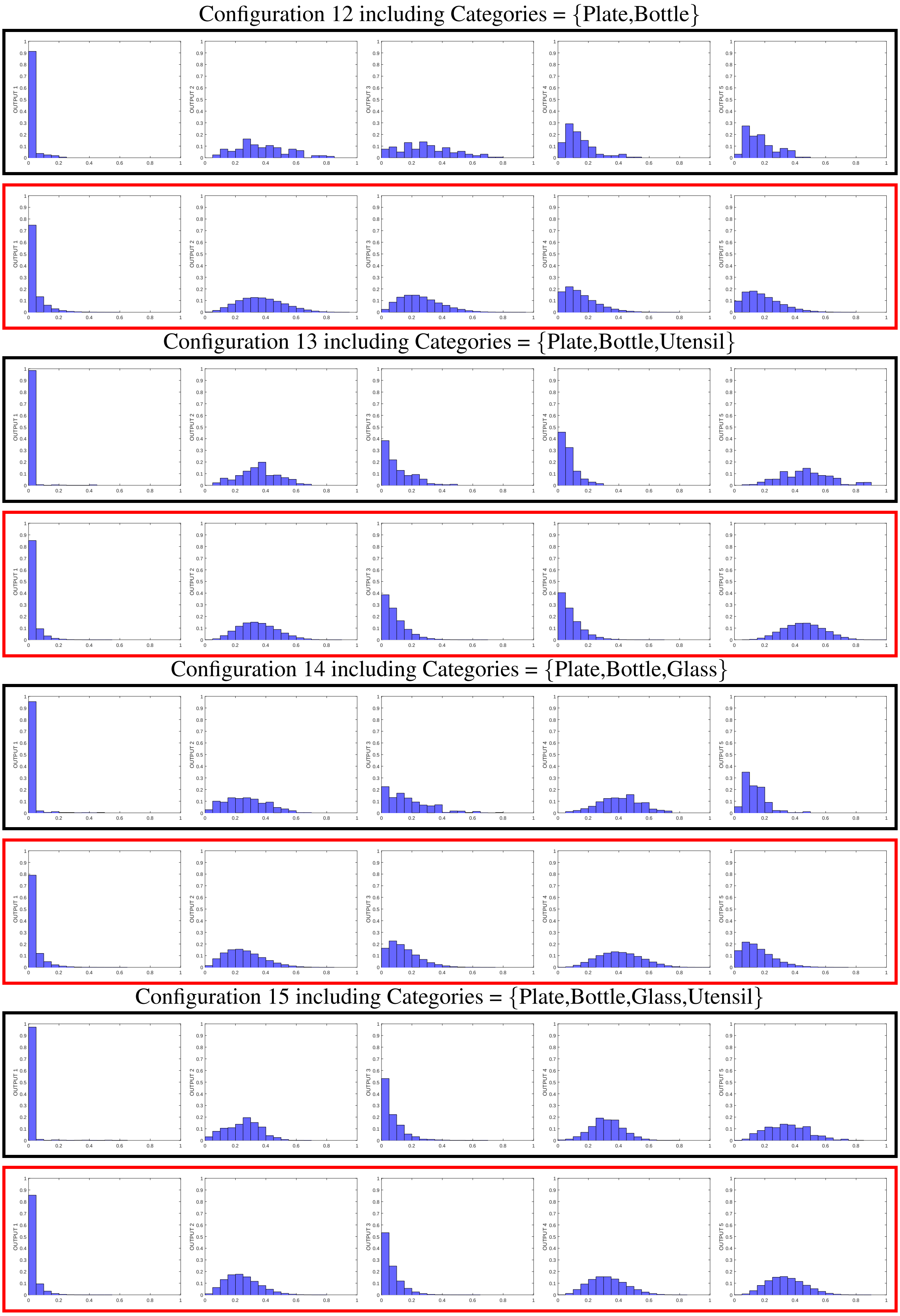
Configuration 8 including Categories = {Plate}

Configuration 9 including Categories = {Plate,Utensil}

Configuration 10 including Categories = {Plate,Glass}

Configuration 11 including Categories = {Plate,Glass,Utensil}

Figure F.3: Normalized histograms of CatNet outputs (framed in black boxes) and Dirichlet Dist. (framed in red boxes).



