Computational Models of Visual Features: From Proto-objects to Object Categories

A Dissertation presented

by

Chen-Ping Yu

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2016

Stony Brook University

The Graduate School

Chen-Ping Yu

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of this dissertation

Dimitris Samaras - Dissertation Advisor
Associate Professor of Computer Science

Gregory Zelinsky - Chairperson of Defense
Professor of Psychology & Computer Science

Minh Hoai Nguyen - Committee Member
Assistant Professor of Computer Science

Talia Konkle - Outside Committee Member
Assistant Professor of Psychology
Harvard University

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Computational Models of Visual Features: From Proto-objects to Object Categories

by

Chen-Ping Yu

Doctor of Philosophy

in

Computer Science

Stony Brook University

2016

The human visual perception is a complex system that excels in tasks such as object recognition and localization, face detection, object segmentation, action classification, and more. While we perform these everyday tasks with ease, little is known about the underlying process of the human visual system. In contrast, computer vision is a field of research that strives for better performance in the aforementioned tasks, which rely on carefully designed statistical machine learning based theories. As the field of computer vision and machine learning has matured over the past decade, there is an abundance of methods and theories that can be utilized for modeling human visual perception, which may shed more light on the underlying processes that make biological visual perception possible. This dissertation discusses novel computational models with the emphasis in visual clutter perception using proto-objects, and categorical search with category consistent features, two important problems in understanding human visual perception.

Visual clutter is a global perception defined as being “crowded disorderly”, affects aspects of our lives ranging from object detection to aesthetics, yet relatively little effort has been made to model this ubiquitous percept. Our approach models clutter as the number of proto-objects segmented from an image, with proto-objects defined as groupings of superpixels that are similar in low-level features. The proto-object model outperforms all other existing models and even a behavioral object segmentation ground truth, which indicates that the number of proto-objects in an image affects clutter perception more than the number of objects or the complexity of features.

In a scope of perception that is more local within a visual field, object category recognition requires one to identify the correct category of a given object from an image. To better understand the human object recognition process, we introduce a generative model of category representation called the category-consistent features (CCFs) from images of category exemplars. The CCF model extracts category representative information from SIFT Bag-of-words (BoW) models and is able to predict human behavior in the context of a categorical search task. Finally, we introduce a ventral-stream inspired deep convolutional neural network (Vs-Net) and a convolutional version of the HMAX model (Deep-HMAX), and analyze these models with the baseline AlexNet under the representational similarity analysis framework (RSA). The results show that the two biologically-inspired models achieve higher object classification accuracies, and the layer-wise representations are more similar between the more biologically-informed models than that of the baseline model.

Table of Contents

List of Figures **vii**

List of Tables **xvi**

Chapter 1
Introduction **1**

Chapter 2
Modeling visual clutter perception **11**

2.1 Background 13
2.1.1 Computational models of clutter 14
2.1.2 Image segmentation and proto-objects 17
2.1.3 The proto-object model of clutter perceiton 19
2.2 Parametric graph partitioning for proto object segmentation (PGP) 20
2.2.1 Superpixel preprocessing and feature similarity comparison . 20
2.2.2 Weibull distribution and EMD for adaptive similarity threshold 22
2.2.3 Weibull mixture model 23
2.2.4 Parameter estimation 24
2.2.5 Visual clutter model with model selection 26
2.2.6 Behavioral data 27
2.2.7 Experiment and results 30
2.2.7.1 Image feature assumptions 30
2.2.7.2 Model evaluation 31
2.2.7.3 Comparison to image segmentation methods 32
2.3 Mean-shift based proto-object segmentation 33
2.3.1 Superpixel clustering with mean-shift 33

2.3.2 Results and discussion 35
2.4 General discussion 39
2.5 Application of PGP to video segmentation 41
2.5.1 Spatio-temporal superpixel graph 43
2.5.2 Feature combination 44
2.5.3 Branch reductions 45
2.5.4 Experiments and results 46
2.5.5 Feature properties 47
2.5.6 Quantitative evaluations 47
2.5.7 Subsampling effect 53

Chapter 3
Modeling object categorical search performance **56**
3.1 Object categories 56
3.2 The generative modeling of visual categories 58
3.3 Hierarchies of categories 58
3.4 Categorical Search 60
3.5 Behavioral methods 62
3.5.1 Participants 62
3.5.2 Stimuli and Apparatus 62
3.5.3 Search procedure 64
3.6 Behavioral results 64
3.7 Model methods 66
3.7.1 Feature extraction 66
3.7.2 Category-consistent features (CCFs) 69
3.8 Model results 71
3.8.1 Test guidance 71
3.8.2 Target verification 74
3.8.2.1 Sibling Distance 74
3.8.2.2 Basic-level Superiority Effect 76
3.8.3 Predicting search behavior using the CCF model 78
3.9 Discussion 79

Chapter 4
Biologically plausible deep learning models **82**
4.1 Deep learning in computer vision 82
4.2 Deep learning in biological vision sciences 86
4.3 Ventral-stream convolutional neural network (VsNet) 87
4.3.1 Convolutional module receptive field sizes 88
4.3.2 Variable RF sizes within an area 90

4.3.3 Bypass connections 91
4.3.4 The V4-like convolutional module 92
4.3.5 Number of convolutional units per module 93
4.4 Deep-HMAX 95
4.5 Model comparison 97
4.5.1 ImageNet training and multi-task fine-tuning 99
4.5.2 Representational similarity analysis 102
4.5.3 Filter visualizations 106
4.6 Discussion and conclusion 108

Appendix A
Maximum Likelyhood Estimation and Non Linear Least Squares110
A.1 MLE using Nelder-Mead algorithm 110
A.2 Nonlinear least squares 111

Bibliography **113**

List of Figures

1.1 (a) shows that the projected subspace (a line) does not separate the blue and the red dots well, but complete separation between the two classes of dots can be achieved with the projected subspace in (b). Image source [3]. 2
1.2 An example illustration of the logistic regression classifier. The blue dots are the 1D training data, the blue curve is the fitted logistic function, and the red line is the decision boundary. Data points that are at the left of the decision boundary are classified as class 1, otherwise class 2. 3
1.3 In the left plot, the blue and the red data points are forming a concentric circle and are not linearly separable; the right plot shows the result of mapping the original data with a polynomial kernel into a high-dimensional space, and the two classes are now linearly separable. Image source [2]. 5
1.4 A simple 3-layer artificial neural network, with 4 input neurons at the input layer, 5 hidden neurons at the hidden layer, and a single output neuron at the output layer. This network can learn to produce real values at the output for regression problems, or learn to produce values close to 0 or 1 for binary classification problems. Image source [8]. 6
1.5 An example process of k-means clustering. Step 1 shows the randomly initialized cluster centers (red and blue thick dots), and as the iteration goes, the algorithm converges at step 6 with no more updates on the cluster centers. Image source [4]. 7
1.6 An example process of Gaussian mixture model. (a) shows the randomly initialized bivariate gaussian distributions (red and blue circles), and as the iteration goes, the algorithm converges at (f) with the model parameters becoming stable. Image source [6]. . . . 8

1.7 An example agglomerative clustering hierarchy. All six letters are in its own cluster initially (top), and they are iteratively merged into a single cluster in the last level. Image source [5]. 9
1.8 An example principle component analysis. PC1 and PC2 are the first and second principle components, and this plot shows that PC1 captures the most variance of the data, as the data forms more of an elliptical shape that is rotated roughly 45°, followed by the second principle component capturing the other part of the data spread. Image source [7]. 10

2.1 How do you quantify set size or the number of "things" in these images? Which scene is more cluttered? 13
2.2 Left: 24 simple objects; Right: 24 compound objects, image source [26]. 15
2.3 From left to right: time lapsed simcity images from rural, suburban, into urban. Image source [107] 16
2.4 Left: input image; Right: proto-object in color blob representation using color blob detector from [49]. Image source [168]. 18
2.5 Left: image of a bathroom. Middle: after superpixel pre-processing [91]. Right: final proto-object segmentation result. 21
2.6 (a) original image, (b) after superpixel pre-processing [10] (977 initial segments), (c) final proto-object partitioning result (150 segments). Each final segment is shown with its mean RGB value to approximate proto-object perception. (d) $\mathcal{W}^2(\mathbf{x}_f;\theta_f)$ optimized using the Nelder-Mead algorithm for intensity, (e) color, and (f) orientation based on the image in (b). The red line indicates the individual Weibull components; and the blue line is the density of the mixture $\mathcal{W}^2(\mathbf{x}_f;\theta_f)$. 25
2.7 Object segmentations from human observers for 4 of the 90 scenes used in this study. Segmentations were provided as part of the SUN09 image collection. To the right of each are lists of the object segment labels (object counts), in matching colors. Top-left: 3 objects. Top-right: 17 objects. Bottom-left: 48 objects. Bottom-right: 57 objects. 29
2.8 Top: Four images from our dataset, rank ordered for clutter perception by human raters, median clutter rank order from left to right: 6, 47, 70, 87. Bottom: Corresponding images after parametric proto-object partitioning, median clutter rank order from left to right: 7, 40, 81, 83. 33

2.9 The computational procedure illustrated for a representative scene. Top row (left to right): a SLIC superpixel segmentation using k = 600 seeds; 51 clusters of median superpixel color using Mean-shift (bandwidth = 4) in HSV color space; 209 proto-objects obtained after merging, normalized visual clutter score = 0.345; a visualization of the proto-object segmentation showing each proto-object filled with the median color from the corresponding pixels in the original image. Bottom row (left to right): an Entropy Rate superpixel segmentation using k = 600 seeds; 47 clusters of median superpixel color using Mean-shift (bandwidth = 4) in HSV color space; 281 proto-objects obtained after merging, normalized visual clutter score = 0.468; a visualization of the proto-object segmentation showing each proto-object filled with the median color from the corresponding pixels in the original image. 35

2.10 Representative examples of 3 of the 90 images used in this study (left column), shown with their corresponding proto-object segmentations (middle column) and reconstructions created by filling each proto-object with its median color (right column). Top row: clutter score = 0.430 (ranked 41st). Middle row: clutter score = 0.540 (ranked 65th). Bottom row: clutter score = 0.692 (ranked 87th). Corresponding rankings from the behavioral participants were: 38th, 73rd, and 89th, respectively. Proto-object model simulations were based on Entropy Rate superpixel segmentations ([91]) using 600 initial seeds and a Mean-shift clustering bandwidth of 4 within an HSV color feature space. 36

2.11 Clutter ranking of the 90 test scenes by the Proto-object model plotted as a function of the median clutter ranking by our 15 behavioral participants for the same 90 scenes. Spearman's $\rho = .814$. . 37

2.12 Spearman's correlations between the mean-shift Proto-object model rankings and behavioral rankings as a function of color space, superpixel segmentation method, and the number of initial superpixel seeds. Leftmost correlations were obtained using Entropy Rate superpixel segmentation; rightmost correlations were obtained using SLIC superpixel segmentation. All correlations used an optimized Mean-shift spatial bandwidth parameter. The highest correlation is indicated in bold. 38

2.13 The overview of video segmentation pipeline using PGP. First, a spatio-temporal superpixel graph is constructed from the video (dashed lines are temporal connections). Then, the edge weights are modeled by a mixture of Weibull distributions, computed separately for individual features. The mixture models generate information about the inter- and intra-cluster edges (intra-cluster edges in bold). The inter-cluster edges are discarded and each of the remaining groups of superpixels becomes a final video segment. . . . 42
2.14 The nonlinear least-squares fits of Weibull Mixture Models (one and two components) on the Earth Mover's Distance statistics of the five features with model selection done for the BMX-1 video. The blue lines are the posterior probability and the red lines are the probability of individual mixture components. The models in the red boxes are the selected ones by AIC. a: intensity, b: hue, c: ab, d: motion, and e: gradient orientation. 44
2.15 Example process of our greedy branch reduction method. Given the nodes (representing superpixels), the red edge is removed first, followed by the green edge. 46
2.16 The plots that show the effects of varying different conditions (initial number of superpixels, size of the sub-volume processing, and with or without motion), for Segtrack V2 dataset: +/-M means with or without motion feature, vol stands for the sub-volume's size as a portion of the original video. These plots suggest that there is no clear indication of the benefit in using the motion feature (dotted lines) within the PGP framework, and that sub-volume processing performs approximately the same as processing the full volume. 48
2.17 The plots show the effects of varying different conditions for Chen's dataset. Sub-volume processing seems to slightly outperform full video mode, and better performance tends to be associated with the exclusion of motion feature. 48
2.18 Example outputs for video BMX from SegTrack v2. From left to right: frame 1, 10, 20, and 30; Top to bottom: original frame, our results of 300 superpixels with motion (NLS+M in Table 1, 300 superixels without motion (NLS-M), and 100 superpixels without motion. Additional results can be found in the supplementary material. 52

2.19 Effect of sub-sampling with respect to 3D segmentation performance. The baseline setting being used here is 400 initial superpixels per frame, 1/4 sub-volume processing, NLS optimization with motion feature (red dotted line). The dotted lines are without any sub-sampling (fs) at 100, 200, 300, and 400 superpixels per frame (x-axis labels in the paranthesis); the solid lines are with sub-sampling at 0.25, 0.5, 0.75 and 1.0 from the edge sets of the baseline. The effects are also analyzed under different sub-volume processing (full (1), 1/4, and 1/8). Note that sub-sampling at 0.25 rate of the original 400 superpixel corresponds to running non sub-sampling on 100 superpixels per frame, therefore making the solid lines and the dotted lines directly comparable. The plots show that sub-sampling barely affects the performance in segtrackV2 dataset, and only observing an effect for the 1/8 sub-volume processing setting in Chen's dataset. 54

3.1 Tiny depictions of the 4,800 images of objects used to train the CCF model - Objects grouped randomly. 59
3.2 Tiny depictions of the 4,800 images of objects used to train the CCF model - Objects grouped into the 4 superordinate-level categories used as stimuli. 60
3.3 Tiny depictions of the 4,800 images of objects used to train the CCF model - Objects grouped into the 48 subordinate-level categories used as stimuli. Note that in Figure 3.2 between-category differences at the superordinate level create clearly separable visual regions, whereas in this figure, some subordinate-level categories are visually distinct from siblings while many others are not. . . . 61
3.4 Procedure for the categorical search task. A target was designated by category name at one of three hierarchical levels, followed after a delay by a six-item target-present/absent search display. The target guidance and verification epochs used for analysis are indicated by the red graphics (not shown to participants) superimposed over the search display. 65
3.5 Behavioral results (dark bars) for the categorical search experiment plotted with the CCF model output (light bars) for (A) time to the first fixation on the target, (B) proportion of immediate fixations on the target, and (C) time from first fixation on the target until the correct target-present button press decision. Error bars indicate 95% confidence intervals. 67

3.6 An overview of the Category-Consistent Features Model. (i) 100 images of object exemplars were collected for 48 subordinate-level categories. These exemplars were combined to create 16 basic-level categories (each with 300 exemplars) and 4 superordinate-level categories (each with 1200 exemplars). (ii) SIFT and color histogram features were extracted from each exemplar and the Bag-of-Words (BoW) method was used to create from these a common feature space consisting of 1064 "visual words". (iii) 1064-bin BoW histograms were obtained for each exemplar, where the bins correspond to the visual words and bin height indicates the frequency of each feature in the exemplar image. BoW histograms were averaged by category to obtain 68 averaged histograms, each now having a mean frequency and variability associated with each visual word. (iv) Features in these averaged histograms having too low of a frequency or too high of a variability were excluded, resulting in a lower-dimensional feature representation of each category that we refer to as category-consistent features (CCFs)those highly informative features that are present both frequently and consistently across the exemplars of a category. 68
3.7 A partial plot (10 of the 1000 SIFT-feature bins) of the histogram obtained by averaging the 100 BoW histograms for the exemplars of the taxi category. Following this averaging each visual word bin has both a mean frequency and a variance (error bars indicate standard error). Blue bars indicate some of the visual words from the averaged histogram that would ultimately become CCFs for the taxi category representation. Gray bars indicate examples of visual words that were excluded from the CCF representation due to either excessive variability or too low of a mean frequency. Note that frequencies are generally low because bin frequency sums to one for representation as a probability distribution. The average bin frequency is therefore 1/1000 or 0.001, considerably smaller than all of the bin frequencies shown. 70

3.8 Histograms of all the exemplars from the taxi category are "stacked" into a rectangular block. Each row shows the entire set of 1064 visual words corresponding to one exemplar, and each column shows the values for a given visual word for each of the 100 exemplars. (A) Averaged histograms of the type plotted in Figure SM2 (blue and gray bins). Brighter pixels denote a higher frequency of features. Note that stacking the histograms makes it possible to visualize feature commonalities across exemplars, with features common to most exemplars appearing as bright vertical bars. Broken vertical bars indicate features appearing inconsistently across exemplars, and dark regions indicate features that are unimportant in the representation of the taxi category. (B) Stacked CCF histograms (only the blue bins from Figure SM2). Vertical bars indicate the CCFs selected from the features in (A), with brightness indicating the mean feature value for a given visual word divided by its standard deviation. Bins 1001-1064 correspond to the hue visual words, where yellows appear prominently across the taxi exemplars. Note that the process of selecting CCFs serves to remove noise from the averaged histograms, thereby accentuating the features most important for the category representation. 72
3.9 Results from the CCF model showing the mean number of category-consistent features (dark bars) and mean sibling distances (light bars) by hierarchical level. Error bars indicate standard error of the mean, computed by treating the number of categories at each level as the number of sample observations (n). 73
3.10 Results from a control experiment testing whether any features from the visual dictionary, regardless of whether they were CCFs, would be as good as the CCF model in characterizing the behavior data. The number of CCF features were found for each of the 68 categories, then the same number of features were randomly selected from the category's averaged BoW histogram (Figure SM2 and SM3A). Sibling distances for each category were then computed based on these non-CCFs and multiplied by the respective CCF number to get a category verification estimate, just as in the CCF model. Averaging these estimates by hierarchical level gives the data in green (leftmost bars), which show a much poorer fit to the behavioral data and CCF model (re-plotted from Figure 3C). This experiment serves as a validation of the CCF approach; only CCFs were able to capture the BSE observed in behavior, the same numbers of randomly selected features could not. 77

3.11 Correlations between the trial-by-trial CCF model predictions and time-to-target (light bars), averaged by level (Fisher z-transform) and plotted with correlations from a corresponding Subject model (dark bars) that captures agreement among participants in their guidance behavior. Data were from all correct trials in which the target was fixated. Error bars indicate one standard error. 79

4.1 Inception module as part of GoogLeNet, image is taken from [150]. 84
4.2 A residual learning block, image is taken from [60]. 85
4.3 The visualization of our ventral-stream network (VsNet). Each convolutional module is colored in blue, with yellow arrows representing dimensionality reductions using 1×1 convolutional filters (similar to GoogLeNet's usage of 1×1 filters within each Inception module); green boxes representing within-module convolutional units and its specification of number of units (red), followed by the RF-size of the units (3×3, 5×5, or 7×7), followed by the stride size, ends with the receptive field size in terms of degrees (5 pixels = 1°); red circles represent concatenation at the depth dimension, and blue arrows represent bypass connections. Every convolutional module is also batch-normalized (BN, [63])and is followed by rectified-linear units (ReLU). Max-pooling is indicated as "mp" if performed at the end of a convolutional module, the specification convention is the same as the green convolutional blocks. VsNet has a total of 62.4 million parameters, comparable to AlexNet's 61 million; a total of 726 convolutional units (excluding the dimensionality reduction 1×1 units) compared to 1152 convolutional units in AlexNet (a more recent version [78]). The architecture of the fully-connected layers is the same as that of AlexNet's fully-connected layers. 89
4.4 The architecture of the convolutional version of an HMAX instantiation [141], which we denote as Deep-HMAX. There are 10 serial and parallel convolutional layers that form a depth of 7 layers. All the simple and complex cells of the original HMAX are replaced with convolutional layers. Following the design principles of the HMAX idea, max-pooling operations are performed between each simple and complex cells. The fully-connected layers contains the same design as the fully-connected layers of AlexNet. 96

4.5 The architecture of the baseline AlexNet model [78]. The only difference from this model to [78] is the addition of batch-normalization at each convolutional layer. We also worked out the RF sizes in both pixels and degrees for each of the convolutional layer for comparison purposes. 98
4.6 An illustration of the three output branches from a pre-trained network, under a multi-task learning setting. 101
4.7 The representational dissimilarity matrices (RDMs) of the AlexNet model, over the hierarchical object category dataset. The rows are the RDMs out of individual layer's responses (from top to bottom: layer 1 to layer 5), and the columns are the RDMs of the different hierarchical levels (from left to right: subordinate, basic, and superordinate). The color indicates the strength of dissimilarity: the warmer (red) the color means the more dissimilar, and the colder (blue) the color means more similar. 103
4.8 The representational dissimilarity matrices (RDMs) of the Deep-HMAX model, over the hierarchical object category dataset. The rows are the RDMs out of individual layer's responses (from top to bottom: v1, v2, v4c, TEOc, and TE2), and the columns are the RDMs of the different hierarchical levels (from left to right: subordinate, basic, and superordinate). The color indicates the strength of dissimilarity: the warmer (red) the color means the more dissimilar, and the colder (blue) the color means more similar. 104
4.9 The representational dissimilarity matrices (RDMs) of the VsNet model, over the hierarchical object category dataset. The rows are the RDMs out of individual layer's responses (from top to bottom: layer 1 to layer 5), and the columns are the RDMs of the different hierarchical levels (from left to right: subordinate, basic, and superordinate). The color indicates the strength of dissimilarity: the warmer (red) the color means the more dissimilar, and the colder (blue) the color means more similar. 105
4.10 Feature visualization from the first convolutional layer of AlexNet, Deep-HMAX, and VsNet. Note that the features of AlexNet were resized from the original size of 11×11; Deep-HMAX features were resized from the original size of 7×7; the original sizes of the VsNet features were 3×3, 5×5, and 7×7. They were enlarged for better visualization purposes. 107