Configuration 12 including Categories = {Plate,Bottle}

Configuration 13 including Categories = {Plate,Bottle,Utensil}

Configuration 14 including Categories = {Plate,Bottle,Glass}

Configuration 15 including Categories = {Plate,Bottle,Glass,Utensil}

Figure F.4: Normalized histograms of CatNet outputs (framed in black boxes) and Dirichlet Dist. (framed in red boxes).

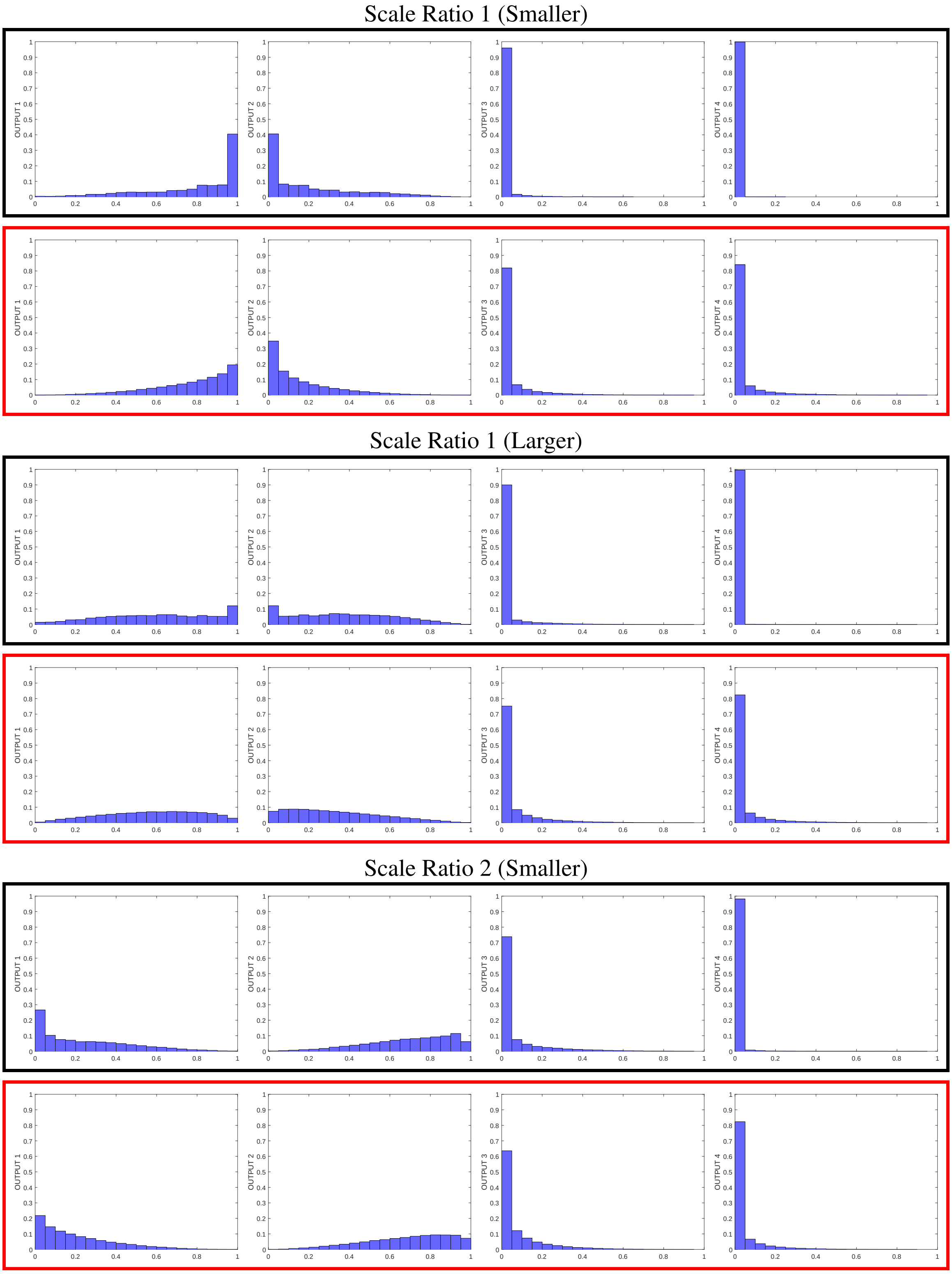
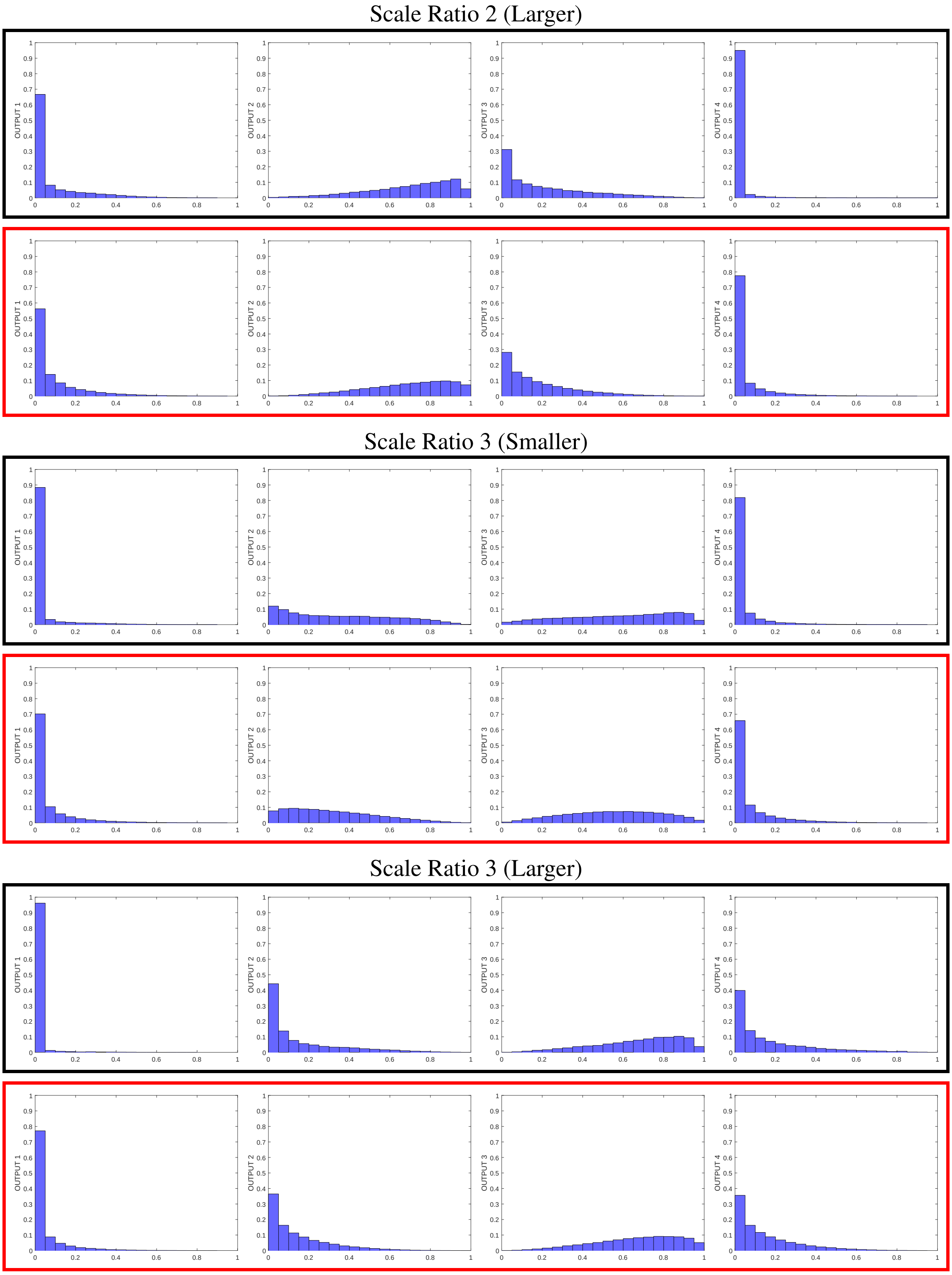


Figure F.5: Normalized histograms of ScaleNet outputs (framed in black boxes) and Dirichlet Dist. (framed in red boxes).

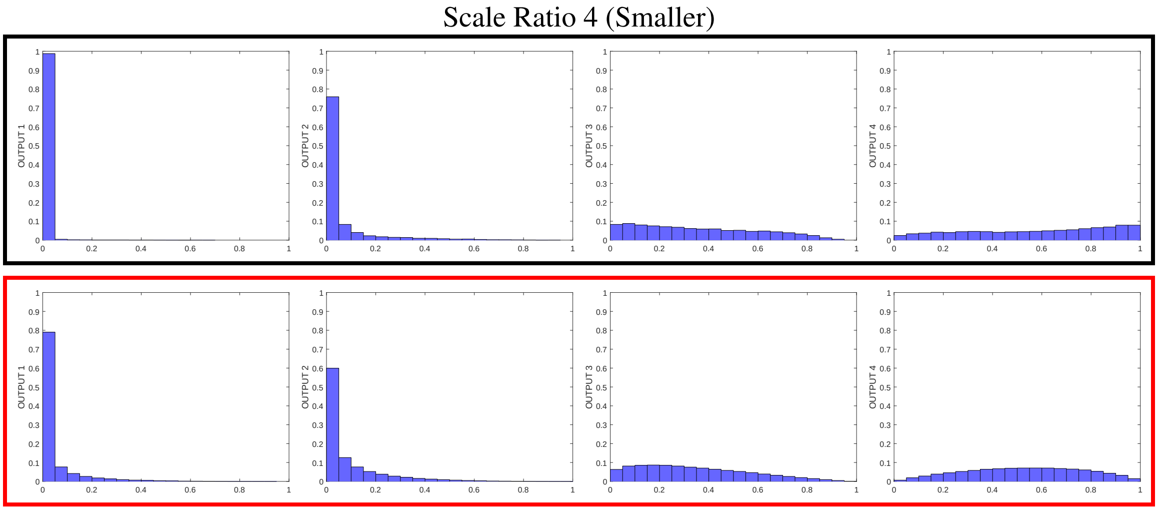


Scale Ratio 2 (Larger)

Scale Ratio 3 (Smaller)

Scale Ratio 3 (Larger)

Figure F.6: Normalized histograms of ScaleNet outputs (framed in black boxes) and Dirichlet Dist. (framed in red boxes).



Scale Ratio 4 (Smaller)

Figure F.7: Normalized histograms of ScaleNet outputs (framed in black boxes) and Dirichlet Dist. (framed in red boxes).

Bibliography

[1] K.B. Athreya and P. E. Ney. *"Branching processes"*. Dover Publications, 2004.

[2] F. Bach and E. Moulines. "Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning". In *NIPS*, pages 451–459, 2011.

[3] M. Bédard. "Optimal acceptance rates for Metropolis algorithms: Moving beyond 0.234". *Stochastic Processes and their Applications*, 118(4):2198–2222.

[4] S. Y. Bao, M. Sun, and S. Savarese. "Toward Coherent Object Detection And Scene Layout Understanding". In *CVPR*, 2010.

[5] Y. Bengio. "Learning Deep Architectures for AI". *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[6] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. "Greedy Layer-Wise Training of Deep Networks". In *Advances in Neural Information Processing Systems*, pages 153–160. MIT Press, 2007.

[7] P. J. Bickel and K. A. Doksum. *“Mathematical Statistics: Basic Ideas and Selected Topics”*. Prentice Hall, 2nd edition, 2000.

[8] I. Biederman. “On the semantics of a glance at a scene”. In *M. Kubovy and J. R. Pomerantz, editors*, Perceptual Organization, chapter 8. Lawrence Erlbaum, 1981.

[9] L. Bottou. “Stochastic Gradient Descent Tricks”, 2012.

[10] L. Bottou and Y. LeCun. “On-line learning for very large data sets”. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

[11] G. Celeux and J. Diebolt. “The SEM Algorithm: A probabilistic teacher algorithm derived from the EM algorithm for the mixture problem”. *Computational Statistics Quarterly*, 2:73–82, 1985.

[12] M. J. Choi, A. Torralba, and A. S. Willsky. “A Tree-Based Context Model for Object Recognition”. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(2):240–252, February 2012.

[13] K. Chua and M. M. Chun. “Implicit scene learning is viewpoint dependent”. *Perception & Psychophysics*, 65(1):72–80, 2003.

[14] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. “Flexible, High Performance Convolutional Neural Networks for Image Classification”. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI’11*, pages 1237–1242. AAAI Press, 2011.

[15] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng. “Large Scale Distributed Deep Networks”. In *Advances in Neural Information Processing Systems*, pages 1232–1240. 2012.