List of Tables

2.1 Correlations between human clutter perception and all the evaluated methods. WMM is the Weibull mixture model underlying our proto-object partitioning approach (-m means with MLE, -n means with NLS), with both optimization methods. 32
2.2 Quantitative evaluation on the SegTrack v2 dataset, [174]a is UES with SWA, and [174]b is UES with GBH. For the 3D UE metric, the lower the error the better. For all the other metrics, the higher the score, the better. Best values are shown in **bold**. For the variants of our methods, the letters stand for the optimization method used, and +/- indicates the use of motion feature. All of the reported variants were based on 300 initial superpixels per frame at a 1/4 sub-volume processing mode. This table shows that our method's averages outperformed [174] in all 4 metrics, while our medians came out on top in 3. The n/a entry indicates that the method failed to converge to a result, the cases here were due to memory overload. 49
2.3 Quantitative evaluation on the Chen Xiph.org dataset. All of the settings are exactly the same as Table 1. For this dataset, both our average and median values outperformed [174] in 3 out of 4 metrics. 50
2.4 Run-time analysis on the effect of sub-sampling on a 85-frame video (Chen's dataset), the base-line (full) setting is using 400 initial superpixels per frame, 1/4 sub-volume processing, NLS optimization with motion feature. The sub-sampling allows PGP to further reduce run-time by over 50%, while maintaining the segmentation performance of the base-line setting (see Figure 1). The experiment is done on a Xeon X3470 at 2.93 Ghz with 32 Gb. The maximum memory usage was recorded at 4 Gb. 55

3.1 Object categories grouped by hierarchical level. 63

4.1 A summary of the RF-size estimates from select literatures, in terms of degree visual angle. Numbers in bold are human estimates, otherwise the estimates were taken from macaques. The * indicates that the RF-size estimates were taken at 5.5° eccentricity, while the others include a range of eccentricities, or pooled from different literatures of different of eccentricities. 90
4.2 Validation set performance for the three models at the end of the 60th training epoch. All results were obtained using the center 224×224 crops. While a deeper model such as Deep-HMAX was able to achieve better performance than AlexNet, VsNet turned out to perform the best even with just five convolutional modules and the least amount of convolutional units. This suggests that the bypass connections and the parallel units within each module that provides flexibility in RF-sizes helped significantly in a model's performance on image classification. 100
4.3 Validation set error for the three models at the end of the 25th training epoch, for the multi-task fine-tuning stage. Only top-1 errors are reported as the top-5 errors are near 0 for all models. . . 101
4.4 Comparing the representational similarities between the layers of different models, in Spearman's ρ. The overall correlations are high, with layer 4 showing larger degree of differences between the three models. 106

Chapter 1

Introduction

The human visual system allows us to perceive the world that we live in. We have incredible abilities to recognize a wide range of object categories, from the simpler ones such as letters and digits, to complex ones such as computers and airplanes. Aside from our robust object recognition abilities, we are also able to track moving objects with ease, even living things that perform articulate movements. To better understand how the brain processes visual information with such efficiency and robustness, popular approaches include measuring the brain's electrical activity using EEG, and measuring the oxygenation of blood flow to the brain using fMRI. These important techniques are used for analyzing and understanding the mechanism of vision, at the neuronal level.

In this thesis work, the aim is to develop computational methods to model human behavioral data given some specific tasks. This is to model the higher-level (behavioral) outputs of the brain's visual processing, with the possibility that an accurate computational model of the behavior may shed some light on understanding how brain processes visual information, in that the assumptions of the computational model may be what is taking place in the brain's processing as well. This thesis mainly focuses on modeling the behavioral experiments of visual clutter perception, and object categorical search, where the models are developed with novel unsupervised computer vision and machine learning techniques.

In computer vision, researchers develop methods for machines to understand

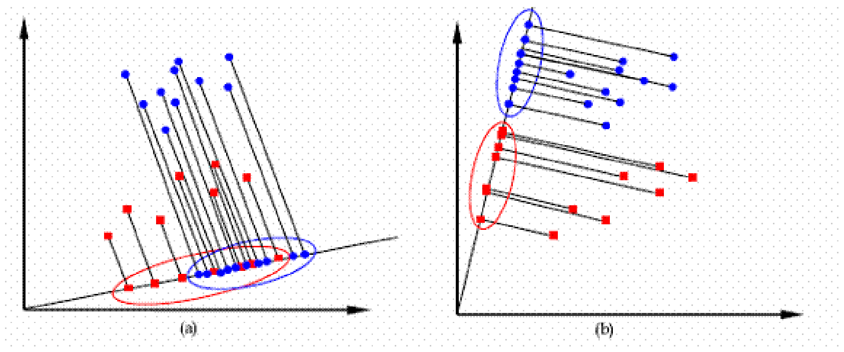


Figure 1.1. (a) shows that the projected subspace (a line) does not separate the blue and the red dots well, but complete separation between the two classes of dots can be achieved with the projected subspace in (b). Image source [3].

the content of images or videos, with tasks that include object recognition, tracking and localization, illuminations and intrinsic images, segmentations, video action recognition, and more. A large part of computer vision can also be viewed as modeling visual recognition problems with statistical machine learning algorithms, where machine learning algorithms can be roughly divided into supervised, and unsupervised methods.

Supervised learning methods mostly involve in training a classifier with the provided training data and labels, where the popular classifiers include Linear Discriminant Analysis, Logistic Regression, Naive Bayes, Decision Trees, AdaBoost, Support Vector Machines, and Artificial Neural Networks. Brief summaries of each of the mentioned methods are provided in the following.

Linear discriminant analysis (LDA) is a dimensionality reduction technique that is mostly used with two-class problems. LDA finds a projection matrix that projects the training data points onto a subspace, by maximizing and minimizing the between-class and within-class scatter matrices, respectively. As a result, the projected data points are optimally separable in a linear setting, shown in Figure 1.1.

Logistic regression is another very popular two-class classifier, where the classifier models the conditional probability of y — x with a logistic function, such that the training data closer to probability 0 are classified as class 1, and the data closer to probability 1 are classified as class 2, with the probabilities taken from the best fitted logistic function over all training data typically using Maximum Likelihood

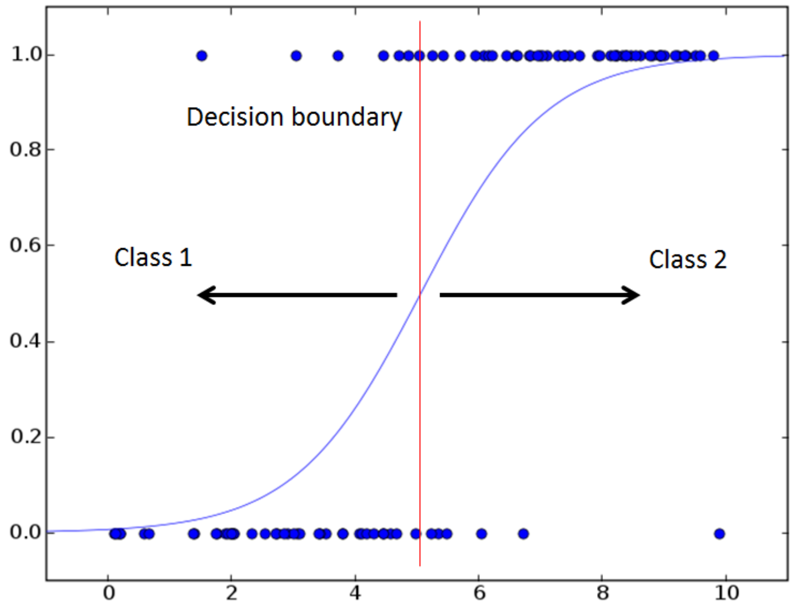


Figure 1.2. An example illustration of the logistic regression classifier. The blue dots are the 1D training data, the blue curve is the fitted logistic function, and the red line is the decision boundary. Data points that are at the left of the decision boundary are classified as class 1, otherwise class 2.

Estimator (MLE). Figure 1.2 shows that the decision boundary between the two classes lies in the middle of the fitted logistic function, which can be used for the final classification outcome. For the case of more than two classes, multinomial logistic regression classifier can be trained to perform the classification.

Naive Bayes is a classifier that is based on the Bayes Theorem:

$$P(C|D) \propto P(D|C) \times P(C) \tag{1.1}$$

The Bayes Theorem is also commonly understood as Posterior = Likelihood x Prior, where the Posterior probability is $P(C|D)$, the likelihood is $P(D|C)$, and the prior is $P(C)$. In Nave Bayes classifier, C is class label, D is the feature, and the prior is the normalized amount of class size. In the case where there are k classes and n features, the Nave Bayes decision rule is to take the most probable posterior probability given each of the classes (maximum a posteriori, MAP):

$$\hat{y} = \underset{k}{\operatorname{argmax}}\, p(C_k) \prod_{i=1}^{n} p(d_i|C_k) \tag{1.2}$$

An important point with Nave Bayes classifier is that it assumes the features are independent to one another. While features are rarely entirely independent from one another in the real world, the independence assumption of Nave Bayes works well in practice, which makes Nave Bayes classifier a popular and easy to use classification method.

Decision tree, as its name suggests, learns a tree (typically binary) from the features of the training data, where the decisions are made by traversing the tree from the root to the leaf nodes, with the leaf nodes being the classification results. In the binary classification scenario, the learning process begins with the root as the most discriminative feature or attribute, where the discriminative scores are computed via some feature ranking algorithm (i.e. mutual information), then the root is split into two children based on the feature value that best separates the two classes, with each child representing some other discriminative feature or attributes. Then, the children are further divided into more children based on the same process as finding the most separable feature value, and this process is repeated until confident classification of the training data is obtained at the leaf nodes.

Boosting classifiers are methods that form a final decision by employing a set of weak classifiers, with the most famous boosting example being the AdaBoost method [52]. The idea is that a strong classifier can be obtained from decisions formed by a set of weak classifiers, and AdaBoost demonstrated the success of this approach by using decision stumps as the weak classifier (simple thresholding). The AdaBoost algorithm runs in multiple iterations, with each iteration assigning weights to individual training examples, such that the incorrectly classified examples from the previous iteration are assigned higher weight, as to place more importance in correctly classifying these examples when learning the model parameters. AdaBoost has an advantage of being less prone to over-fitting, but is sometimes sensitive to noise and outliers. In contrast to the previously mentioned classifiers, AdaBoost is a non-linear classifier that is able to produce nonlinear

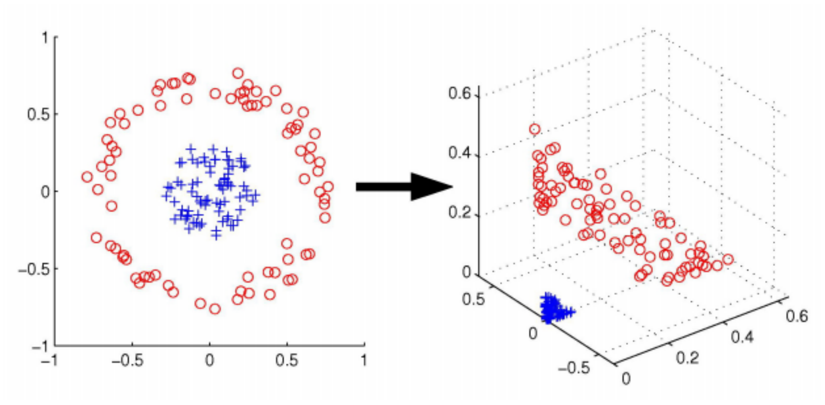


Figure 1.3. In the left plot, the blue and the red data points are forming a concentric circle and are not linearly separable; the right plot shows the result of mapping the original data with a polynomial kernel into a high-dimensional space, and the two classes are now linearly separable. Image source [2].

decision boundaries for more flexible classification results.

The most famous and widely used supervised learning method in the recent decades is the Support Vector Machine (SVM). SVM is a max-margin classifier, which means the model finds a hyperplane that best separates the classes with the maximum margin between the boarding data points, where those boarding data points are called the support vectors. In addition, different kernels can be applied to transform the data points into high-dimensional spaces, where the original non-linear problem could become linearly separable in the high-dimensional space, as shown in Figure 1.3. This famous mapping is called the kernel trick, and is an important part of SVM's success.

Finally, artificial neural networks (ANN) are very different kinds of supervised learning models, and an ANN is generally referred to as a "black box", due to the complexity of the network internals. An ANN is essentially a directed graph model, where a graph node is considered as a neuron, and a directed edge being a connection within a network. Each edge is weighted, and a neuron pools all the incoming weighted inputs by summing them, followed by a transformation using some nonlinear activation function (i.e. sigmoid). The neurons are organized into sequential layers of neurons, with each layer outputting information to the next layer in a feed-forward fasion. The ANN is trained via back-propagation, which updates the weights of all the connections by computing and propagating the error

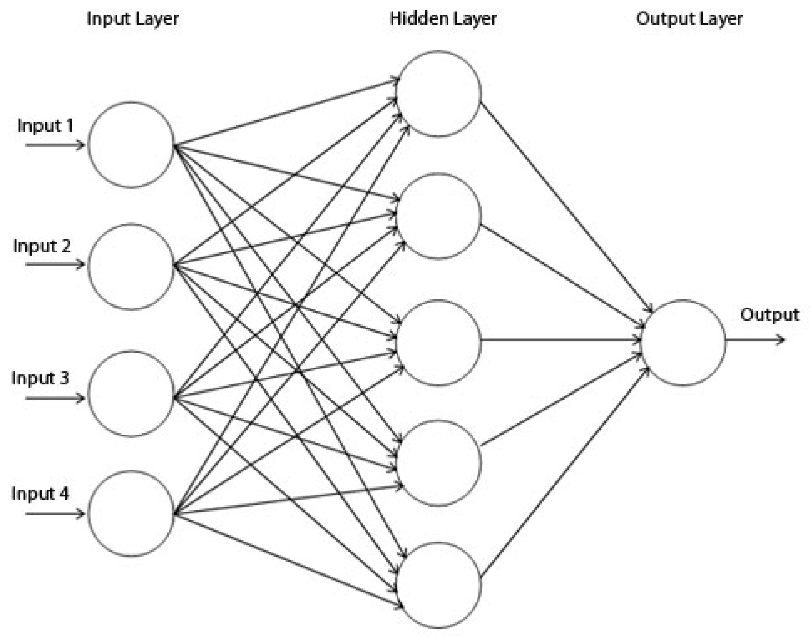


Figure 1.4. A simple 3-layer artificial neural network, with 4 input neurons at the input layer, 5 hidden neurons at the hidden layer, and a single output neuron at the output layer. This network can learn to produce real values at the output for regression problems, or learn to produce values close to 0 or 1 for binary classification problems. Image source [8].

gradient from the output layer back to the input layer. Figure 1.4 shows an example of a 3-layer network with one input layer, one hidden layer, and one output layer. ANNs are also nonlinear models due to the nonlinear activation functions at every neuron.

In contrast to the supervised methods, unsupervised methods learn from data without class labels, and this makes unsupervised learning very different from its supervised counterpart. Unsupervised learning methods are typically synonymous to clustering methods, which include k-means, mixture models, and hierarchical clustering. Unsupervised methods are also widely used for exploratory data analysis, such as principle component analysis (PCA), which is a popular dimensionality reduction technique. The followings is a brief overview of these widely used methods.

K-means clustering is the most well-known clustering method. It is fast, easy

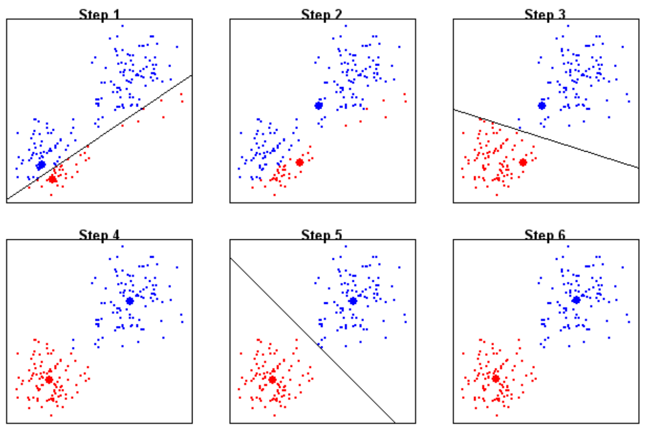


Figure 1.5. An example process of k-means clustering. Step 1 shows the randomly initialized cluster centers (red and blue thick dots), and as the iteration goes, the algorithm converges at step 6 with no more updates on the cluster centers. Image source [4].

to use, and straight-forward in concept and implementation. Given a set of data points and the number of desired clusters (k), k-means partitions the data points into k clusters that minimizes the sum-of-squares distances with each cluster members to the cluster centers. The k-means algorithm initializes the center of each cluster randomly, then iteratively repeats a two-step process: the assignment step that assigns each data point to the closest cluster in Euclidean distance, and the update step that computes and updates the new cluster centers for each cluster, based on mean Euclidean distance of all of its member data points, and this process ends when the update does not vary from the previous update. An example convergence of k-means is shown in Figure 1.5.

Mixture models in concept are very similar to k-means, but based on some parametric distribution as the model's building blocks. A mixture model is a probabilistic model that represents sub-populations of the entire data with some probabilistic function, most notably the Gaussian function, and the Gaussian Mixture Model (GMM). The learning and the fitting of a mixture model is just like k-means: first the distribution family is decided (i.e. Gaussian) along with the number of mixture components (k), the model parameters are then randomly chosen (or, with problem-specific heuristics) to begin the parameter optimization process. The parameter optimization is typically done using Expectation Maximization (EM), which is a two-step process: in the Expectation step, the membership

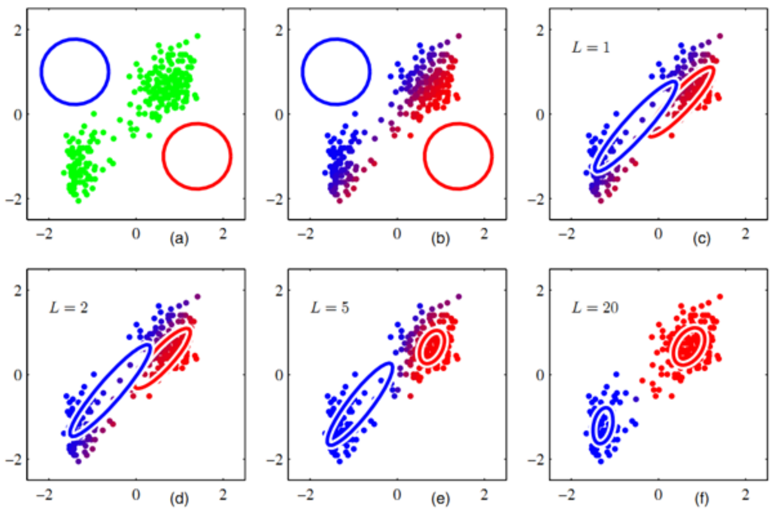


Figure 1.6. An example process of Gaussian mixture model. (a) shows the randomly initialized bivariate gaussian distributions (red and blue circles), and as the iteration goes, the algorithm converges at (f) with the model parameters becoming stable. Image source [6].

probabilities of the data points are computed given the current parameter settings; in the Maximization step, the model parameters are updated based on the MLE for the distribution components (i.e. Gaussian). Notice that the expectation and the maximization steps are just like the assignment and update step of k-means algorithm in concept, just with a probabilistic twist to them. An example for Gaussian mixture model's convergence is shown in Figure 1.6. One advantage of mixture model to k-means is that mixture models are able to provide soft-membership of data points to classes with the class conditional probabilities, while k-means produces hard class assignments of the data points.

Another unsupervised learning method is hierarchical clustering, and includes the agglomerative (bottom up) and the divisive (top down) variants. In agglomerative clustering, each data point is its own cluster, and a hierarchy of the clusters is built by merging clusters using some distance metric; the divisive variant is the opposite of agglomerative type, where all data points belong to a single cluster, and this cluster is divided into smaller clusters until every point is in its own cluster. Typically one obtains the desired clustering result by taking the clustering assignments at some hierarchy level via thresholding. Figure 1.7 shows an example of

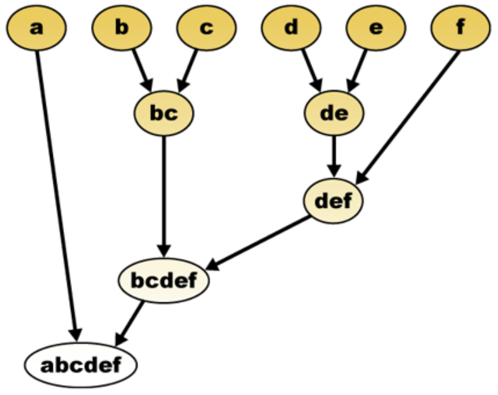


Figure 1.7. An example agglomerative clustering hierarchy. All six letters are in its own cluster initially (top), and they are iteratively merged into a single cluster in the last level. Image source [5].

the agglomerative hierarchy over six alphabets.

In a related but different vein, principle component analysis is a procedure that finds a smaller number of uncorrelated bases that are underlying the original data, and is a dimensionality reduction technique for both exploratory data analysis and predictive modeling. The uncorrelated basis (principle components) are found by performing eigenvalue decomposition over the covariance matrix of the dataset, where the principle components are ordered by the amount of data variance each explains. Figure 1.8 shows an example of the extracted principle components over a set of data points, where the first principle component is showing to explain more variance of the data then the second.

This thesis work utilizes unsupervised learning methods to achieve human behavioral modeling, drawing from the inspiration that a baby learns to recognize and process the surroundings with little to no supervision. However, there are vast numbers of different visual behavioral tasks that can be studied and modeled, therefore this thesis focuses on modeling two very different perceptual tasks: visual clutter perception, and the categorical object search performance. Finally, biologically-plausible deep learning models are also explored, and they are applied on the categorical object search modeling task for model comparison and evaluations.

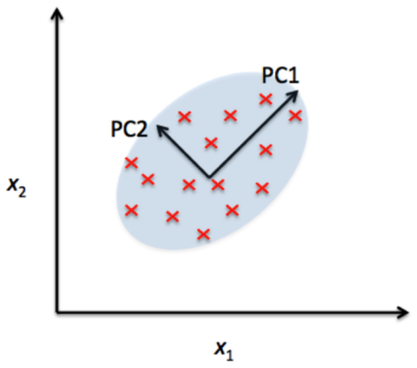


Figure 1.8. An example principle component analysis. PC1 and PC2 are the first and second principle components, and this plot shows that PC1 captures the most variance of the data, as the data forms more of an elliptical shape that is rotated roughly 45°, followed by the second principle component capturing the other part of the data spread. Image source [7].

Chapter 2

Modeling visual clutter perception

Clutter is defined colloquially as a “confused collection” or a “crowded disorderly state”. A more operational definition has also been proposed, defining clutter as the state in which excess items, or their representation or organization, lead to a degradation of performance at some task [134]. Whatever definition we choose, visual clutter is a factor affecting all of our lives in many ways. It affects how we look for things, how products are marketed and sold to us, the efficiency in which we interact with devices, and even whether we find some image aesthetically pleasing or not. For these reasons, clutter perception and its effects have been actively researched over the past decade in fields such as psychology and vision science, marketing, visualization, and interface design. The goal of this study is to apply techniques from computer vision to better quantify the perception of clutter, and ultimately to better understand how visual clutter affects human behavior.

The effects of visual clutter have been studied most aggressively in the context of a search task, where several models now exist describing how increasing clutter negatively impacts the time taken to find a target in a scene [95][134][159][92][26]. Fueling this interest in clutter among visual search researchers is the idea of a set size effect - the seminal finding showing that search performance degrades as objects are added to a display. Hundreds of studies have been devoted to describing and explaining set size effects [171], but the vast majority of these have been in the context of very simple displays consisting of well segmented objects. Quantifying set size in such displays is trivial - one need only count the number of objects. But how many objects are there in an image of a forest, or a city, or even a kitchen?

Is each tree or window a different object? What about each branch or patch of texture on a wall? It has even been argued that the task of quantifying a set size in a realistic scene is not only difficult, it may be ill-posed [106]. As the visual search community has moved over the past decade to more realistic scenes, they have therefore faced the prospect of abandoning their most cherished theoretical concept - the set size effect.

Visual clutter quantification has been suggested as a solution to this problem. Given that search performance also degrades with increasing clutter [107], clutter has been proposed as a surrogate measure of a set size effect, one that can be applied to images of scenes. One of the earliest attempts to model visual clutter used edge density - the ratio between the number of edge pixels in an image and the image size [95]. This was followed shortly after by the more elaborate feature congestion model, which estimates clutter in terms of the density of intensity, color, and texture features in an image [134]. Despite other more recent modeling efforts [159][92][26], the simplicity and early success of the feature congestion model has made it the standard model in comparisons of visual clutter perception.

More recent work showing limitations of the feature congestion model [61][107] have led to a renewed interest in quantifying clutter in terms of objects and *proto-objects* - patches of locally similar features, or simple fragments of coherent regions that form a stable object, detected at an early stage of human visual processing [124]. Proto-objects have been segmented from saliency maps [62][164], which are early representations of feature contrast in an image [66][65], and used in conjunction with color blob detectors to model visual attention [168]. Defining proto-objects in terms of salient blobs, however, is limited in that it produces elliptical proto-objects that may not capture the complexity of irregularly shaped objects in most images. It would be preferable to apply image segmentation methods to the entire image, allowing for more precise quantifications of the number of proto-objects in scenes and perhaps better estimates of a scene's set size and visual clutter.

Our visual clutter model uses the number of proto-objects segmented from an image to estimate clutter and to represent set size. Proto-object segmentation differs from standard image segmentation methods in one key respect. Standard methods aim to match human segmentation as ground truths, where each segment

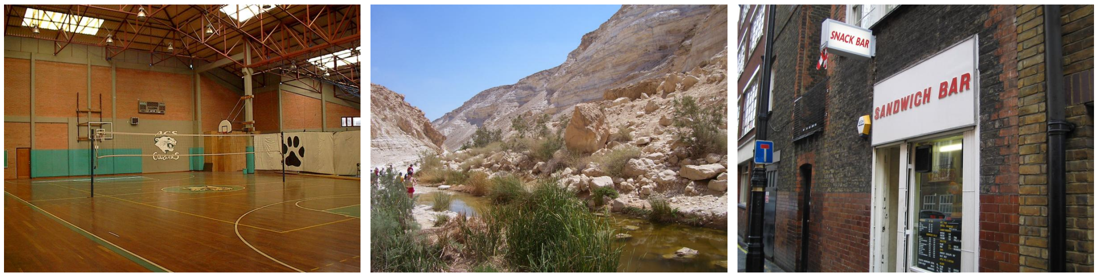


Figure 2.1. How do you quantify set size or the number of "things" in these images? Which scene is more cluttered?

corresponds to a complete and (hopefully) recognizable object. One example of this is the Berkeley Segmentation Dataset [14], a popular benchmark dataset for image segmentation methods. However, proto-objects are the fragments from which objects are built, and there exists no ground truth for proto-object segmentation that standard image segmentation methods can use to tune parameters. We therefore propose a method of proto-object segmentation for our visual clutter model that merges adjacent image fragments into a coherent proto-object region using a parametric graph-cut framework. Graph-cuts [142] is a popular image segmentation framework, in which pixels are the nodes connected to their neighbors via edges that are, weighted by a metric of node similarity using an edge cutting criterion. Segmentation is performed by removing edges from the graph using feature-based criteria commonly defined in non-parametric fashion (image features do not follow distributions on which to compute thresholds) [148][70][166][180].

We formulate our graph with image fragments (superpixels) as the nodes and focus on feature similarity distances between adjacent pairs of nodes on a graph. We perform a pair-wise comparison of feature histograms using Earth Mover's Distance (EMD) [136], based on the observation that EMD can be modeled by a Weibull distribution [29], which allows us to determine the graph cut criteria using parametric statistics.

2.1 Background

Modelling visual clutter comprises aspects of both psychology and computer vision, and it is important to discuss the literatures relating to this subject in both fields.

In this chapter, a background survey on notable and recent clutter models are discussed. Then, set-size and theories of proto-object related works are surveyed followed by image segmentation literatures.

2.1.1 Computational models of clutter

In one of the earliest studies relating clutter to visual search, [25] used simple stimuli, defined as objects composed of one material, and "compound" stimuli (Figure 2.2), defined as objects having two or more parts, and found an interesting interaction between this manipulation and clutter. Search performance for simple and compound stimuli was roughly comparable when these objects were arranged into sparse search displays. However, when these objects were densely packed in displays, a condition that would likely be perceived as more cluttered, search efficiency was found to degrade significantly for the compound objects. This observation led to their quantification of clutter using a Power Law model [26]. This model uses a graph-based image segmentation method ([48]) and a power law function having the form $y = cx^k$, where x is a smallest-segment-size parameter. Setting the exponent k to -1.32, they find the best fitting c and use it to estimate the clutter for a given image. Using 160 "what's-in-your-bag" images (http://whatsinyourbag.com/), they reported a correlation of 0.62 between these clutter estimates and behavioral search time [26].

As [25] were conducting their seminal clutter experiments, Rozenholtz and colleagues were developing their aforementioned Feature Congestion model of visual clutter [133]. This influential model extracts color, luminance, and orientation information from an image, with color and luminance obtained after conversion to CIElab color space ([38]) and orientation obtained by using orientation specific filters ([18]) to compute oriented opponent energy. The local variance of these features, computed through a combination of linear filtering and nonlinear operations, is then used to build a three-dimensional covariance ellipse. The volume of this ellipse therefore becomes a measure of feature variability in an image, which is used by the model as the clutter estimatethe larger the volume, the greater the clutter (Rosenholtz, et al., 2007). Using a variety of map stimuli, they tested their model against the Edge Density model ([95]) and found that both predicted search

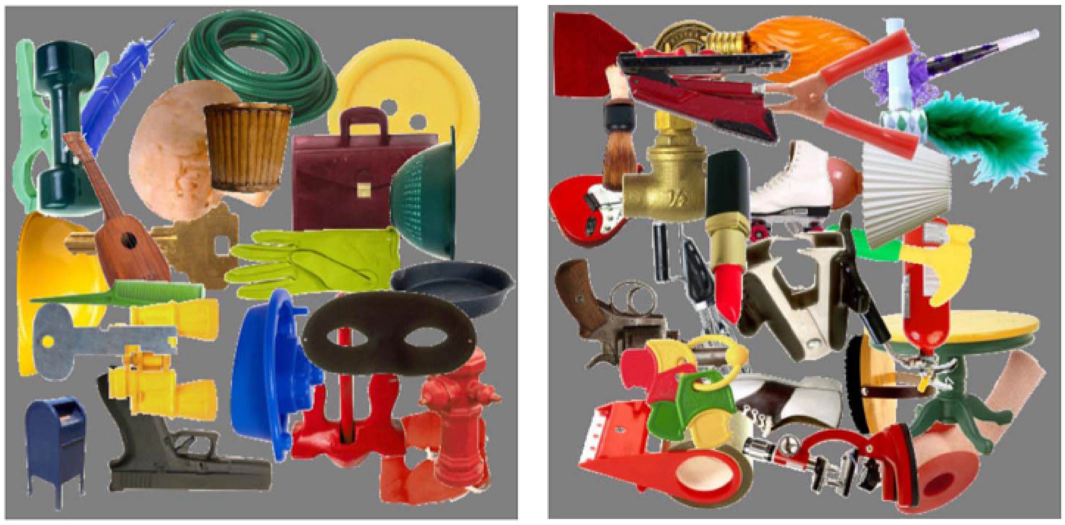


Figure 2.2. Left: 24 simple objects; Right: 24 compound objects, image source [26].

times reasonably well (Experiment 1; r = 0.75 for Feature Congestion; r = 0.83 for Edge Density). However, when the target was defined by the contrast threshold needed to achieve a given level of search performance (Experiment 2) the Feature Congestion model (r = 0.93) outperformed the Edge Density model (r = 0.83).

More recently, Lohrenz and colleagues ([92]) proposed their C3 (Color-Cluster Clutter) model of clutter, which derives clutter estimates by combining color density with global saliency. Color density is computed by clustering into polygons those pixels that are similar in both location and color. Global saliency is computed by taking the weighted average of the distances between each of the color density clusters. They tested their model in two experiments: one using 58 displays depicting six categories of maps (airport terminal maps, flowcharts, road maps, subway maps, topographic charts, and weather maps) and another using 54 images of aeronautical charts. Behavioral clutter ratings were obtained for both stimulus sets. These behavioral ratings were found to correlate highly with clutter estimates from the C3 model (r = 0.76 and r = 0.86 in Experiments 1 and 2, respectively), more so than correlations obtained from the Feature Congestion model (r = 0.68 and r = 0.75, respectively).

Another recent approach, the Crowding model of visual clutter, focuses on the density of information in a display [159]. Images are first converted to CIElab color

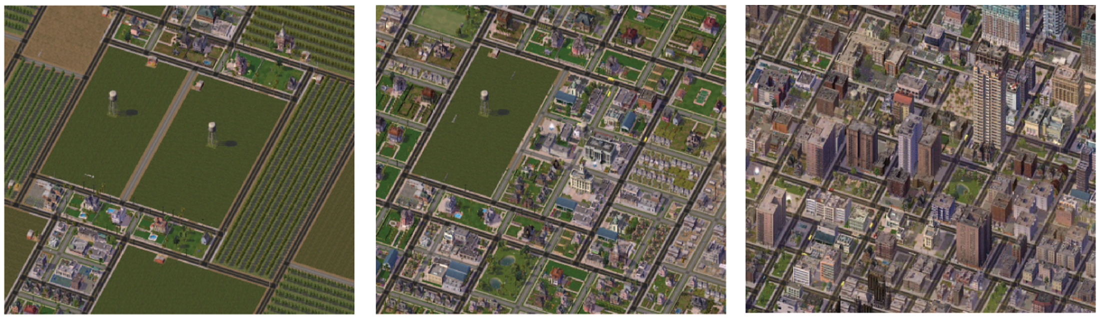


Figure 2.3. From left to right: time lapsed simcity images from rural, suburban, into urban. Image source [107]

space, then decomposed using oriented Gabors and a Gaussian pyramid ([30]) to obtain color and luminance channels. The luminance channel of the image is then filtered with difference-of-Gaussian filters to obtain a contrast image, and all of the channels are post-processed with local averaging. It is this local averaging that is hypothesized to be the mechanism of crowding under this model. The channels are then pooled by taking a weighted average with respect to the center of the image, resulting in a progressive blurring radiating out from the images center. Pooled results are compared to the original channels using a sliding window that computes the KL-divergence between the two, thereby quantifying the loss of information due to possible crowding, and this procedure is repeated over all scales and features and finally combined by taking a weighted sum to produce the final clutter score. They evaluated their model on the 25 map images used to test the original version of the Feature Congestion model [133], and found a comparable correlation with the behavioral ratings ($r = 0.84$; [160]).

More recently, Henderson and colleagues studied the influence of clutter on real-world images with search performance and eye-movement [61], which is a change from the work done previously using synthetic images. They found that clutter does have a significant impact on search performance, and it can be used as a stand-in for set size in real-world scenes. This reiterates the importance of quantifying visual clutter in terms of set size as pointed out by Neider and Zelinsky in 2011 using time lapsed SimCity images (Fig. 2.3), showing that feature congestion does not explain enough of clutter in terms of conceptual congestion (different numbers and types of buildings) [107].

2.1.2 Image segmentation and proto-objects

Motivating the study of clutter is the assumption that objects cannot be meaningfully segmented from images of arbitrary scenes, but is this true? The computer vision community has been working for decades on this problem and has made good progress. Of the hundreds of scholarly reports on this topic, the ones that are most relevant to the goal of quantifying the number of objects in a scene (i.e., obtaining a set size) are those that use an unsupervised analysis of an image that requires no prior training or knowledge of particular object classes. Among these methods, the most popular have been Normalized Cut [142], Mean-shift image segmentation [42], and a graph-based method developed by [48]. However, despite clear advances and an impressive level of success, these methods are still far from perfect. Crucially, these methods are typically evaluated against a ground truth of object segmentations obtained from human raters (the Berkeley segmentation dataset; [14][96]), which, as already discussed, is purely subjective and also imperfect. This reliance on a human ground truth means that image segmentation methods, regardless of how accurate they become, will not be able to answer the question of how many objects exist in a scene, as this answer ultimately depends on what people believe is, and is not, an object.

Recognizing the futility of obtaining objective and quantifiable counts of the objects in scenes, the approach taken by most existing models of clutter (reviewed above) has been to abandon the notion of objects entirely. The clearest example of this is the Feature Congestion model, which quantifies the feature variability in an image irrespective of any notion of an object. Abandoning objects altogether, however, seems to us an overly extreme conceptual movement in the opposite direction, and that there exists an alternative that finds a middle ground; rather than attempting to quantify clutter in terms of features or objects, attempt this quantification using something between the two proto-objects.

The term proto-object, or pre-attentive object [120], was coined by [126] and elaborated in later work by Rensink on coherence theory [124]. Coherence theory states that proto-objects are low-level representations of feature information computed automatically by the visual system over local regions of space, and that attention is the process that combines or groups these proto-objects to form objects. Under this view proto-objects are therefore the representations from which objects

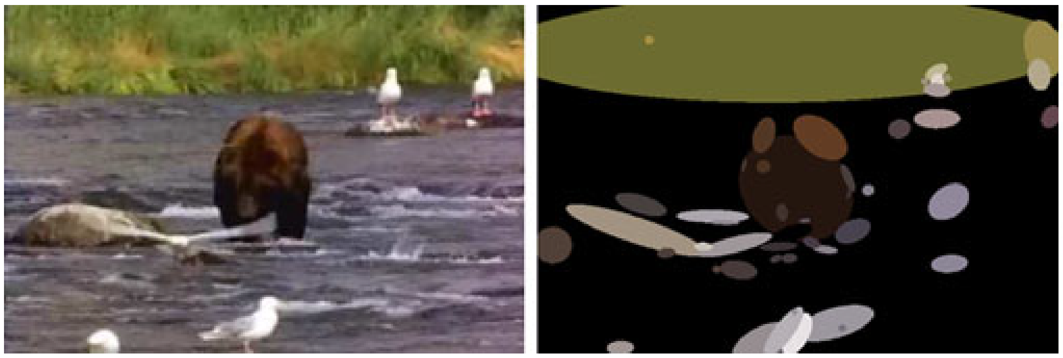


Figure 2.4. Left: input image; Right: proto-object in color blob representation using color blob detector from [49]. Image source [168].

are built, with attention being the metaphorical hand that holds them together. Part of the appeal of proto-objects is that they are biologically plausiblerequiring only the grouping of similar low-level features from neighboring regions of space. This is consistent with the integration of information over increasingly large regions of space as processing moves farther from the feature detectors found in V1 [114][47].

Since their proposal, proto-objects have appeared as prominent components in several models of visual attention. Orabona and colleagues ([116]) proposed a model based on proto-objects that are segmented using blob detectors, operators that extract blobs using Difference-of-Gaussian or Laplacian-of-Gaussian filters [90], which are combined into a saliency map for their visual attention model. A similar approach was adopted by Wischnewski and colleagues [169], who proposed a model of visual attention that uses a color blob detector ([49], Figure 2.4) to form proto-objects. These proto-objects are then combined with the Theory of Visual Attention (TVA, [28]) to produce a priority map that captures both top-down and bottom-up contributions of attention, with the bottom-up contribution being the locally grouped features represented by proto-objects. Follow-up work has since extended this proto-object based model from static images to video, thereby demonstrating the generality of the approach [168].

2.1.3 The proto-object model of clutter perceiton

Underlying our approach is the assumption that, whereas quantifying and counting the number of objects in a scene is a futile effort, quantifying and counting proto-objects is not. We define proto-objects as fragments of locally similar features that become merged to create coherent regions that can be used by the visual system to build perceptual objects. While conceptually related to other proto-object segmentation approaches reported in the behavioral vision literature, our approach differs from these in one key respect. Although previous approaches have used blob detectors to segment proto-objects from saliency maps [62][164], bottom-up representations of feature contrast in an image [66][65], or applied color blob detectors to an actual image or video [168], this reliance on blob detection likely results in only a rough approximation of the information used to create proto-objects. Blob detectors, by definition, constrain proto-objects to have an elliptical shape, and this lose of edge information might be expected to lower the precision of any segmentation. The necessary consequence of this is that approaches using blob detection will fail to capture the fine-grained spatial structure of irregularly shaped real-world objects. It would be preferable to extract proto-objects using methods that retain this spatial structure so as to better approximate the visual complexity of objects in our everyday world. For this we turn to image segmentation methods from computer vision.

We propose the Proto-object model of clutter perception, which combines superpixel image segmentation with a clustering method that we propose in the next section, to merge featurally similar superpixels into proto-objects. Our Proto-object model differs from standard image segmentation methods in one important respect. Standard methods aim to match extracted segments to a labeled ground truth segmentation of objects, as determined by human observers, where each segment corresponds to a complete and (hopefully) recognizable object. One example of this is the Berkeley Segmentation Dataset [14], a currently popular benchmark against which image segmentation methods can be tested and their parameters tuned. However, proto-objects are the fragments from which objects are built, making these object-based ground truths not applicable. Nor is it reasonable to ask observers to reach down into their mid-level visual systems to perform a comparable labeling of proto-objects. For better or for worse, there exists no ground truth for proto-object segmentation that can be used to evaluate models or tune parameters. We therefore use as a ground truth behaviorally-obtained rankings of image clutter, and then determine how well our Proto-object model, and the models of others, can predict these rankings. Our approach is therefore interdisciplinary, applying superpixel segmentation and clustering methods from computer vision to the task of modeling human clutter perception.