[16] S. Della Pietra, V. Della Pietra, and J. Lafferty. “Inducing Features of Random Fields”. *IEEE Trans. on Patt. Anal. and Mach. Intel.*, 19(4):380–393, April 1997.

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. *J. of the Royal Stat. Soc. Series B (Methodological)*, pages 1–38, 1977.

[18] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. “Large-Scale Object Classification using Label Relation Graphs”. In *ECCV*, 2014.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “ImageNet: A Large-Scale Hierarchical Image Database”. In *CVPR09*, 2009.

[20] C. Desai, D. Ramanan, and C. Fowlkes. “Discriminative models for multi-class object layout”. *Int. J. Comput. Vision*, 2011.

[21] J. Duchi and Y. Singer. “Efficient online and batch learning using forward backward splitting”. *J. Mach. Learn. Res.*, 10:2899–2934, 2009.

[22] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. “Object Detection with Discriminatively Trained Part-Based Models”. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010.

[23] M. A. Fischler and R. A. Elschlager. “The Representation and Matching of Pictorial Structures”. *IEEE Trans. Comput.*, 22(1):67–92, January 1973.

[24] K. Fukushima. “Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position”. *Biological Cybernetics*, 36:193–202, 1980.

[25] C. Galleguillos and S. Belongie. “Context based object categorization: A critical survey”. *Comput. Vis. Image Underst.*, 114(6):712–722, June 2010.

[26] D. Geman, S. Geman, N. Hallonquist, and L. Younes. “A visual Turing test for computer vision systems”. In *Proceedings of the National Academy of Sciences*, 2014.

[27] D. Geman and B. Jedynak. “An active testing model for tracking roads in satellite images”. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(1):1–14, 1996.

[28] S. Geman, E. Bienenstock, and R. Doursat. “Neural Networks and the Bias/Variance Dilemma”. *Neural Comput.*, 4(1):1–58, January 1992.

[29] S. Geman and D. Geman. “Stochastic relaxation, Gibbs distributions, and the

Bayesian restoration of images". *IEEE Trans. Pattern Anal. Machine Intell*, 6:721–741, 1984.

[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 38(1):142–158, Jan 2016.

[31] Ross Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.

[32] C. V. Gottesman. "Viewpoint changes affect priming of spatial layout". *Journal of Vision*, 3(9), 2003.

[33] G. D. Hager and B. Wegbreit. "Scene parsing using a prior world model". *Int. J. of Rob. Res.*, 30(12):1477–1507, 2011.

[34] A. Hanson and E. Riseman. "VISIONS: A computer vision system for interpreting scenes". *Computer Vision Systems.*, pages 303–334, 1978.

[35] R. Hartley and A. Zisserman. *"Multiple View Geometry in Computer Vision"*. Cambridge, 2nd edition, 2004.

[36] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". *Biometrika*, 57(1):97–109, 1970.

[37] E. Hazan, A. Agarwal, and S. Kale. "Logarithmic regret algorithms for online convex optimization". *Machine Learning*, 69(2-3):169–192, 2007.

[38] G. E. Hinton, S. Osindero, and Y-W. Teh. “A Fast Learning Algorithm for Deep Belief Nets”. *Neural Comput.*, 18(7):1527–1554, July 2006.

[39] M. Hoai and A. Zisserman. “Talking Heads: Detecting Humans and Recognizing Their Interactions”. In *CVPR*, 2014.

[40] D. Hoiem, A. A. Efros, and M. Hebert. “Recovering Surface Layout from an Image”. *Int. J. Comput. Vision*, 75(1):151–172, October 2007.

[41] D. Hoiem and S. Savarese. *“Representations and Techniques for 3D Object Recognition and Scene Interpretation”*. Synth. Lec. on Art. Int. and Mach. Learn. Morgan & Claypool Pub., 2011.

[42] T. Homma, L. E. Atlas, and R. J. Marks II. “An Artificial Neural Network for Spatio-Temporal Bipolar Patterns: Application to Phoneme Classification”. In *Neural Information Processing Systems*, pages 31–40. 1988.

[43] D. H. Hubel and T. N. Wiesel. “Receptive Fields, Binocular Interaction, and Functional Architecture in the Cat’s Visual Cortex”. *Journal of Physiology (London)*, 160:106–154, 1962.

[44] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. “Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition”. In *NIPS Deep Learning Workshop*, 2014.