2.2 Parametric graph partitioning for proto object segmentation (PGP)

In this section, we describe a novel clustering method called the parametric graph partitioning (PGP) for proto-object segmentation step by step: first we pre-process an image into superpixels, compute pair-wise superpixel feature similarity using EMD on three feature histograms (intensity, color, texture); then we fit a mixture of Weibull distributions over all the pair-wise similarity distances (EMD) using non-linear least squares to obtain the optimal feature similarity threshold, which serves as our graph-cut criterion for merging similar superpixels into a coherent proto-object region. We call this general parametric edge modeling as parametric graph partitioning (PGP)

2.2.1 Superpixel preprocessing and feature similarity comparison

To merge similar fragments into a coherent proto-object region, the term fragment and the measure of coherence (similarity) must be defined. We define an image fragment as a group of pixels that share similar low-level image features: intensity, color, and orientation. This conforms with processing in the human visual system, and also makes a fragment analogous to an image superpixel, which is a perceptually meaningful atomic region that contains pixels similar in color and texture [163]. However, superpixel segmentation methods in general produce a fixed number of superpixels from an image, and groups of nearby superpixels may belong to the same proto-object due to the intended over-segmentation. Therefore, we extract superpixels as image fragments for pre-processing, and subsequently merge

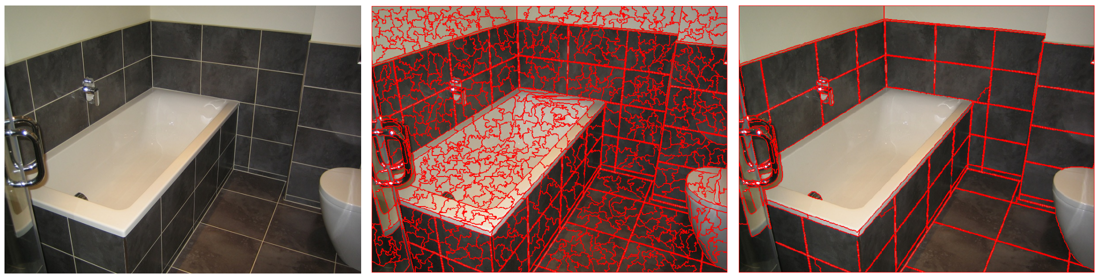


Figure 2.5. Left: image of a bathroom. Middle: after superpixel pre-processing [91]. Right: final proto-object segmentation result.

similar superpixels into proto-objects. We define that a pair of adjacent superpixels belong to a coherent proto-object if they are similar in *all* three low-level image features. Thus we need to determine a similarity threshold for each of the three features, that separates the similarity distance values into "similar", and "dissimilar" populations, detailed in Section 2.2.2.

In this work, the similarity statistics are based on comparing histograms of intensity, color, and orientation features from an image fragment. The intensity feature is a 1D 256 bin histogram, the color feature is a 76×76 (8 bit color) 2D histogram using hue and saturation from the HSV colorspace, and the orientation feature is a symmetrical 1D 360 bin histogram using gradient orientations, similar to the HOG feature [45]. All three feature histograms are normalized to have the same total mass, such that bin counts sum to one.

We use Earth Mover's Distance (EMD) to compute the similarity distance between feature histograms [136], which is known to be robust to partially matching histograms. For any pair of adjacent superpixels v_a and v_b, their normalized feature similarity distances for each of the intensity, color, and orientation features are computed as: $x_{n;f} = \text{EMD}(v_{a;f}, v_{b;f})/\widehat{\text{EMD}}_f$, where $x_{n;f}$ denotes the similarity (0 is exactly the same, and 1 means completely opposite) between the n^{th} pair ($n = 1, ..., N$) of nodes v_a and v_b under feature $f \in \{i, c, o\}$ as intensity, color, and orientation. $\widehat{\text{EMD}}_f$ is the maximum possible EMD for each of the three image features; it is well defined in this situation such that the largest difference between intensities is black to white, hues that are 180° apart, and a horizontal gradient against a vertical gradient. Therefore, $\widehat{\text{EMD}}_f$ normalizes $x_{n;f} \in [0, 1]$. In the subsequent sections, we explain our proposed method for finding the adaptive similarity threshold from $\mathbf{x}_f$, which is the EMDs of all pairs of adjacent nodes.

2.2.2 Weibull distribution and EMD for adaptive similarity threshold

Any pair of adjacent superpixels are either similar enough to belong to the same proto-object, or they belong to different proto-objects, as separated by the adaptive similarity threshold γ_f that is different for every image. We formulate this as an edge labeling problem: given a graph $G = (V, E)$, where $v_a \in V$ and $v_b \in V$ are two adjacent nodes (superpixels) having edge $e_{a,b} \in E, a \neq b$ between them, also the n^{th} edge of G. The task is to label the binary indicator variable $y_{n;f} = I(x_{n;f} < \gamma_f)$ on edge $e_{a,b}$ such that $y_{n;f} = 1$ if $x_{n;f} < \gamma_f$, which means $v_{a;f}$ and $v_{b;f}$ are similar (belongs to the same proto-object), otherwise $y_{n;f} = 0$ if $v_{a;f}$ and $v_{b;f}$ are dissimilar (belongs to different proto-objects). Once γ_f is computed, removing the edges such that $\mathbf{y}_f = 0$ results in isolated clusters of locally similar image patches, which are the desired groups of proto-objects.

Intuitively, any pair of adjacent nodes is either within the same proto-object cluster, or between different clusters ($y_{n;f} = \{1, 0\}$), therefore we consider two populations (the within-cluster edges, and the between-cluster edges) to be modeled from the density of $\mathbf{x}_f$ in a given image. In theory, this would mean that the density of $\mathbf{x}_f$ is a distribution exhibiting bi-modality, such that the left mode corresponds to the set of $\mathbf{x}_f$ that are considered similar and coherent, while the right mode contains the set of $\mathbf{x}_f$ that represent dissimilarity. At first thought, applying k-means with $k = 2$ or a mixture of two Gaussians would allow estimation of the two populations. However, there is no evidence showing that similarity distances follow symmetrical or normal distributions. In the following, we argue that the similarity distances $\mathbf{x}_f$ computed by EMD follow Weibull distribution, which is a distribution of the Exponential family that is skewed in shape.

The EMD is defined to minimize the following with an optimal flow f'_{ij}:

$$EMD(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} f'_{ij} d_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f'_{ij}} \tag{2.1}$$

$$\text{s.t.} \quad \sum_j f'_{ij} \le p_i \ , \ \sum_i f'_{ij} \le q_j \ , \ \sum_{i,j} f'_{i,j} = \min(\sum_i p_i, \sum_j q_j) \ , \ f'_{ij} \ge 0 \tag{2.2}$$

where $P = \{(x_1, p_1), ..., (x_m, p_m)\}$ and $Q = \{(y_1, q_1), ..., (y_n, q_n)\}$ are the two signatures to be compared, and d_{ij} denotes a dissimilarity metric (i.e. L_2 distance) between x_i and y_j in R^d. When P and Q are normalized to have the same total mass, EMD becomes identical to Mallows distance [86], defined as:

$$M_p(X,Y) = (\frac{1}{n}\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p} \tag{2.3}$$

where X and Y are sorted vectors of the same size, and Mallows distance is an L_p-norm based distance measurement. Furthermore, L_p-norm based distance metrics are Weibull distributed if the two feature vectors to be compared are correlated and non-identically distributed [29]. We show that our features assumptions are satisfied in Section 2.2.7.1. Hence, we can model each feature of $\mathbf{x}_f$ as a mixture of two Weibull distributions separately, and compute the corresponding γ_f as the boundary locations between the two components of the mixtures. Although the Weibull distribution has been used in modeling actual image features such as texture and edges [53][179], it has not been used to model EMD similarity distance statistics until now.

2.2.3 Weibull mixture model

Our Weibull mixture model (WMM) takes the following general form:

$$\mathcal{W}^K(\mathbf{x};\theta) = \sum_{k=1}^{K} \pi_k \phi_k(\mathbf{x};\theta_k) \ , \quad \phi(x;\alpha,\beta,c) = \frac{\beta}{\alpha}(\frac{x-c}{\alpha})^{\beta-1} e^{-(\frac{x-c}{\alpha})^{\beta}} \tag{2.4}$$

where $\theta_k = (\alpha_k, \beta_k, c_k)$ is the parameter vector for the k^{th} mixture component, and ϕ denotes the three-parameter Weibull pdf with the scale (α), shape (β), and location (c) parameter, and the mixing parameter π such that $\sum_k \pi_k = 1$. In this case, our two-component WMM contains a 7-parameter vector $\theta = (\alpha_1, \beta_1, c_1, \alpha_2, \beta_2, c_2, \pi)$ that yields the following complete form:

$$\mathcal{W}^2(\mathbf{x};\theta) = \pi(\frac{\beta_1}{\alpha_1}(\frac{\mathbf{x}-c_1}{\alpha_1})^{\beta_1-1})e^{-(\frac{\mathbf{x}-c_1}{\alpha_1})^{\beta_1}} + (1-\pi)(\frac{\beta_2}{\alpha_2}(\frac{\mathbf{x}-c_2}{\alpha_2})^{\beta_2-1})e^{-(\frac{\mathbf{x}-c_2}{\alpha_2})^{\beta_2}} \quad (2.5)$$

To estimate the parameters of $\mathcal{W}^2(\mathbf{x};\theta)$, we tested two optimization methods: maximum likelihood estmation (MLE), and nonlinear least squares minimization (NLS). Both MLE and NLS requires an initial parameter vector θ' to begin the optimization, and the choice of θ' is crucial to the convergence of the optimal parameter vector $\hat{\theta}$. In our case, the initial guess is quite well defined: for any node of a specific feature $v_{j;f}$, and its set of adjacent neighbors $\mathbf{v}^N_{j;f} = N(v_{j;f})$, the neighbor that is most similar to $v_{j;f}$ is most likely to belong to the same cluster as $v_{j;f}$, and it is especially true under an over-segmentation scenario. Therefore, the initial guess for the first mixture component $\phi_{1;f}$ is the MLE of $\phi_{1;f}(\theta'_{1;f};\mathbf{x}'_f)$, such that $\mathbf{x}'_f = \{\min(\text{EMD}(v_{j;f}, \mathbf{v}^N_{j;f}))|v_{j;f}; j = 1, ..., z, f \in \{i, c, o\}\}$, where z is the total number of superpixels, and $\mathbf{x}'_f \subset \mathbf{x}_f$. After obtaining $\theta'_1 = (\alpha'_1, \beta'_1, c'_1)$, several θ'_2 can be computed for the re-start purpose via MLE from the data taken by $Pr(\mathbf{x}_f|\theta'_1) > \mathbf{p}$, where Pr is the cumulative distribution function, and $\mathbf{p}$ is a range of percentiles. Together, they form the complete initial guess parameter vector $\theta' = (\alpha'_1, \beta'_1, c'_1, \alpha'_2, \beta'_2, c'_2, \pi')$ where $\pi' = 0.5$.

2.2.4 Parameter estimation

Maximum likelihood estimation (MLE) is a criterion for estimating the parameters by maximizing the log-likelihood function of the observed samples. The log-likelihood function of $\mathcal{W}^2(\mathbf{x};\theta)$ is given by:

$$\begin{split}\ln\mathcal{L}(\theta;\mathbf{x}) = \sum_{n=1}^{N} \ln\{\pi(\frac{\beta_1}{\alpha_1}(\frac{x_n-c_1}{\alpha_1})^{\beta_1-1})e^{-(\frac{x_n-c_1}{\alpha_1})^{\beta_1}} + \\ (1-\pi)(\frac{\beta_2}{\alpha_2}(\frac{x_n-c_2}{\alpha_2})^{\beta_2-1})e^{-(\frac{x_n-c_2}{\alpha_2})^{\beta_2}}\}\end{split} \quad (2.6)$$

Due to the complexity of this log-likelihood function and the presence of the location parameters $c_{1,2}$, we adopt the Nelder-Mead method as a derivative-free optimization of MLE that performs parameter estimation with direct-search [108][82],

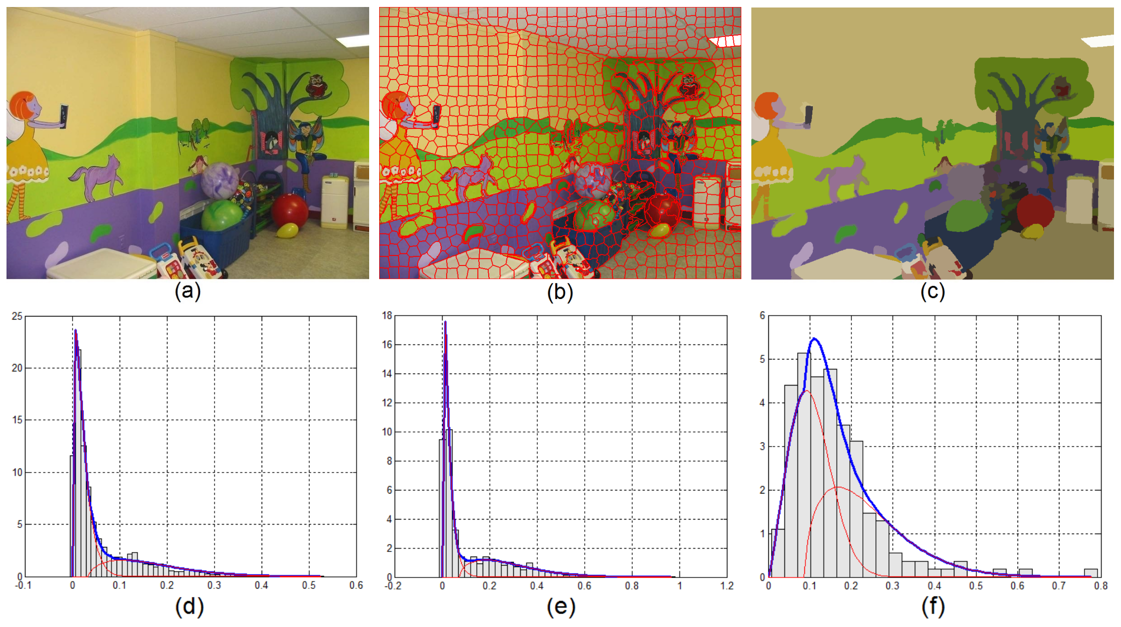


Figure 2.6. (a) original image, (b) after superpixel pre-processing [10] (977 initial segments), (c) final proto-object partitioning result (150 segments). Each final segment is shown with its mean RGB value to approximate proto-object perception. (d) $\mathcal{W}^2(\mathbf{x}_f; \theta_f)$ optimized using the Nelder-Mead algorithm for intensity, (e) color, and (f) orientation based on the image in (b). The red line indicates the individual Weibull components; and the blue line is the density of the mixture $\mathcal{W}^2(\mathbf{x}_f; \theta_f)$.

by minimizing the negative log-likelihood function of Eq. 2.6.

For the NLS optimization method, first $\mathbf{x}_f$ are approximated with histograms much like a box filter that smoothes a curve. The appropriate histogram bin-width for data representation is computed by $w = 2(IQR)n^{-1/3}$, where IQR is the interquartile range of the data with n observations [67]. This allows us to optimize a two component WMM to the height of each bin with NLS as a curve fitting problem, which is a robust alternative to MLE when the noise level can be reduced by some approximation scheme. Then, we find the least squares minimizer by using the trust-region method [147][154]. Both the Nelder-Mead MLE algorithm and the NLS method are detailed in the Appendix.

Figure 2.6 shows the WMM fit using the Nelder-Mead MLE method. In addition to the good fit of the mixture model to the data, it also shows that the right skewed data (EMD statistics) is remarkably Weibull, this further validates that EMD statistics follow Weibull distribution both in theory and experiments.

2.2.5 Visual clutter model with model selection

At times, the dissimilar population can be highly mixed in with the similar population, the density of which would resemble more of a single Weibull in shape such as Figure 2.6d. Therefore, we fit a single Weibull as well as a two component WMM over $\mathbf{x}_f$, and apply the Akaike Information Criterion (AIC) to prevent any possible over-fittings by the two component WMM. AIC tends to place a heavier penalty on the simpler model, which is suitable in our case to ensure that the preference is placed on the two-population mixture models. For models optimized using MLE, the standard AIC is used; for the NLS cases, the *corrected* AIC (AICc) for smaller sample size (generally when $n/k \leq 40$) with residual sum of squares (RSS) is used, and it is defined as $AICc = n\ln(RSS/n) + 2k + 2k(k+1)/(n-k-1)$, where k is the number of model parameters, n is the number of samples.

The optimal γ_f can then be determined as follows:

$$\gamma_f = \begin{cases} \max(x, \epsilon), \quad \text{s.t. } \pi_1\phi_{1;f}(x|\theta_{1;f}) = \pi_2\phi_{2;f}(x|\theta_{2;f}) & \text{AIC}(\mathcal{W}^2) \leq \text{AIC}(\mathcal{W}^1) \\ \max(-\alpha_1(\ln(1-\tau))^{1/\beta_1}, \epsilon) & \text{Otherwise} \end{cases} \tag{2.7}$$

The first case is when the mixture model is preferred, then the optimal γ_f is the crossing point between the mixture components, and the equality can be solved in linear time by searching over the values of the vector $\mathbf{x}_f$; in the second case when the single Weibull is preferred by model selection, γ_f is calculated by the inverse CDF of $\mathcal{W}^1$, which computes the location of a given percentile parameter τ. Note that γ_f is lower bounded by a tolerance parameter ϵ in both cases to prevent unusual behaviors when an image is nearly blank ($\gamma_f \in [\epsilon, 1]$), making τ and ϵ the only model parameters in our framework.

We perform Principle Component Analysis (PCA) on the similarity distance values $\mathbf{x}_f$ of intensity, color, and orientation and obtain the combined distance feature value by projecting $\mathbf{x}_f$ to the first principle component, such that the relative importance of each distance feature is captured by its variance through PCA. This projected distance feature is used to construct a minimum spanning tree over the superpixels to form the structure of graph G, which weakens the inter-cluster connectivity by removing cycles and other excessive graph connections. Finally, each edge of G is labeled according to Section 2.2.2 given the computed γ_f, such that an edge is labeled as 1 (similar) only if the pair of superpixels are similar in all three features. Edges labeled as 0 (dissimilar) are removed from G to form isolated clusters (proto-objects), and our visual clutter model produces a normalized clutter measure that is between 0 and 1 by dividing the number of proto-objects by the initial number of superpixels such that it is invariant to different scales of superpixel over-segmentation.

2.2.6 Behavioral data

Behavioral data collection was limited to the creation of a set of clutter ranked images. We did this out of concern that the previous image sets used to evaluate models were limited in various respects, especially in that some of these sets contained only a small number of images and that some scene types were disproportionately represented among these imagesboth factors that might severely reduce their generalization to realistic scenes. Specifically, these image sets were: 25 depictions of U.S. maps and weather forecasts [133]; 58 depictions of six map categories in varying resolution, including airport terminal maps, flowcharts, road maps, subway maps, topographic charts, and weather maps [92]; 25 depictions of 6, 12, or 24 objects arranged into organized arrays [25][159]; 60 images of real-world scenes with embedded T and L stimuli [61], 90 images of synthetic cities obtained from the game SimCity [107], and 160 images depicting the contents of handbags [26]. We were also concerned that two different ground truths were used by these studies. Whereas some studies evaluated models against a ground truth of explicit clutter rankings of images [133][92][159][107], other studies adopted an indirect clutter ground truth consisting of manual and oculomotor measures of search efficiency [134][26][61][17]. Although a search-based ground truth is closer to the goal of using clutter as a surrogate measure of set size, it raises the possibility that these models were predicting something specific to the search behavior and were not estimating clutter perception per se.

Our goal in this study was to model human clutter perception, leaving the task of modeling the relationship between clutter and search to future work. To accomplish this we used a relatively large set of images of random-category scenes that were explicitly rank ordered by observers for visual clutter. We then determined how well models of clutter, including the Proto-object model introduced here, could predict these clutter rankings.

Stimuli consisted of 90 800 600 images of real-world random-category scenes from the SUN09 image collection [172]. Image selection was restricted to those scenes for which there existed behaviorally-obtained segmentations of the objects, and consequently, object counts. We did this so as to have another point of comparison for our evaluationhow well does the number of objects in a scene predict clutter perception, and how does this compare to predictions from the models? Object count was also used to ensure that selected scenes spanned a reasonable range of image complexity, at least with respect to the number of objects in the scenes. This was done by selecting images so as to fill 6 object count bins, with each bin having 15 scenes. Bin 1 contained images with object counts in the 1-10 range, Bin 2 contained images with object counts in the 11-20 range, up to Bin 6 which contained images with object counts in the 51-60 range. Other than these constraints, and the resolution restriction (800 $\times$ 600 pixels), image selection was random from the targeted subset of SUN09 images. Given that the accuracy and detail of the object segmentations vary greatly in the SUN09 collection, selected images were also visually inspected to make certain that they were not under-segmented. Figure 2.7 shows these behaviorally-obtained object segmentations for 4 of the 90 images used in this study.