[45] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama,

and T. Darrell. “Caffe: Convolutional Architecture for Fast Feature Embedding”. *arXiv preprint arXiv:1408.5093*, 2014.

[46] Y. Jiang, H. S. Koppula, and A. Saxena. “Hallucinated Humans as the Hidden Context for Labeling 3D Scenes”. In *CVPR*, 2013.

[47] Y. Jin and S. Geman. “Context and Hierarchy in a Probabilistic Image Model”. In *CVPR*, 2006.

[48] R. Johnson and T. Zhang. “Accelerating stochastic gradient descent using predictive variance reduction”. In *Adv. NIPS*, 2013.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In *NIPS*. 2012.

[50] T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. K. Mansinghka. “Picture: A probabilistic programming language for scene perception”. In *CVPR*, pages 4390–4399, 2015.

[51] B. M. Lake and Tenenbaum J. B. Salakhutdinov, R. and. “Human-level concept learning through probabilistic program induction”. *Science*, 350(6266):1332–1338, December 2015.

[52] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition”. *Neural Computation*, 1(4):541–551, Dec 1989.

[53] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[54] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. “Estimating Spatial Layout of Rooms using Volumetric Reasoning about Objects and Surfaces”. In *NIPS*, 2010.

[55] F-F. Li, R. Fergus, and P. Perona. “A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories”. In *International Conference on Computer Vision*, volume 2, page 1134, 2003.

[56] F-F. Li and P. Perona. “A Bayesian Hierarchical Model for Learning Natural Scene Categories”. In *CVPR*, pages 524–531, Washington, DC, USA, 2005.

[57] L-J. Li, R. Socher, and F-F. Li. “Towards total scene understanding: Classification, annotation and segmentation in an automatic framework.”. In *CVPR*, pages 2036–2043, 2009.

[58] X. Liu, Y. Zhao, and SC. Zhu. “Single-View 3D Scene Parsing by Attributed Grammar”. In *CVPR*, 2014.

[59] D. G. Lowe. “Object Recognition from Local Scale-Invariant Features”. In *International Conference on Computer Vision*, volume 2, pages 1150–1157, Washington, DC, USA, 1999.

[60] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. *Int. J. Comput. Vision*, 60(2):91–110, Nov 2004.

[61] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. "Equation of State Calculations by Fast Computing Machines". *Journal of Chemical Physics*, 21:1087–1092, 1953.

[62] T. P. Minka. "The Fastfit Matlab toolbox". `http://research.microsoft.com/en-us/um/people/minka/software/fastfit/`, 2012. [Online; accessed 15-Dec-2015].

[63] C. J. Mode. *"Multitype branching processes; theory and applications"*. American Elsevier Pub. Co New York, 1971.

[64] R. Mottaghi, X. Chen, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. "The Role of Context for Object Detection and Semantic Segmentation in the Wild". In *CVPR*, 2014.

[65] R. Mottaghi, S. Fidler, J. Yao, R. Urtasun, and D. Parikh. "Analyzing Semantic Segmentation Using Hybrid Human-Machine CRFs". In *CVPR*, 2013.

[66] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. "Robust stochastic approximation approach to stochastic programming". *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[67] Y. Nesterov and J. Vial. "Confidence level solutions for stochastic programming". *Automatica*, 44(6):1559–1568, 2008.

[68] T. A. Poggio. “The quest for a theory of vision: from the level framework (revised) to the invariances of the ventral stream”. http://videolectures.net/eccv2012_poggio_stream/, 2012. [Online; accessed 24-Dec-2015].

[69] B. Polyak and A. Juditsky. “Acceleration of Stochastic Approximation by Averaging”. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[70] B.T. Polyak. “Some methods of speeding up the convergence of iteration methods”. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[71] J. Porway, K. Wang, and S. C. Zhu. “A Hierarchical and Contextual Model for Aerial Image Understanding”. *Int'l Journal of Computer Vision*, 88(2):254–283, 2010.

[72] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. “Objects in context”. In *ICCV*, 2007.

[73] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. “Efficient Learning of Sparse Representations with an Energy-Based Model”. In *Advances in Neural Information Processing Systems*, pages 1137–1144. MIT Press, 2007.

[74] J. H. Reynolds, L Chelazzi, and R Desimone. “Competitive mechanisms subserve attention in macaque areas V2 and V4”. *Journal of Neuroscience*, 19:1736–1753, 1999.