Fifteen undergraduate students from Stony Brook University (ages 18 to 30) participated for course credit. All had normal or corrected-to-normal vision, by self-report, and were nave with respect to how their clutter judgments would be used. Participants were told that they would see 90 images, one at a time, and would have to rank order these images from least to most in visual clutter. No definition of clutter was suggested to participants; they were instead instructed to come up with their own definition and to use it consistently throughout their task. Participants were asked to rank order a 12-image list of practice scenes, so as to help them formulate their idiosyncratic clutter scale and to familiarize them with the interface used to assign these rankings (described below). Immediately following the practice block was the experimental block, in which participants had

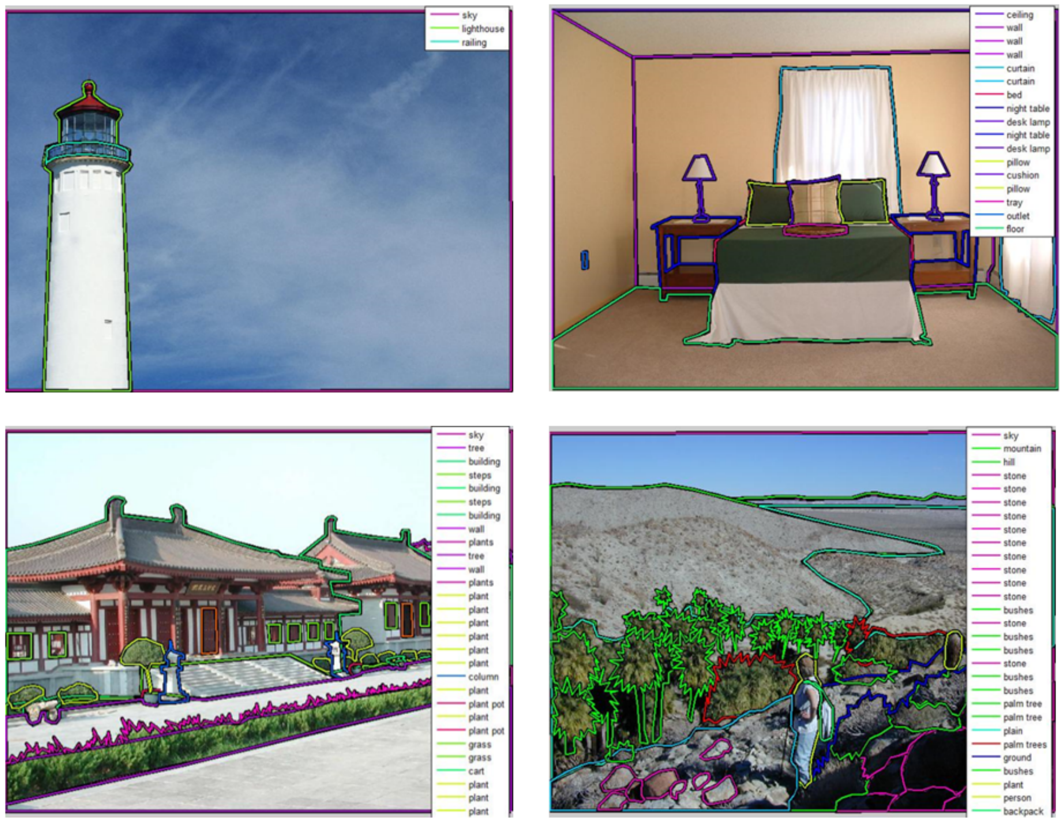


Figure 2.7. Object segmentations from human observers for 4 of the 90 scenes used in this study. Segmentations were provided as part of the SUN09 image collection. To the right of each are lists of the object segment labels (object counts), in matching colors. Top-left: 3 objects. Top-right: 17 objects. Bottom-left: 48 objects. Bottom-right: 57 objects.

to rank order the 90 images selected using the above-described criteria. Each testing session, practice and experimental rankings combined, lasted between 60 and 90 minutes, and an experimenter remained in the room with participants so as to observe whether they were making thoughtful deliberations about where each image should be inserted into the rank-ordered clutter sequence.

Participants made their clutter rankings using an interface written in MATLAB and running on a Windows 7 PC with two LCD monitors. The monitors were arranged vertically, with one on top and the other below. The bottom monitor displayed the scene that was currently being ranked for clutter, which subtended a visual angle of approximately 27° horizontally and 20° vertically. Images were presented to each participant in a different randomized order so as to remove potential bias. The top monitor displayed pairs of scenes that had already been ranked, with the less cluttered scene on the left and the more cluttered scene on the right. Importantly, the program kept an ongoing record of the images that had been rank ordered by the participant. This means that, although only two images were displayed at any one time on the top monitor, participants were able to scroll through the entire set of images that they had ranked when deciding where in this list the new image (displayed on the bottom monitor) should be inserted. Once an image was inserted into the rank ordered set, that image would be displayed on the top monitor (left position) and a new image would appear on the bottom monitor. This procedure repeated until all 90 images were ranked for clutter, with the leftmost image in the set having the lowest clutter ranking and the rightmost image having the highest. If at any time during the ranking procedure participants felt that they had made a mistake, or if their clutter scale simply evolved and they wanted to re-rank the images, the opportunity existed for them to do so (and they were encouraged to avail themselves of this opportunity). This could be accomplished by selecting and removing an image from the rank-ordered set, then re-inserting it in the new desired location.

2.2.7 Experiment and results

2.2.7.1 Image feature assumptions

In their demonstration that similarity distances adhere to a Weibull distribution, Burghouts et al. [29] derived and related L_p-norm based distances from the statistics of sums [19][20] such that for non-identical and correlated random variables X_i, the sum $\sum_{i=1}^{N} X_i$ is Weibull distributed if X_i are upper-bounded with a finite N, where $X_i = |s_i - T_i|^p$ such that N is the dimensionality of the feature vector, i is the index, and s, $t \in T$ are different sample vectors of the same feature.

The three image features used in this model are finite and upper bounded, and we follow the procedure from [29] with L_2 distance to determine whether they are correlated. We consider distances from one reference superpixel feature vector s to 100 other randomly selected superpixel feature vectors T (of the same feature), and compute the differences at index i such that we are obtaining the random variable $X_i = |s_i - T_i|^p$. Pearson's correlation is then used to determine the relationship between X_i and X_j, $i \neq j$ at a confidence level of 0.05. This procedure is repeated 500 times per image for all three feature types over all 90 images. As predicted, we found an almost perfect correlation between feature value differences for each of the features tested: Intensity: 100%, Hue: 99.2%, Orientation: 98.97%). This confirms that the low level image features used in this study follow a Weibull distribution.

2.2.7.2 Model evaluation

We ran our model with different parameter settings of $\epsilon \in \{0.01, 0.02, ..., 0.20\}$ and $\tau \in \{0.5, 0.6, ..., 0.9\}$ using SLIC superpixels [10] initialized at 1000 seeds. We then correlated the number of proto-objects formed after superpixel merging with the ground truth behavioral clutter perception estimates by computing the Spearman's Rank Correlation (Spearman's ρ) following the convention of [134][25][159][92].

A model using MLE as the optimization criterion achieved the highest correlation, $\rho = 0.8038$, $p < 0.001$ with $\epsilon = 0.14$ and $\tau = 0.8$. Because we did not have separate training/testing sets, we performed 10-fold cross-validation and obtained an average testing correlation of $r = 0.7599$, $p < 0.001$. When optimized using NLS, the model achieved a maximum correlation of $\rho = 0.7966$, $p < 0.001$ with $\epsilon = 0.14$ and $\tau = 0.4$, and the corresponding 10-fold cross-validation yielded an average testing correlation of $r = 0.7375$, $p < 0.001$. The high cross-validation averages indicate that our model is highly robust, and generalizable to unseen data.

It is worth pointing out that, the optimal value of the tolerance parameter ϵ showed a peak correlation at 0.14. To the extent that this is meaningful and extends to people, it suggests that visual clutter perception may ignore feature dissimilarity on the order of 14% when deciding whether two adjacent regions are similar and should be merged.

We compared our model to four other state-of-the-art models of clutter perception: the feature congestion model [134], the edge density method [95], the power-law model [26], and the C3 model [92]. Table 2.1 shows that our model significantly out-performed all of these previously reported methods. The relatively poor performance of the recent C3 model was surprising, and can probably be attributed to the previous evaluation of that model using charts and maps rather than arbitrary

WMM-m	**WMM-n**	MS[42]	GB[48]	PL[26]	ED[95]	FC[134]	# Obj	C3[92]
0.8038	**0.7966**	0.7262	0.6612	0.6439	0.6231	0.5337	0.5255	0.4810

Table 2.1. Correlations between human clutter perception and all the evaluated methods. WMM is the Weibull mixture model underlying our proto-object partitioning approach (-m means with MLE, -n means with NLS), with both optimization methods.

realistic scenes (personal communication with authors). Collectively, these results suggest that a model that merges superpixels into proto-objects best describes human clutter perception, and that the benefit of using a proto-object model for clutter prediction is not small; our model resulted in an improvement of at least 15% over existing models of clutter perception. Although we did not record run-time statistics on the other models, our model, implemented in Matlab, had an end-to-end (excluding superpixel pre-processing) run-time of 15-20 seconds using 800×600 images running on an Win7 Intel Core i-7 computer with 8 Gb RAM.

2.2.7.3 Comparison to image segmentation methods

We also attempted to compare our method to state of the art image segmentation algorithms such as gPb-ucm [14], but found that the method was unable to process our image dataset using either an Intel Core i-7 machine with 8 Gb RAM or an Intel Xeon machine with 16 Gb RAM, at the high image resolutions required by our behavioral clutter estimates. A similar limitation was found for image segmentation methods that utilizes gPb contour detection as pre-processing, such as [33][64], while [121][178] took 10 hours on a single image and did not converge.

Therefore, we limit our evaluation to mean-shift [42] and Graph-based method [48], as they are able to produce variable numbers of segments based on the unsupervised partitioning of the 90 images from our dataset. Despite using the best dataset parameter settings for these unsupervised methods, our method remains the highest correlated model with the clutter perception groundtruth as shown in Table 2.1, and that methods allowing quantification of proto-object set size (WMM, Mean-shift, and Graph-based) outperformed all of the previous clutter models .

We also correlated the number of objects segmented by humans (as provided in the SUN Dataset) with the clutter perception ground truth, denoted as # obj

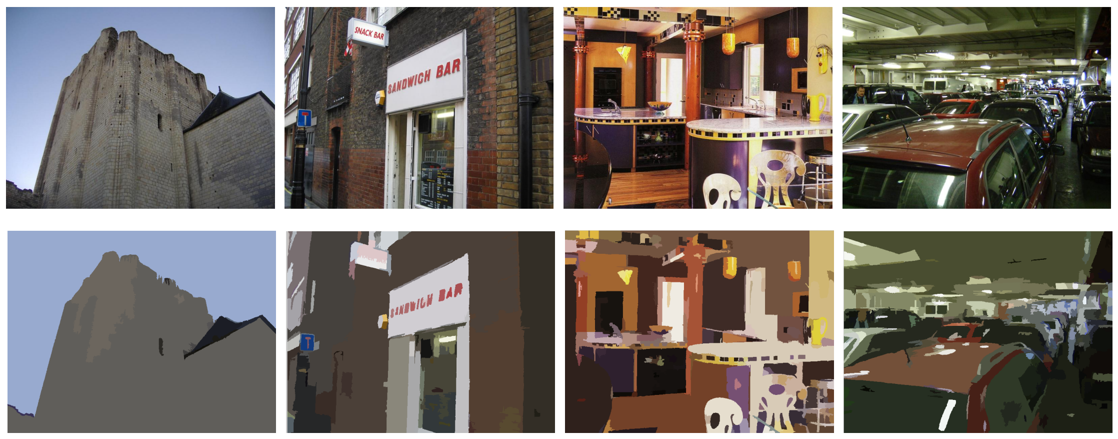


Figure 2.8. Top: Four images from our dataset, rank ordered for clutter perception by human raters, median clutter rank order from left to right: 6, 47, 70, 87. Bottom: Corresponding images after parametric proto-object partitioning, median clutter rank order from left to right: 7, 40, 81, 83.

in Table 2.1. Interestingly, despite object count being a human-derived estimate, it produced among the lowest correlations with clutter perception. This suggests that clutter perception is not determined by simply the number of objects in a scene; it is the proto-object composition of these objects that is important.

2.3 Mean-shift based proto-object segmentation

In addition to PGP as the superpixel clustering method, we also experiment with using Mean-shift clustering [42] for clustering superpixels, and evaluate its efficacy as a related but different proto-object segmentation method.

2.3.1 Superpixel clustering with mean-shift

To merge neighboring superpixels having similar features we perform a cluster analysis on the color feature space using Mean-shift. Three different color spaces are explored: RGB, HSV, and CIElab. In this respect, the mean-shift based approach is related to the C3 clutter model, which groups similar pixels by spatial proximity and color similarity if they fall under a threshold [92]. However, our work differs from this previous model by using Mean-shift to find the color clusters in a given image, then assigning superpixels to one of these clusters based on the median color over the image fragment in the color feature space. We then merge adjacent superpixels (ones that share a boundary) falling within the same color cluster into a larger region, thereby forming a proto-object.

We should note that the Mean-shift algorithm has itself been used as an image segmentation method [42], and indeed is one of the methods that we evaluate in our comparative analysis. Mean-shift clusters data into an optimal number of groups by iteratively shifting every data point to a common density mode, with a bandwidth parameter determining the search area for the shift directions; the data that converge to the same density mode are considered to belong to the same cluster. This clustering algorithm has been applied to image segmentation by finding a density mode for every image pixel, then assigning pixels that converge to a common mode to the same cluster, again based on spatial proximity and color similarity. Doing this for all common modes results in a segmentation of pixels into coherent regions. Our approach differs from this standard application of Mean-shift in that we use the algorithm, not for segmentation, but only for clustering. Specifically, Mean-shift is applied solely to the space of color medians in an image (i.e., using only the feature-space bandwidth parameter, and not both the feature-space and spatial bandwidth parameters as in the original formulation of the algorithm), where each median corresponds to a superpixel, and it returns the optimal number of color clusters in this space. Having clustered the data, we then perform the above described assignment of superpixels to clusters, followed by merging, outside of the Mean-shift segmentation method. By applying Mean-shift at the level of superpixels, and by using our own merging method, we will show that the mean-shift Proto-object model is also better predictor of human clutter perception than standard Mean-shift image segmentation.

Figure 2.9 illustrates the key stages of the Proto-object model of clutter perception, which can be summarized as follows:

1. Obtain superpixels for an image and find the median color for each. We will argue that our model is robust with respect to the specific superpixel segmentation method used, and will show that the best results were obtained with Entropy Rate superpixels ([91]) using k = 600 initial seeds.

2. Apply Mean-shift clustering to the color space defined by the superpixel medians to obtain the optimal number of color clusters in the feature space. We

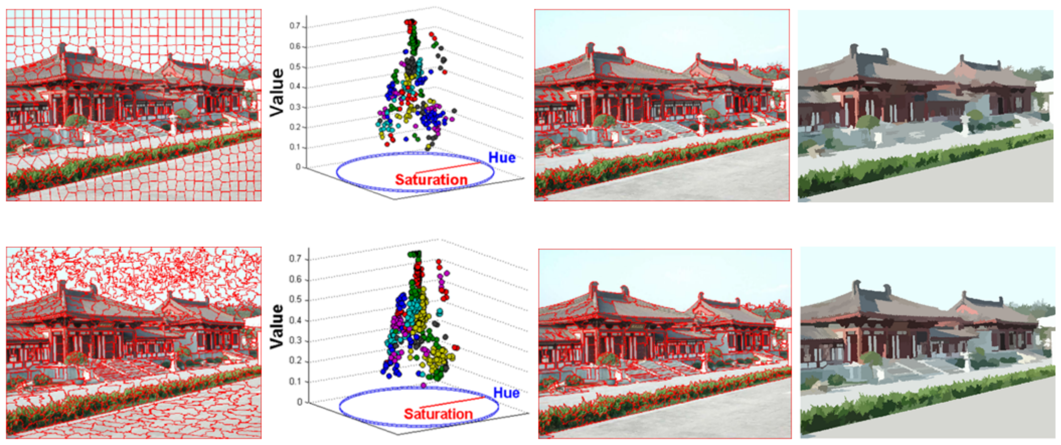


Figure 2.9. The computational procedure illustrated for a representative scene. Top row (left to right): a SLIC superpixel segmentation using k = 600 seeds; 51 clusters of median superpixel color using Mean-shift (bandwidth = 4) in HSV color space; 209 proto-objects obtained after merging, normalized visual clutter score = 0.345; a visualization of the proto-object segmentation showing each proto-object filled with the median color from the corresponding pixels in the original image. Bottom row (left to right): an Entropy Rate superpixel segmentation using k = 600 seeds; 47 clusters of median superpixel color using Mean-shift (bandwidth = 4) in HSV color space; 281 proto-objects obtained after merging, normalized visual clutter score = 0.468; a visualization of the proto-object segmentation showing each proto-object filled with the median color from the corresponding pixels in the original image.

will again argue that our model is robust with respect to the specific color space that is used, but that slightly better correlations with human clutter rankings were found using a bandwidth of 4 in an HSV color feature space.

3. Assign each superpixel to a color cluster based on the median color similarity, and merge adjacent superpixels falling into the same cluster to create a proto-object segmentation.

4. Normalize the proto-object quantification between 0 and 1 by dividing the final number of proto-objects computed for an image by the initial k number of superpixel seeds. Higher normalized values indicate more cluttered images.

2.3.2 Results and discussion

We computed a mean-shift proto-object segmentation for each of the 90 images (see Figure 2.10 for some examples), counted the number of proto-objects in each, normalized this count by dividing it by the number of initial superpixels, then rank

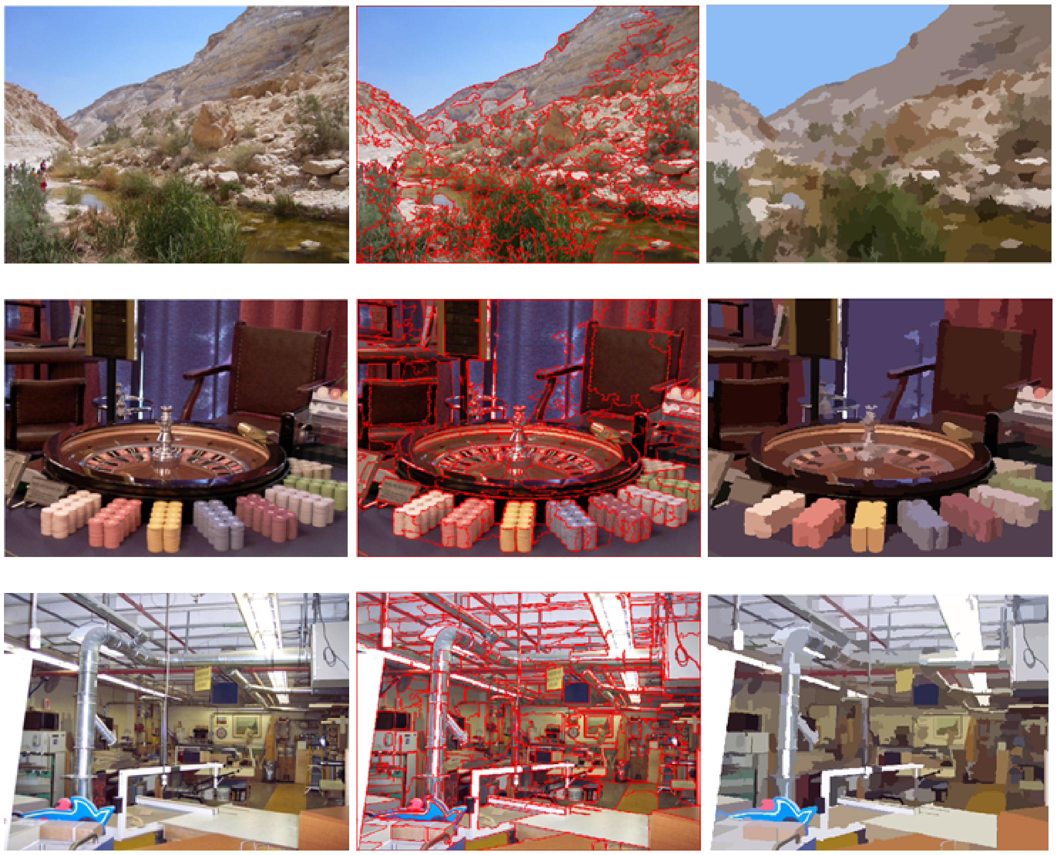


Figure 2.10. Representative examples of 3 of the 90 images used in this study (left column), shown with their corresponding proto-object segmentations (middle column) and reconstructions created by filling each proto-object with its median color (right column). Top row: clutter score = 0.430 (ranked 41st). Middle row: clutter score = 0.540 (ranked 65th). Bottom row: clutter score = 0.692 (ranked 87th). Corresponding rankings from the behavioral participants were: 38th, 73rd, and 89th, respectively. Proto-object model simulations were based on Entropy Rate superpixel segmentations ([91]) using 600 initial seeds and a Mean-shift clustering bandwidth of 4 within an HSV color feature space.

ordered these estimates of image clutter from least to most, paralleling the behavioral task. Correlating this ranking from the model with the median ranking from participants produced a Spearman's $\rho = .814$ ($p < .001$). This indicates that the mean-shift Proto-object model is also a very good predictor of how behavioral participants rank order scenes for visual clutter compares to the PGP-based method in the earlier section; the scenes that were ranked as least (most) cluttered by participants tended also to be the scenes that were ranked as least (most) cluttered

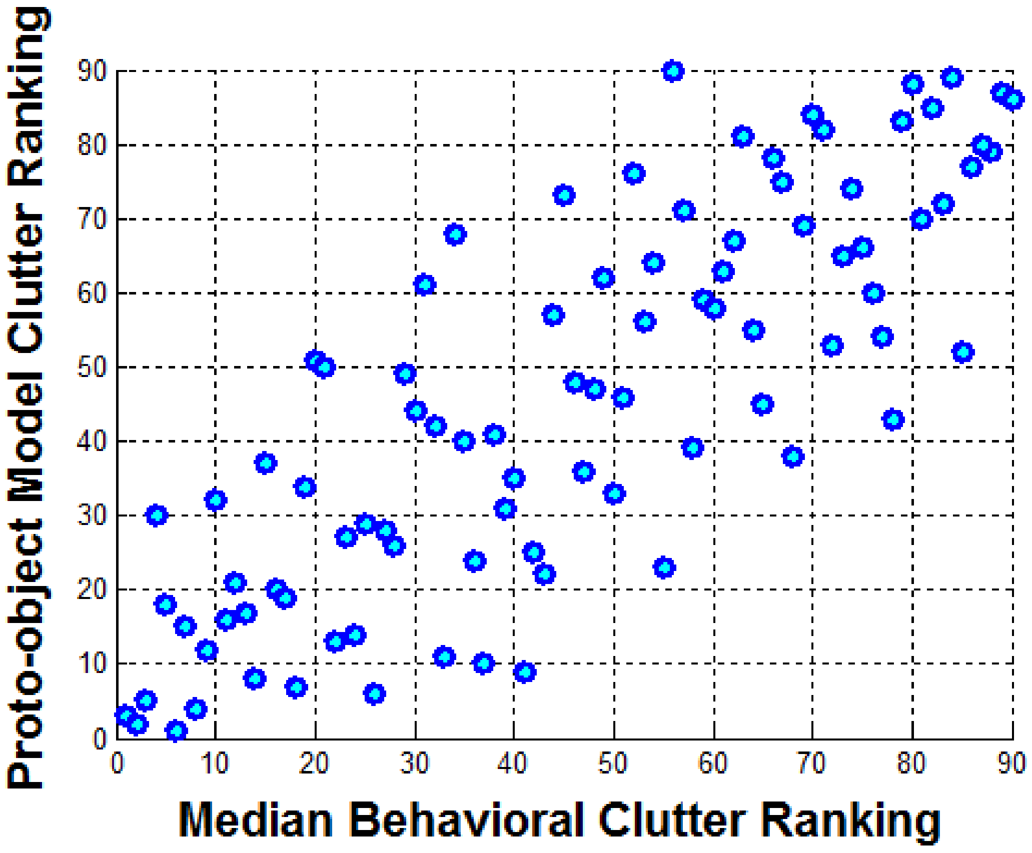
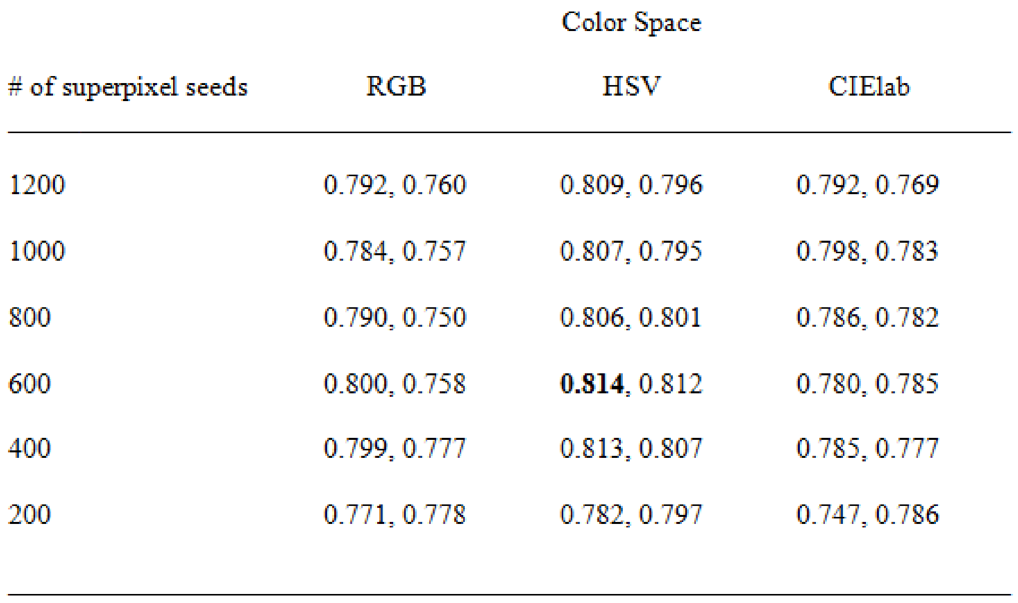


Figure 2.11. Clutter ranking of the 90 test scenes by the Proto-object model plotted as a function of the median clutter ranking by our 15 behavioral participants for the same 90 scenes. Spearman's $\rho = .814$.

by the model (Figure 2.11).

The high correlation between model and behavioral clutter rankings reported in some sense represents the best that the mean-shift Proto-object model could do over its parameter space, which includes the segmentation method used to obtain superpixels, the number of seeds used by this method, the color space used by the Mean-shift algorithm, and the bandwidth parameter that largely determines Mean-shift clustering. Specifically, the correlation of .814 was obtained using the Entropy Rate superpixel segmentation method [91], 600 initial seeds, an HSV color space, and a Mean-shift clustering bandwidth parameter of 4 in the feature space (settings used to produce Figures 2.10 and 2.11); would the mean-shift Proto-object model remain a good predictor of clutter perception if these parameters



	Color Space		
# of superpixel seeds	RGB	HSV	CIElab
1200	0.792, 0.760	0.809, 0.796	0.792, 0.769
1000	0.784, 0.757	0.807, 0.795	0.798, 0.783
800	0.790, 0.750	0.806, 0.801	0.786, 0.782
600	0.800, 0.758	**0.814**, 0.812	0.780, 0.785
400	0.799, 0.777	0.813, 0.807	0.785, 0.777
200	0.771, 0.778	0.782, 0.797	0.747, 0.786

Figure 2.12. Spearman's correlations between the mean-shift Proto-object model rankings and behavioral rankings as a function of color space, superpixel segmentation method, and the number of initial superpixel seeds. Leftmost correlations were obtained using Entropy Rate superpixel segmentation; rightmost correlations were obtained using SLIC superpixel segmentation. All correlations used an optimized Mean-shift spatial bandwidth parameter. The highest correlation is indicated in bold.

were changed?

Figure 2.12 answers this question with respect to the choice of segmentation method, number of seeds, and color space. Reported is a 236 matrix of correlations obtained by crossing 2 segmentation methods (Entropy Rate, [91], and SLIC, [10]), 3 color spaces (RGB, HSV, and CIElab), and 6 seed initializations (k = 200, 400, 600, 800, 1000, 1200). An optimal Mean-shift clustering bandwidth was computed separately for each of these 36 simulations. The largest difference in correlation between any two cells of this matrix was only .067, contrasting Entropy Rate superpixels using an HSV color space and 600 seeds with Entropy Rate superpixels using a CIElab color space and 200 seeds. Within any given dimension, these maximum differences were .067, .051, and .056 for segmentation method, color space, and seed number, respectively. The clear take-away message from this exploration is that, although varying segmentation method, color space, and seed initialization did affect the models predictive success, our Proto-object model was exceptionally robust to changes in these parameter values and performed extremely well across its parameter space. At its worst, the mean-shift Proto-object model still produced a respectable .747 correlation with the behavioral clutter rankings.

These results also suggest that a proto-object model that is based on clustering superpixels outperform all other types of models, as PGP-based Proto-object segmentation resulted in a correlation of 0.8038, and the mean-shift based Proto-object method resulted in a correlation of 0.814. These experiments provided further evidence that proto-objects are important to the perception of visual clutter, and a model that incorporates the idea of proto-objects is able to best predict visual clutter perception.

2.4 General discussion

To be sure, our visual world consists of features and objects, but there is also something in betweena level of visual representation consisting of proto-objects. Proto-objects are the fragments of our perception, and as such likely mediate many if not all of our everyday percepts and behaviors. In this study we focused on the perception of visual clutter. Clutter perception is interesting in that it colors all of our visual experience; we seem to know without thinking whether a scene is cluttered or not, and this knowledge impacts our thoughts, our actions, and even our emotions. This largely automatic perception of clutter points to a fairly low-level process operating in the background of our consciousness, one that does not require extensive training, expectation, or experience with what is, or is not, a cluttered scene.

We modeled this process by performing clustering methods to obtain an unsupervised segmentation of a scene into superixels, then merged these superpixels based on their low-level feature similarities to obtain what we refer to here as proto-objects, spatially extended regions of coherent features. This Proto-object model estimates clutter perception as a simple count of the number of proto-objects extracted from an image, with a larger number predicting a more cluttered percept. We tested this model against a relatively large set of realistic scenes that were behaviorally ranked for visual clutter and found that it was highly successful in predicting this clutter ranking. We showed that this model was generalizable to unseen images and highly robust to changes in its parameters. It also outperformed, in some cases dramatically, all existing models of clutter, making it the new standard against which future models of clutter perception should be compared.

Future work will apply the Proto-object model of clutter to a visual search task. The authors of previous clutter models have done this with the hope of using clutter estimates as a surrogate measure of the number of objects in a scene. Underlying this motivation is the assumption that if one were to actually know the number of objects in a scene that this object count would predict how cluttered that scene would appear (but see [134]). We directly tested this assumption and found that, while a measure of object count did predict clutter, it did so far less successfully than did the Proto-object model, and indeed any model that used superpixel segmentation as a pre-process. This raises the possibility that the number of proto-objects, and not the number of actual objects, might be a better surrogate measure of search set size effects in realistic scenes. In addition to exploring further this possibility we will also ask how the distribution of fixations over a scene might be predicted by the distribution of proto-objects, would targets be found more quickly if they were embedded in regions of high proto-object density? This might be the case if the fixations made during a search or a free viewing task were biased to proto-object clusters.

In future work we will also seek to extend the Proto-object model by considering features other than just color when merging superpixels into proto-objects. The features used in proto-object formation are still largely unknown, and indeed the concept of what a proto-object is, and is not, is currently evolving. The term proto-object, as originally formulated by [123], relied heavily on a simple grouping of visual features into small clusters, each having a very limited, approximately two degrees of visual angle, spatial extent. Although our proto-objects are not spatially constrained in the same way, and are able to grow and merge with neighboring image patches depending on their preattentive feature similarity, this early definition seems aligned most closely with the conception of proto-objects used in our model.5 However, more recent usages of the term proto-object ([125]) have assumed the additional contribution of 3-D features to obtain local estimates of scene structure, making these more complex proto-objects very different from the simpler entities proposed in the present study. Starting with basic features (e.g., intensity, orientation, texture) and working up to more complex, we will explore how the systematic addition of features to the Proto-object model affects its performance. This effort will give us a richer understanding of the mid-level vision features, and visual statistics computed from these features, that are useful for predicting clutter perception and potentially related visual behaviors [50].

Underlying these efforts is the belief that our perception of the world is a piecemeal construction; it starts with pixels and features, but these quickly become merged into locally (superpixel) and globally (proto-object) coherent regions, and, eventually, object parts and objects. The fact that superpixel-based approaches were found to outperform feature-based approaches in this study is telling, and speaks to the potential importance of this intermediate proto-object level of visual representation. The theoretical implications of this observation are profound. It may be the case that the gap between pixels and objects is just too great, and that robust computational methods for understanding object perception and detection can only emerge by considering the intermediate fragments of our perception.

2.5 Application of PGP to video segmentation

PGP was demonstrated to be successful at segmenting proto-objects from images, for clutter perception modeling. In a different application here, we show that PGP can also be applied to the problem of video segmentation, which is to group pixels into contiguous spatio-temporal segments. Video segmentation methods in general attempt to cluster similar pixels together under a spatio-temporal setting, either by methods that generate a set of hierarchical segmentations (from very detailed to more coarse) [57][43][175], or by methods that group superpixels together to form spatio-temporal superpixels or supervoxels [35][87][162]. The latter methods are significantly more efficient computationally, albeit less accurate. In this section, we propose a superpixel grouping method that improves the state-of-the-art by as much as 30%, and is approximately 20 times faster.

Hierarchical video segmentation methods have demonstrated excellent 3D (spatiotemporal) segmentation results on standard datasets such as Segtrack [155] and Chen's Xiph.org [36] using a variety of metrics [173]. However, their applicability in video processing pipelines remains limited, due to computational complexity

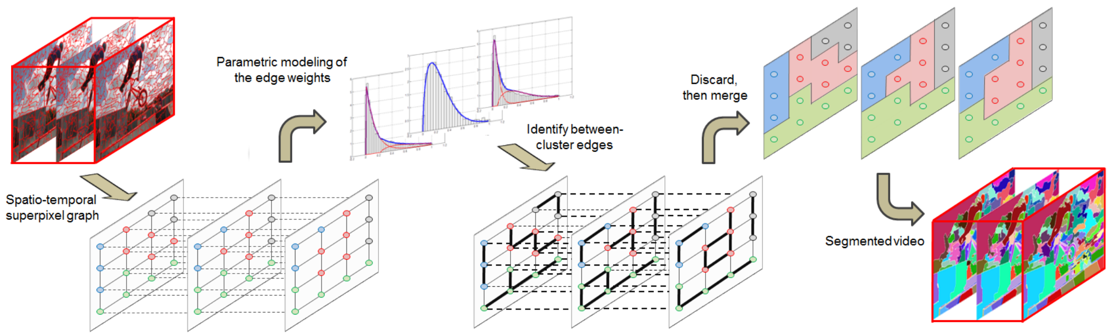

Parametric modeling of the edge weights
Identify between-cluster edges
Discard, then merge
Spatio-temporal superpixel graph
Segmented video

Figure 2.13. The overview of video segmentation pipeline using PGP. First, a spatio-temporal superpixel graph is constructed from the video (dashed lines are temporal connections). Then, the edge weights are modeled by a mixture of Weibull distributions, computed separately for individual features. The mixture models generate information about the inter- and intra-cluster edges (intra-cluster edges in bold). The inter-cluster edges are discarded and each of the remaining groups of superpixels becomes a final video segment.

and the difficulty of automatically selecting the appropriate hierarchical layer for particular applications. As an attempt to address these obstacles, the recently proposed Uniform Entropy Slice (UES) method ([174]) selects different supervoxels from the several hierarchical layers to form a single output segmentation, by balancing the amount of feature information of the selected supervoxels [174]. UES builds on top of hierarchical segmentation pre-processing techniques such as [57] and [43] to produce a single segmentation that is more practical for further use. This comes at the cost of increased computation and decreased performance in 3D quantitative performance metrics compared to [57] and [43]. To obviate the use of expensive hierarchical segmentation as pre-processing, we apply the PGP pipeline on the spatio-temporal superpixel graph, and Figure 2.13 gives an overview of our method.

To recap, PGP is a graph partitioning method that performs clustering without needing the user to specify the number of clusters, or search-window parameters as in mean-shift [42]. The method optimizes a number of two-component parametric mixture models over the edge weights of a graph (one model per feature type). The edges are bi-partitioned into a within-cluster and a between-cluster group by performing inference on these mixture models. Thus the between-cluster edges of the graph can be identified and removed, in order to create a variable amount of isolated clusters in an efficient manner. Each of these clusters corresponds to a 3D video segment. It is safe to assume that L_p-norm based similarity distance statistics in general are Weibull distributed [29]. Based on this observation, the previous section produced high quality single image partitioning results and segmented contiguous image regions well. Using PGP has a number of advantages:

- PGP is computationally efficient; its run time is linear in the number of superpixels. This is especially suitable for processing big data such as video.
- PGP is a one-pass algorithm that produces a non-hierarchical result, which eliminates the need to select the appropriate hierarchical layer.

2.5.1 Spatio-temporal superpixel graph

Given a graph $G = (V, E)$, an edge $e_{u,v} \in E$ connects two neighboring nodes (superpixels) $u, v \in V$. Let x_i be the weight of the i^{th} edge e_i of the graph, the task is to assign a binary label to e_i by an indicator function $y_i = I(x_i)$, such that $y_i = 1$ if e_i is an intra-cluster edge that should be retained, or $y_i = 0$ if e_i is a inter-cluster edge that should be removed from the graph. For a given feature $f \in \mathbf{F}$, x_i is the similarity distance between the feature histograms of nodes (superpixels) v_a and v_b connected by the i^{th} edge e_i such that $x_i = D(v_a, v_b|f)$, and we denote $\mathbf{x}$ as similarity distances of different features.

Neighboring nodes in a spatio-temporal graph are defined as nodes that are spatially or temporally adjacent to one another, where temporal adjacency in our framework is defined differently depending on whether the motion feature (described in Section 3.2) is used: if the motion feature is used, the temporal neighbors of v_a are nodes located within a $n \times n$ window on the next temporal frame, where the center of the window is specified by the mean motion vector of v_a; if the motion feature is not used, then temporal adjacency is defined by a $4n \times 4n$ window directly on the next temporal frame using the centroid of v_a as the center of the window.

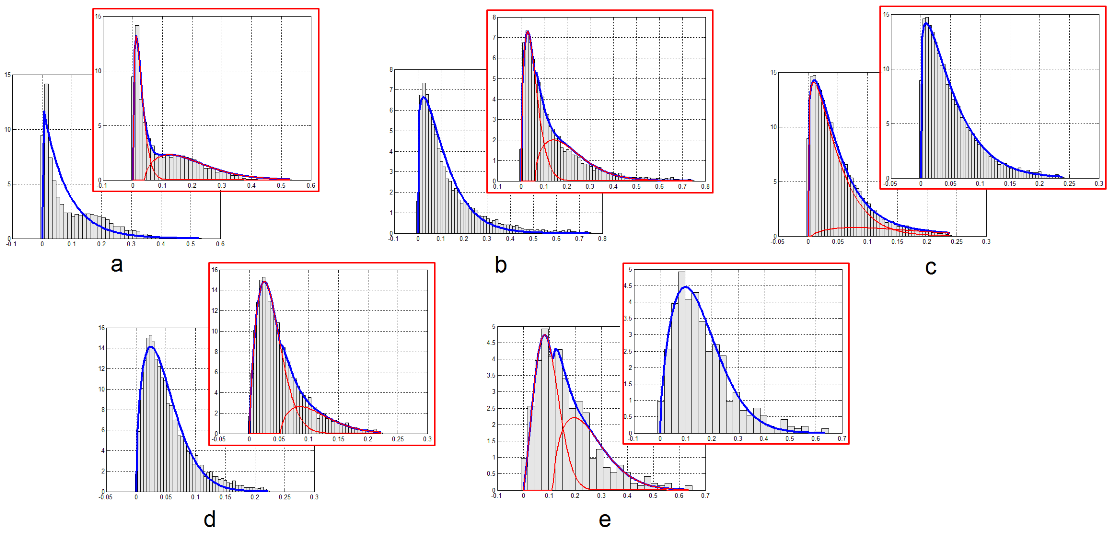


Figure 2.14. The nonlinear least-squares fits of Weibull Mixture Models (one and two components) on the Earth Mover's Distance statistics of the five features with model selection done for the BMX-1 video. The blue lines are the posterior probability and the red lines are the probability of individual mixture components. The models in the red boxes are the selected ones by AIC. a: intensity, b: hue, c: ab, d: motion, and e: gradient orientation.

2.5.2 Feature combination

After obtaining the spatio-temporal graph, the WMM fitting method (see Section 2.2.3, 2.2.4) is performed to obtain the cross point γ of the mixtures, and Figure 2.14 shows an example set of results of the fitted WMMs, displaying excellent fits and remarkable Weibull shapes. we combine the n features to compute the label for x_i as the weighted-sum over the scaled $\mathbf{x}$:

$$y_i = I(\sum_{k=1}^{n} w_i \sigma(x_i, \gamma) < 0 \mid f_k) \tag{2.8}$$

where $\sigma(x_i, \gamma)$ is a scaling function that linearly scales $[\min(\mathbf{x}), \gamma]$ to $[-1, 0]$, and $[\gamma, \max(\mathbf{x})]$ to $[0, 1]$. $\sigma(x_i, \gamma)$ is a piecewise function with γ being the breakpoint:

$$\sigma(x_i, \gamma) = \begin{cases} (x_i - \gamma)/(\gamma - \min(\mathbf{x})) & \text{if } \; x_i < \gamma \\ \\ (x_i - \gamma)/(\max(\mathbf{x}) - \gamma) & \text{if } \; x_i \geq \gamma \end{cases} \tag{2.9}$$

Notice that when there is only one feature being considered, Eqs. 2.8 and 2.9 combine into a simple threshold function and γ becomes a threshold that partitions the graph such that $y_i = I(x_i >= \gamma)$.

When multiple features are considered, we expect that in different parts of the image, different features will be most prominent. For instance if two neighboring superpixels both undergo significant motions, their mean motion feature value will be higher than most other superpixel pairs, indicating that motion similarity is of higher importance when combined with the other features in Eq. 2.8. Hence, for each such superpixel pair, we want to promote the weight w_i of the most prominent feature by computing w_i adaptively. This adaptive scheme comes from the intuition that the importance of a feature depends on how much of it is present. w_i is the mean feature value of v_a and v_b (connected by e_i), normalized by the maximum feature value for the entire video: $w_i = avg((f_k|v_a), (f_k|v_b))/\max(f_k)$. Specifically, the saturation value is used to measure how much color is present, while the intensity feature weight is 1 minus the weight of the color feature.