[75] H. Robbins and S. Monro. “A Stochastic Approximation Method”. *The Ann. of Math. Stat.*, 22(3):400–407, 1951.

[76] G. O. Roberts and J. S. Rosenthal. “Optimal scaling for various Metropolis-Hastings algorithms”. *Statist. Sci.*, 16(4):351–367, 2001.

[77] G. Ronning. “Maximum-likelihood estimation of dirichlet distributions”. *Journal of Statistical Computation and Simulation*, 32(4):215221, 1989.

[78] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Neurocomputing: Foundations of Research”. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[79] D.E. Rumelhart, G.E. Hintont, and R.J. Williams. “Learning representations by back-propagating errors”. *Nature*, 323(6088):533–536, 1986.

[80] D. Ruppert. “Stochastic Approximation”. *Handbook of Sequential Analysis, B. K. Ghosh and P. K. Sen, eds.*, pages 503–529, 2000.

[81] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[82] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. “LabelMe: A Database and Web-Based Tool for Image Annotation”. *Int. J. Comput. Vision*, 77(1-3):157–173, 2008.

[83] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison.

"SLAM++: Simultaneous Localisation and Mapping at the Level of Objects". In *CVPR*, 2013.

[84] A. Saxena, M. Sun, and A. Y. Ng. "Make3D: Learning 3D Scene Structure from a Single Still Image". *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, May 2009.

[85] J. T. Serences and S. Yantis. "Selective visual attention and perceptual coherence". *Trends in Cognitive Sciences*, 10(1):38–45, 2006.

[86] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In *International Conference on Learning Representations (ICLR 2014)*. CBLS, April 2014.

[87] T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio. "A quantitative theory of immediate visual recognition". *Progress in Brain Research, Computational Neuroscience: Theoretical Insights into Brain Function*, 165:33–56, 2007.

[88] S. Shalev-Shwartz, Y. Singer, and N. Srebro. "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM". In *ICML*, 2007.

[89] S. Shalev-Shwartz and A. Tewari. "Stochastic methods for l_1 regularized loss minimization". In *ICML*, 2009.

[90] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. “Indoor Segmentation and Support Inference from RGBD Images”. In *ECCV*, 2012.

[91] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. *CoRR*, abs/1409.1556, 2014.

[92] J. C. Spall. *“Introduction to Stochastic Search and Optimization”*. John Wiley & Sons, Inc., 1 edition, 2003.

[93] M. Sun, B. Kim, P. Kohli, and S. Savarese. “Relating Things and Stuff via Object-Property Interactions”. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1370–1383, 2014.

[94] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. “On the importance of initialization and momentum in deep learning”. In *Proceedings of International Conference on Machine Learning*, volume 28, pages 1139–1147, May 2013.

[95] R. Sznitman and B. Jedynak. “Active Testing for Face Detection and Localization”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1914–1920, 2010.

[96] R. Sznitman, R. Richa, R. H. Taylor, B. Jedynak, and G. D. Hager. “Unified Detection and Tracking of Instruments during Retinal Microsurgery”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1263–1273, 2013.

[97] M. Tarr and S. Pinker. “Mental rotation and orientation-dependence in shape recognition”. *Cognitive Phycology*, 21(2):233–282, 1989.

[98] A. Torralba, K. P. Murphy, and W. T. Freeman. “Using the forest to see the trees: exploiting context for visual object detection and localization”. *Commun. ACM*, 53(3):107–114, March 2010.

[99] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. “Selective Search for Object Recognition”. *International Journal of Computer Vision*, 2013.

[100] G. C. G. Wei and M. A. Tanner. “A Monte Carlo Implementation of the EM Algorithm and the Poor Man’s Data Augmentation Algorithms”. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

[101] P. J. Werbos. *“Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences”*. PhD thesis, Harvard University, 1974.

[102] L. Xiao. “Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization”. *J. Mach. Learn. Res.*, 11(2543-2596):4, 2010.

[103] M. D. Zeiler and R. Fergus. “Visualizing and Understanding Convolutional Networks”. *CoRR, Published in Proc. ECCV, 2014*, abs/1311.2901, 2013.

[104] S. C. Zhu and D. Mumford. “A stochastic grammar of images”. *Found. Trends. Comput. Graph. Vis.*, 2(4):259–362, January 2006.