As a final step before labeling the edges, we compute the minimum spanning tree (MST) of the fully connected graph G before making the cuts. This step was shown in the Section 2.2.5 to reduce the under-segmentation of the graph by removing cycles and retaining only the most essential edges of the graph. The MST is computed over the product of the edge weights for the n features, further multiplied by the distance $\mathbf{d}$ between the centroids of the neighboring superpixels, to ensure that closer neighbors are more likely to be under the same cluster. E' is the list of edges from the MST:

$$E' = MST(\mathbf{d} \prod_{k=1}^{n} \mathbf{x}_k \mid f_k) \tag{2.10}$$

Edge labeling is performed over E' according to Eq. 2.8.

2.5.3 Branch reductions

After removing the between-cluster edges that are identified by PGP, we obtain a set of disjoint superpixel clusters which are merged into separate video segments. However, a single frame slice of a spatio-temporal segment may result in several non-contiguous regions (branches) on that single frame. Although the branches

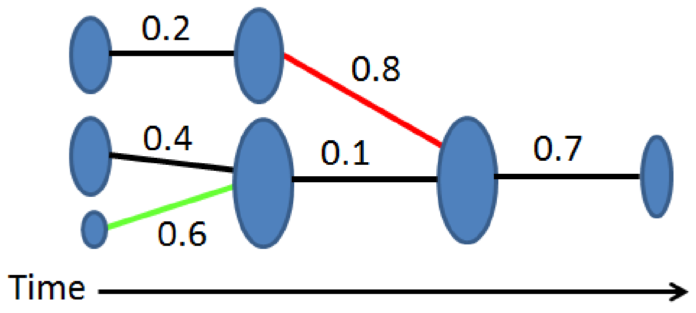


Figure 2.15. Example process of our greedy branch reduction method. Given the nodes (representing superpixels), the red edge is removed first, followed by the green edge.

may be the result of minor occlusion, they are undesirable in most cases and care should be taken to address this issue [115]. Therefore, we separate the branches of the spatio-temporal segments by post-processing using a greedy algorithm: we iterate through every spatio-temporal segment and check if it produces non-contiguous regions in any single video frame. If found, the algorithm picks the two largest regions on that frame and removes the edge with the highest weight along the path between the two regions. This step is repeated until all branches of the given spatio-temporal segment are removed, or a pre-defined number of iterations is reached . Figure 2.15 illustrates an example of this process.

2.5.4 Experiments and results

We evaluate the PGP video segmentation application quantitatively using two video datasets: Segtrack V2 [87] and Chen's Xiph.org [36]. The recently proposed Segtrack V2 is an extended version of the Segtrack V1 dataset [155], where the number of videos increased from 6 to 14. Videos vary in length and each has multiple objects and pixel-level ground-truth. Chen's dataset is a 8-video subset of the Xiph.org videos that contains semantic pixel-level labeling of multiple classes as ground-truth.

The features **f** that we use in this work are i) intensity (256-bin 1D histogram), ii) hue of the HSV colorspace (77×77 2D histogram), iii) the color component (AB) of the LAB colorspace (20×20 2D histogram), iv) motion optic flow (50×50 2D histogram), and v) gradient orientation (360-bin 1D histogram). Earth Mover's

Distance (EMD) is used for all features. We generate superpixels using [91]. For the temporal neighbors' $n \times n$ search window size, we empirically set n to be 2.5% of the (video width + video height)/2; and cap the branch reduction iteration at 100 per spatio-temporal segment.

2.5.5 Feature properties

In order for the L_p distance statistics to follow a Weibull distribution, the compared feature differences must be correlated but non-identically distributed random variables. We again follow [29] to test the 5 features used in this paper.

In summary, for each feature, we take its feature histogram from a randomly selected reference superpixel s and 100 other randomly selected superpixels T, and compute the difference $\Delta_i = |s_i - T_i|^p$, at all histogram bins i. Then, we compute the Wilcoxon rank sum test between the difference values Δ_i and Δ_j of all possible pairs of indices, $i \neq j$. We set the confidence level at 0.05, and re-sample s and T 500 times per video, for all 5 features. As a result, we found that over 98% of the feature differences for all 5 features for both video datasets are non-identically distributed. This procedure is repeated for testing whether the feature differences are correlated, the second required assumption for satisfying the Weibull distribution property of L_p-norm based distance statistics. In this case we used Pearson's correlation, and we again found that over 98% of all feature differences for all 5 features are significantly correlated. This is not surprising as [29] showed that even for hand-designed-features such as SIFT, SPIN, and GLOH over 85% of the feature differences satisfy both conditions.

2.5.6 Quantitative evaluations

In the following, we compare our methods with the state-of-the-art related method on two datasets, and discuss the effects of the motion feature, number of superpixels extracted in pre-processing, sub-volume processing, the percentile parameter τ, and provide run time analysis.

Method Comparisons. Table 2.2 and 2.3 shows our quantitative results using the metrics proposed by [173] and compared to the Uniform Entropy Slice (UES) method [174]. Like our method, UES also aims to produce just one single

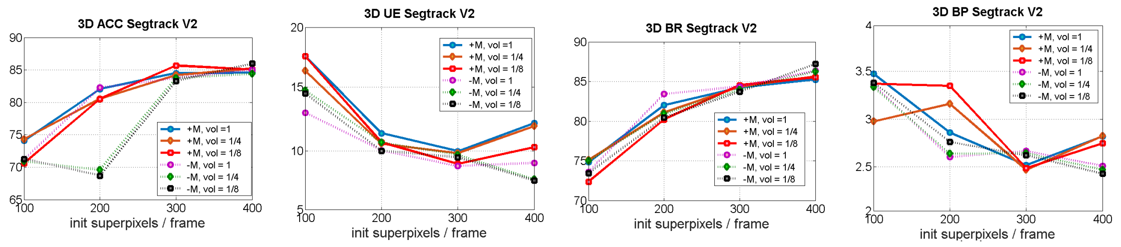


Figure 2.16. The plots that show the effects of varying different conditions (initial number of superpixels, size of the sub-volume processing, and with or without motion), for Segtrack V2 dataset: +/-M means with or without motion feature, vol stands for the sub-volume's size as a portion of the original video. These plots suggest that there is no clear indication of the benefit in using the motion feature (dotted lines) within the PGP framework, and that sub-volume processing performs approximately the same as processing the full volume.

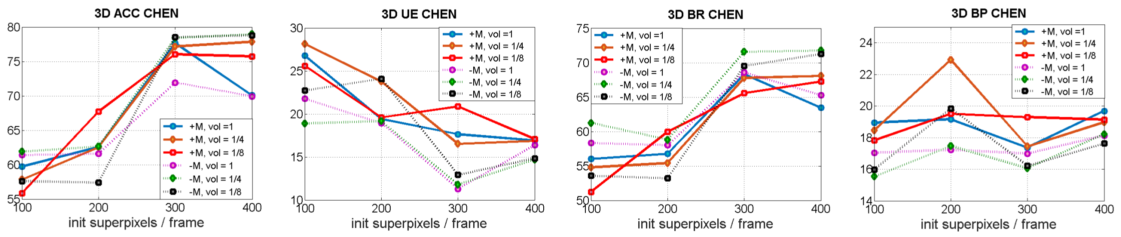


Figure 2.17. The plots show the effects of varying different conditions for Chen's dataset. Sub-volume processing seems to slightly outperform full video mode, and better performance tends to be associated with the exclusion of motion feature.

segmentation output, by automatically selecting and combining the appropriate supervoxels from the multiple layers of segmentations using SWA [43] or GBH [57]. We test several variants of our method: MLE+ for MLE optimized results, NLS+ and NLS- for optimization done using NLS, where +/- refers to the inclusion of motion feature or not, respectively. Due to space constraints, we report only one variant of MLE, since in our experiments MLE performed slightly but consistently worse than NLS optimization. The table shows that all the variants of our method outperform considerably both UES variants in all categories other than 3D BP in Chen's dataset. Their higher 3D BP is likely due to the significant overall under-segmentation of UES-GBH which heavily raises precision values.

Motion feature. When the motion feature is used, our algorithm uses the optical flow vectors for a more refined search of the temporal neighbors. Similar to the other features, we model a WMM over the similarity distance statistics based

	SegTrack V2																			
	3D Accuracy (AC)					3D Under-segmentation Error (UE)					3D Boundary Recall (BR)					3D Boundary Precision (BP)				
Video-obj	MLE+	NLS-	NLS+	[174]a	[174]b	MLE+	NLS-	NLS+	[174]a	[174]b	MLE+	NLS-	NLS+	[174]a	[174]b	MLE+	NLS-	NLS+	[174]a	[174]b
Bird of P.	93.2	96.3	93.5	**97.6**	95.3	3.1	2.9	3.0	**1.6**	2.6	85.4	91.7	87.8	**94.8**	91.6	10	6.7	**8.2**	6.9	5.8
Birdfall	67.3	**71.3**	70.7	65.3	53.8	81.5	50.9	44.4	28.8	**23.2**	83.6	84.7	83.8	**88.5**	82.1	**1.5**	0.8	1.1	0.8	0.9
BMX-1	**95.6**	95.0	**95.6**	90.3	65.6	7.1	7.1	6.9	**6.3**	7.7	97.3	97.4	97.5	**97.7**	93.6	4.7	4.2	4.5	**5.1**	3.6
BMX-2	78.2	78.9	**79.0**	44.3	27.1	9.4	**8.4**	10.0	11.7	16.8	90.6	91.5	91.0	**92.6**	88.1	4.3	3.9	4.1	**4.7**	3.3
Cheetah-1	72.8	**74.3**	73.9	0	39.4	9.0	**6.5**	9.5	47.4	34.1	93.4	**97.2**	94.7	63.8	75.3	1.1	1.1	1.1	1.4	**1.6**
Cheetah-2	69.9	**70.1**	69.7	0	12.0	8.7	**6.9**	8.9	54.5	34.4	98.4	**98.5**	98.4	76.8	75.3	1.5	1.4	1.5	**2.2**	2.0
Drift-1	92.4	92.2	**92.6**	83.0	75.6	6.7	**3.9**	6.0	3.5	7.1	92.9	93.6	93.5	**94.9**	90.6	**1.2**	1.0	1.1	1.0	**1.2**
Drift-2	91.9	**93.2**	92.8	84.9	56.8	7.5	**4.1**	7.3	4.2	10.0	91.3	**92.7**	91.6	87.6	82.9	**0.9**	0.8	**0.9**	0.7	0.8
Frog	14.3	33.5	25.6	n/a	**62.4**	16.5	15.8	15.7	n/a	**13.1**	29.5	55.1	44.6	n/a	**81.4**	**13.2**	8.7	7.1	n/a	1.7
Girl	87.2	**88.4**	87.9	65.5	63.5	9.9	10.3	**9.3**	12.3	13.5	90.3	91.7	**92.0**	75.6	83.2	4.3	**5.2**	4.1	**5.2**	4.6
Hbird-1	64.5	64.1	**69.5**	55.9	0	9.1	20.1	**8.8**	7.8	14.5	76.3	80.7	**86.8**	79.8	35.0	2.8	5.8	3.2	**6.3**	3.3
Hbird-2	78.4	67.5	**81.9**	70.6	0	**7.8**	11.2	7.9	8.0	13.4	90.4	83.3	**95.5**	92.7	86.0	5.0	9.0	5.3	**11**	12
Monkey	88.8	87.5	**90.7**	87.5	0	6.7	**3.5**	5.7	11.7	19.6	91.8	**94.3**	93.6	92.0	64.0	1.7	1.5	1.6	1.7	**3.4**
Mdog-1	86.7	**88.2**	88.1	41.7	79.9	12.6	**11.0**	11.6	41.4	43.2	94.8	**95.7**	**95.7**	94.3	91.0	1.6	1.5	1.5	2.1	**3.0**
Mdog-2	56.9	65.0	**66.7**	43.2	0	8.0	8.1	**6.5**	27.9	22.0	84.5	**88.5**	**90.0**	78.5	44.0	1.0	1.0	1.0	**1.3**	1.0
Parachute	**93.5**	93.4	93.4	90.9	89.3	22.0	**7.2**	19.9	18.5	38.6	94.9	**97.3**	96.0	95.7	87.4	1.3	0.7	1.1	**1.5**	10
Penguin-1	96.2	**96.4**	94.6	94.8	85.0	3.4	3.3	3.3	2.2	**1.8**	49.3	48.8	49.3	**77.3**	65.5	1.1	0.9	1.1	**1.4**	0.9
Penguin-2	96.4	96.5	**96.6**	85.1	93.1	4.7	4.6	4.7	3.3	**2.1**	69.6	**74.9**	71.7	66.8	75.3	**1.6**	1.5	**1.6**	1.3	1.1
Penguin-3	**96.2**	95.6	96.1	87.6	83.7	4.0	3.8	4.0	2.9	**2.5**	70.7	**74.2**	71.6	54.2	72.7	**1.6**	1.4	**1.6**	1.0	1.0
Penguin-4	96.0	95.8	**96.1**	83.8	82.8	3.9	3.9	3.8	**2.0**	2.4	73.4	**75.4**	74.0	45.7	56.7	**1.4**	1.2	**1.4**	0.7	0.7
Penguin-5	87.9	**89.3**	**89.3**	81.8	72.3	8.2	9.7	8.1	4.6	**4.1**	72.9	**76.0**	74.9	59.2	54.0	**1.2**	1.1	**1.2**	0.8	0.6
Penguin-6	96.7	**98.8**	97.0	85.7	86.8	4.1	4.1	4.0	**2.2**	2.5	61.1	47.1	66.2	**66.4**	63.3	1.2	0.8	**1.3**	1.1	0.8
Soldier	88.5	87.1	**89.5**	65.3	67.2	10.0	**4.9**	**4.9**	8.3	10.8	91.6	**94.4**	94.3	88.4	86.3	**3.2**	1.9	2.2	1.9	2.5
Worm	90.9	**92.4**	91.2	n/a	0	23.0	**20.3**	21.3	n/a	32.8	87.1	**90.3**	90.0	n/a	69.2	**1.9**	1.5	1.6	n/a	1.8
Average	82.5	83.8	**84.2**	68.4	53.8	12.0	**9.7**	9.8	14.2	15.5	81.7	84.0	**84.4**	80.2	74.8	**2.9**	2.6	2.5	2.7	2.8
Median	88.7	88.9	**89.6**	82.4	64.6	8.1	**7.0**	7.6	7.9	13.3	88.7	**90.9**	90.5	83.7	78.4	1.6	1.5	1.5	1.5	**1.8**

Table 2.2. Quantitative evaluation on the SegTrack v2 dataset, [174]a is UES with SWA, and [174]b is UES with GBH. For the 3D UE metric, the lower the error the better. For all the other metrics, the higher the score, the better. Best values are shown in **bold**. For the variants of our methods, the letters stand for the optimization method used, and +/- indicates the use of motion feature. All of the reported variants were based on 300 initial superpixels per frame at a 1/4 sub-volume processing mode. This table shows that our method's averages outperformed [174] in all 4 metrics, while our medians came out on top in 3. The n/a entry indicates that the method failed to converge to a result, the cases here were due to memory overload.

	Chen Xiph.org																			
	3D Accuracy (AC)					3D Under-segmentation Error (UE)					3D Boundary Recall (BR)					3D Boundary Precision (BP)				
Video	MLE+	NLS-	NLS+	[174]a	[174]b	MLE+	NLS-	NLS+	[174]a	[174]b	MLE+	NLS-	NLS+	[174]a	[174]b	MLE+	NLS-	NLS+	[174]a	[174]b
Bus	61.7	68.2	**69.3**	55.5	8.1	37.6	**6.8**	37.0	33.2	656	73.8	80.7	74.6	**84.5**	29.1	**41.5**	38.4	39.2	35.5	55.5
Container	82.2	89.8	**90.2**	89.2	77.3	11.6	3.6	7.5	**1.8**	4.5	58.6	**71.8**	69.4	64.7	51.5	**14.9**	8.3	**14.9**	9.9	11.6
Garden	82.7	**83.8**	83.3	**83.8**	63.1	1.8	1.9	1.8	**1.6**	3.3	69.9	73.6	70.6	**76.1**	40.0	13.9	13.4	13.9	12.1	**25.3**
Ice	**89.6**	87.6	87.5	79.4	46.6	29.3	26.0	27.1	**16.6**	69.1	83.3	**84.3**	83.0	80.6	50.3	33.8	31.4	32.4	36.1	**43.8**
Paris	49.5	**52.0**	50.9	47.6	2.0	14.7	**12.4**	14.3	19.2	40.2	46.4	**53.0**	50.8	44.8	34.1	4.7	4.4	4.6	3.8	**5.3**
Salesman	71.1	**82.2**	72.5	64.6	0	4.0	4.4	4.1	**3.4**	12.7	38.0	**51.0**	38.0	40.0	1.0	6.4	**7.1**	6.6	5.7	0.7
Soccer	72.9	**79.4**	78.3	70.9	26.4	46.6	32.7	34.0	**17.0**	145	73.3	**76.1**	74.9	75.2	58.8	16.7	14.4	16.3	16.3	**29.2**
Stefan	74.5	84.3	**85.4**	65.5	64.3	8.2	6.6	**6.3**	19.0	13.4	62.7	**82.2**	81.0	72.1	63.9	12.7	11.1	11.5	13.2	**13.4**
Avgerage	73.0	**78.4**	77.2	69.6	36.0	19.2	**11.8**	16.5	14.0	107	63.2	**71.6**	67.8	67.2	41.1	18.1	16.1	17.4	16.6	**23.1**
Median	73.7	**83.0**	80.1	68,2	36.5	13.2	**6.7**	10.9	16.8	26.8	66.3	**74.8**	72.6	73.7	45.2	14.4	12.3	14.4	12.7	**19.4**

Table 2.3. Quantitative evaluation on the Chen Xiph.org dataset. All of the settings are exactly the same as Table 1. For this dataset, both our average and median values outperformed [174] in 3 out of 4 metrics.

on motion feature histograms, and find the critical point γ that indicates the point of dissimilarity, which defines the inter-cluster values. In particular, motion feature similarity is considered only by spatial neighbors. The temporal neighbors just use motion vectors to specify the location of the search window. If the motion feature is not used, we search for temporal neighbors within a pre-defined window. We extract the motion feature using [149], but any optical flow extraction algorithm can be used. As both tables 1 and 2, and Figures 2.16 and 2.17 show, the results of using motion features within our framework are mixed, as motion seems to have improved the results for the SegTrack v2 dataset but not for Chen's dataset. Furthermore, not using motion lowers run time because optical flow extraction methods can be time consuming. Furthermore, Figure 2.18 shows that example results with (2nd row) and without (3rd row) motion are qualitatively similar.

Initial superpixel resolution. While our method produces a single segmentation result, the effects of varying the initial superpixel resolution are worth investigating. We evaluate our method with 100, 200, 300, and 400 initial superpixels per frame. Figures 2.16 and 2.17 show that a low number of superpixels tends to cause under-segmented results, most likely because the initial segmentation is less precise. The plots in Figures 2.16 and 2.17 also indicate that, after a point, increasing the superpixel resolution does not further improve results. Our method is robust to different initial superpixel resolutions as long as the number of superpixels is enough to produce a good initial over-segmentation (i.e. 300 per frame for the tested datasets). Figure 2.18 shows examples of the results that started with 300 (2nd and 3rd row) and 100 (4th row, somewhat under-segmented result) initial superpixels per frame.

Sub-volume processing. So far, we have been describing how our method processes the edge weights from the entire video. However, it can also be used to fit WMMs over sub-volumes and find the 'local critical values' that best describe the feature similarities at certain shots. The streaming GBH method [175] processes videos in chunks for efficiency, but loses information when optimizing only a group of frames at a time. In contrast, our PGP framework benefits from making the right divisions into sub-volumes for the WMM to optimize locally, as the feature similarities are more specific within a shot boundary. An example would be a change in activity: a triathlete is bike riding for the first 10 seconds of the video,

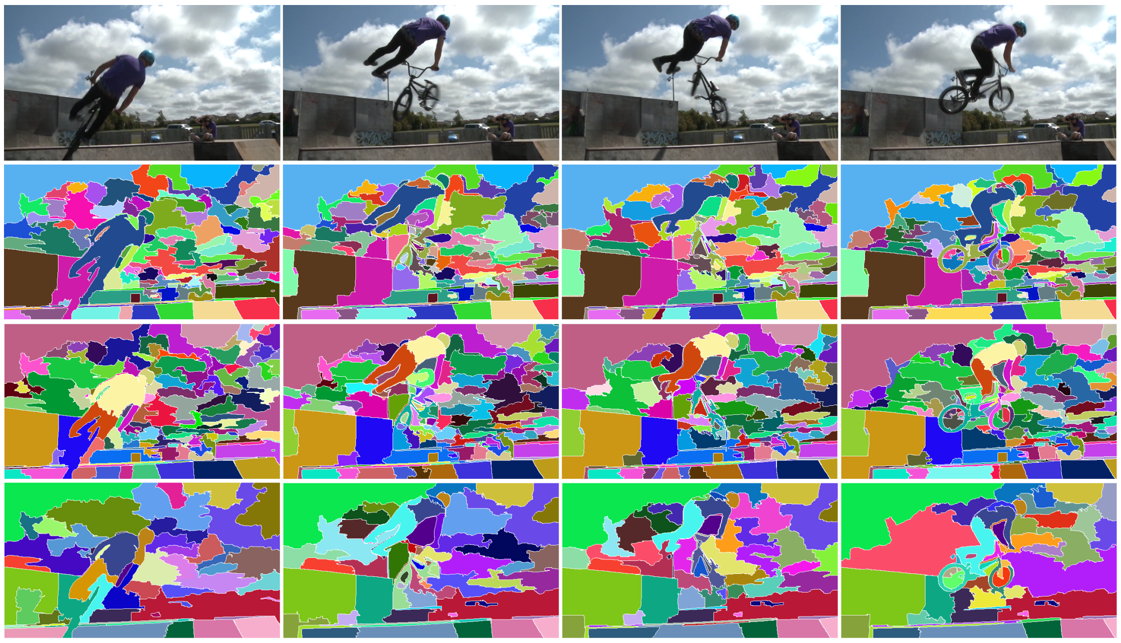


Figure 2.18. Example outputs for video BMX from SegTrack v2. From left to right: frame 1, 10, 20, and 30; Top to bottom: original frame, our results of 300 superpixels with motion (NLS+M in Table 1, 300 superixels without motion (NLS-M), and 100 superpixels without motion. Additional results can be found in the supplementary material.

followed by the swimming part of the contest for the next 10 seconds. Optimizing the entire video would effectively scramble the similarity distance values from both shots and result in a γ that is non-specific. However, if the video is processed at the two shots separately, the PGP method would obtain more specific, shot-appropriate γ's. Figures 2.16 and 2.17 show that optimizing at the entire video (vol = 1) is not always optimal, and better performance is achived when sub-dividing the volume for processing. Although we used a fixed set of subdivisions: 1, 1/4, and 1/8, processing smaller, appropriate sub-volumes is still beneficial. This would also allow for parallel video processing, where the sub-volumes can be optimized separately without performance cost.

Method parameter τ. Our proposed PGP video segmentation method has only one parameter value τ, used when a single Weibull model is selected by the model selection, to obtain the critical value γ at the τ percentile of the fitted Weibull. We have observed that the two-component WMM were selected by AIC in the vast majority of the cases; hence the selection of τ has a minimal effect on the overall accuracy. Indeed we varied τ from 0.5 to 0.9 and the resulting accuracies did not vary more than 1%. We have uniformly set τ at 0.6 throughout the presented evaluations.

Run time analysis. We conducted our experiments on a Xeon X3470 at 2.93Ghz with 32 Gb of memory. All experiments used a single core. Superpixels take roughly 1 second per frame, and our method on average takes about 170 seconds for an 85-frame video after superpixel extraction for the results of Table 2. This is more than 20 times faster than [174]'s processing time of around 4000 seconds; our combined end-to-end run time of 250 seconds on average is again 20 times faster than the total run time of 4700 - 6600 seconds of [174], which include the expensive GBH and SWA methods. Furthermore, our run time is on par with the leading streaming video segmentation method [175] although our current implementation is offline and not optimized for parallel processing.

2.5.7 Subsampling effect

In this section, additional evidence of the efficiency and robustness of the PGP video segmentation is shown, by evaluating an important advantage of the parametric nature of PGP: It allows subsampling only a subset of the edges of the graph in order to estimate the critical values that define whether two super-pixels are similar or dissimilar without any drop in performance. As we have discussed in Sec. 3.1 of the main paper, the method fits WMM over the distribution of edge weights where the L_p distance statistics are shown to be Weibull distributed in theory. Therefore we can also work with a subset of the edge weights (by uniformly random sampling) and the inherent distribution would remain Weibull, while the WMM fit should (in theory) remain similar. This is especially useful for larger videos with HD frames and/or longer duration, where the number of edges in the spatio-temporal graph is very large, and by fitting WMM over a subset of those edges would offer even higher computational efficiency.

To evaluate the effect of sub-sampling on segmentation and run-time performance, we conducted the following experiment: we use the setting of NLS optimization with motion feature with 400 superpixels per frame as the base-line PGP setting, and we progressively sub-sample at 100%, 75%, 50%, and 25% of the total

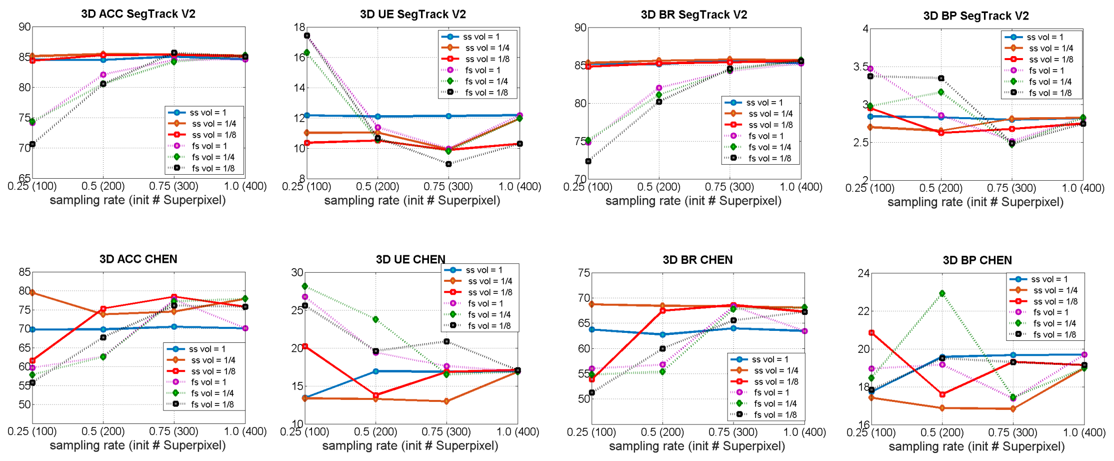


Figure 2.19. Effect of sub-sampling with respect to 3D segmentation performance. The baseline setting being used here is 400 initial superpixels per frame, 1/4 sub-volume processing, NLS optimization with motion feature (red dotted line). The dotted lines are without any sub-sampling (fs) at 100, 200, 300, and 400 superpixels per frame (x-axis labels in the paranthesis); the solid lines are with sub-sampling at 0.25, 0.5, 0.75 and 1.0 from the edge sets of the baseline. The effects are also analyzed under different sub-volume processing (full (1), 1/4, and 1/8). Note that sub-sampling at 0.25 rate of the original 400 superpixel corresponds to running non sub-sampling on 100 superpixels per frame, therefore making the solid lines and the dotted lines directly comparable. The plots show that sub-sampling barely affects the performance in segtrackV2 dataset, and only observing an effect for the 1/8 sub-volume processing setting in Chen's dataset.

number of edges. For instance, sample rate of 25% means using a random subset of one quarter of the edges for fitting the WMMs. Note that these sub-sampling rates from 400 superpixels exactly correspond to the number of edges if starting with 100, 200, or 300 superpixels per frame, thereby allowing a direct comparison with Figure 3 and 4 of the main manuscript.

The results of the experiment are shown in Figure 2.19, where the solid lines are the sub-sampled results (ss) that correspond to the sub-sample rate of the x-axis, and the dotted lines are the full-sample results (fs) that correspond to the number of superpixels (in parenthesis) of the x-axis labels. The plots of the segtrackV2 dataset indicate that sub-sampling barely affects any of the measured segmentation performance as compared to the base-line setting, while the fluctuation is more pronounced for Chen's dataset for the case of 1/8 sub-volume processing. The result is especially apparent in the segtrackV2 plots, where sub-sampling re-

Sampling Rate	Full (1.0)	0.75	0.5	0.25
Run time (sec)	345.02	279.5	221.6	168.9

Table 2.4. Run-time analysis on the effect of sub-sampling on a 85-frame video (Chen's dataset), the base-line (full) setting is using 400 initial superpixels per frame, 1/4 sub-volume processing, NLS optimization with motion feature. The sub-sampling allows PGP to further reduce run-time by over 50%, while maintaining the segmentation performance of the base-line setting (see Figure 1). The experiment is done on a Xeon X3470 at 2.93 Ghz with 32 Gb. The maximum memory usage was recorded at 4 Gb.

sults in much higher 3d segmentation performance than starting at a lower number of superpixels across the four quantitative metrics. This result suggests that one can segment to a higher number of initial superpixels in order to achieve better boundary preservation, and then still use only one quarter of the data for optimization to achieve the same level of accuracy at approximately 50% of the original run time.

Table 2.4 shows the run time analysis on performing sub-sampling at the levels of 0.75, 0.5, and 0.25 as compared to full use of the data (1.0), excluding the superpixel extraction time which takes about 1 second per frame. The run time decreased by more than 50% from the original full processing mode when working with one quarter of the edge weights. In our implementation, the result of sub-sampling is deterministic as we compute the linearly spaced vector from the full set of edges, such that the sub-sampled edges are the edges at linearly equally spaced points between edge 1 and edge n, where n is the ending edge of the graph. This is done to eliminate any randomness of the algorithm that may be undesirable for practical applications.

Chapter 3

Modeling object categorical search performance

In this second part of the computational modeling efforts, an unsupervised, generative model of category representation is introduced that uses computer vision methods to extract category-consistent features (CCFs) directly from images of category exemplars. The model was trained on 4800 images of common objects, and CCFs were obtained for 68 categories spanning subordinate, basic, and superordinate levels in a category hierarchy. Participants searched for these same categories. Targets cued at the subordinate level were preferentially fixated, but fixated targets were verified faster following a basic-level cue. The subordinate-level advantage in guidance is explained by the number of target category CCFs, a measure of category specificity that decreases with movement up the category hierarchy. The basic-level advantage in verification is explained by multiplying CCF number by sibling distance, a measure of category distinctiveness. With this model the visual representations of real-world object categories, each learned from the vast numbers of image exemplars accumulated throughout our everyday experience, can finally be studied.

3.1 Object categories

Our categories make us who we are; they are the skeleton upon which grows the rest of our psychological being. Reflecting their diverse importance, categories have been studied from multiple perspectives: as a lens through which we perceive visual and acoustic objects in the world [88][122] and the similarity relationships between these objects [99][54], and as the structure of concepts that organize our knowledge and define who we are [69][118][102]. Some approaches are also highly quantitative. The semantic organization literature uses formal methods from logic theory to understand the division of information into clusters of semantic nodes [12][39], and the category learning literature models how corrective feedback about category membership can shape our categorization decisions [13][15][80][93][110][111].

All of these approaches, however, have skirted a basic question of category representationhow might the visual features of common object categories be extracted from the many exemplar images of these objects that we encounter in our day-to-day lives?

Growing in parallel with these largely behavioral literatures has been another literature that may help answer this question. The field of computer vision is rich with operators and algorithms developed to detect members of object classes directly from pixels in images [46]. Moreover, these tools work with featurally-complex real-world categories, and their performance is evaluated using new or unseen category exemplars not used during model training. In contrast, behavioral work on category learning has placed less emphasis on real-world application and model prediction, focusing instead on how categories defined by a small number of simple features are learned from feedback (see [16]). A gap therefore exists in our fundamental understanding of categories; much is known about how simple features can be learned and used to discriminate one category from another, but little is known about the features composing the categories of common objects that populate our everyday experience. By bridging these different approaches we achieve a new understanding of categories. Our premise is that tools from computer vision can, and should, be exploited to characterize the feature representations of categories as they exist in the wild, formed simply from a lifetime of experience seeing category exemplars.

3.2 The generative modeling of visual categories

We adopt a generative modeling approach. Because generative models are usually unsupervised, they capture the implicit learning from exemplars that people and other animals use to acquire the within-category feature structure of visual object categories. Figures 3.1, 3.2, 3.3 help to make this point. A generative model learns the features that are common among the objects of a category, much like the human visual system causes the perception of rectangles in this figure by finding common features among category exemplars grouped at the basic level.

Generative models can be contrasted with discriminative models, which use supervised error feedback to learn features that discriminate target from non-target categories [157]. The vast majority of category learning studies adopt a discriminative modeling approach [15][80][110][111], which is appropriate given the heavy reliance on the artificial classification learning paradigm in this literature. A generative approach, however, is more appropriate when modeling data that do not reflect explicit classification decisions [81][85], see also [37], such as the visual search data in the present study. Our position is that generative models better capture the features of a category used to construct visual-working-memory representations of search targets, similar to the features one might call to mind when forming a mental image of a target category. If searching for a Pekin duck one would probably look for a white, mailbox-sized object with orange at the top and bottom, despite these features potentially yielding poor discrimination from poodles and pumpkins.

3.3 Hierarchies of categories

We evaluate our model within the context of a simple conceptual structure, a three-level hierarchy. Objects can be categorized at multiple levels in a conceptual hierarchy. A sea vessel powered by wind can be categorized as a sail boat (subordinate level), simply a boat (basic level), or more broadly as a vehicle (superordinate level). The basic-level superiority effect (BSE) refers to the finding that the acquisition and access of categorical information seems anchored around the basic level. It was first reported by Rosch and colleagues [132] using a speeded

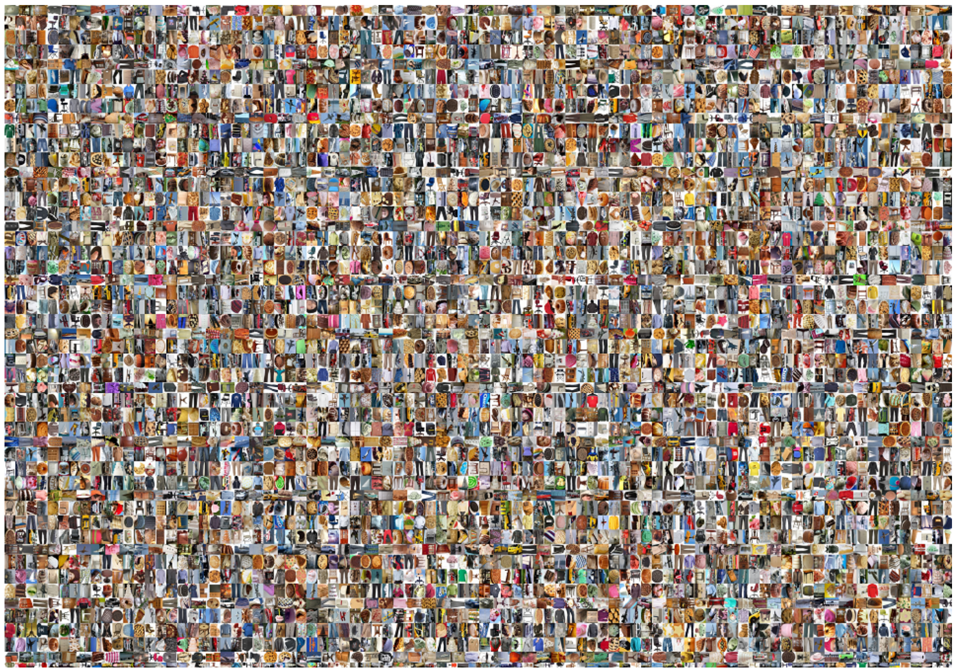


Figure 3.1. Tiny depictions of the 4,800 images of objects used to train the CCF model - Objects grouped randomly.

category verification task, where they found that people were faster in judging a picture of an object as a member of a cued category when the cue was at the basic level. Subsequent work broadened the scope of the BSE by showing it to be the preferred level used in speech, and the first nouns generally learned and spoken by children [100][131].

Explanation of the BSE has appealed to similarity relationships within and between categories. Basic-level categories are thought to maximize within-category similarity while simultaneously minimizing between-category similarity; subordinate or superordinate-level categories do one or the other, but not both [132]. [103] advanced this idea by theorizing that the BSE was a by-product of concurrent specificity and distinctiveness processes pulling categorization in opposing directions. Subordinate-level categories tend to have very specific features; Collies are medium-sized dogs with thin snouts, upright ears and white hair around their shoulders. However, these features overlap with other dog categories, making Collies sometimes challenging to distinguish from German Shepherds or Shelties. Superordinate-level categories have the opposite strengths and weaknesses. The

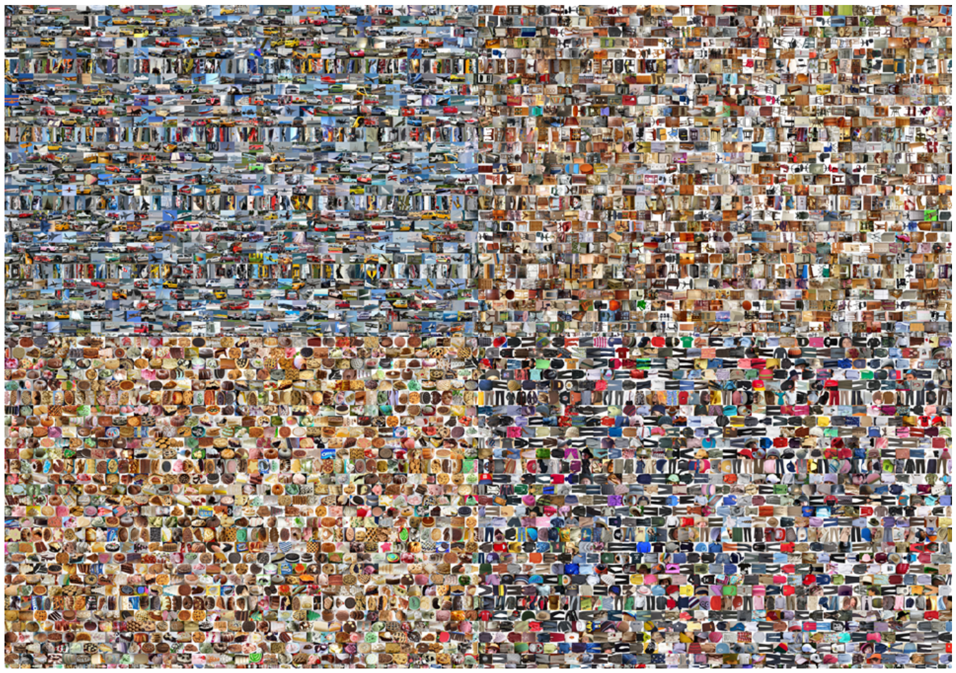


Figure 3.2. Tiny depictions of the 4,800 images of objects used to train the CCF model - Objects grouped into the 4 superordinate-level categories used as stimuli.

features of animals overlap minimally with vehicles or musical instruments, making the category distinct. However, animal features are also highly variable, making superordinate categories lacking in specificity. The basic level strikes a balance between these opposing processes, and this balance is believed to underlie the BSE. Despite their variability in appearance, dogs have many features in common yet are still relatively distinct from ducks and dolphins and dinosaurs. The present work builds on this framework by making the processes of specificity and distinctiveness computationally explicit, and applying these principles directly to images of category exemplars.

3.4 Categorical Search

We evaluate the visual representation of common object categories using a categorical search task [97][140][182][184][185]. Categorical search differs from standard visual search in that targets are designated by category (e.g., the word dog) instead of by a picture pre-cue (e.g., an image of a specific dog), a situation that rarely

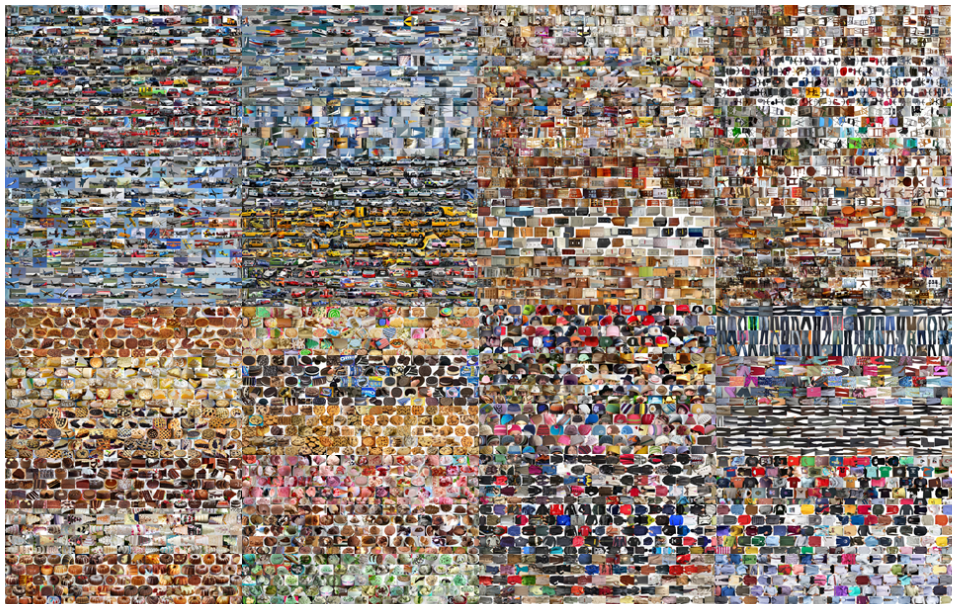


Figure 3.3. Tiny depictions of the 4,800 images of objects used to train the CCF model - Objects grouped into the 48 subordinate-level categories used as stimuli. Note that in Figure 3.2 between-category differences at the superordinate level create clearly separable visual regions, whereas in this figure, some subordinate-level categories are visually distinct from siblings while many others are not.

exists outside the laboratory. Moreover, categorical search can be meaningfully divided into two epochs, one being the time between search display onset and first fixation on a target (search guidance) and the other being the time between first fixation on the target and the correct target-present judgment (target verification). Categorical search therefore embeds a standard category verification task within a search task, making it a powerful paradigm for studying the relationship between overt attention and categorization.

We introduce a method for quantifying the visual features of common object categories, and show that these features serve both to guide overt attention to a target and to categorize it after its fixation, with a BSE appearing during this latter target-verification epoch. The fact that our model captured these disparate behavioral measures provides converging evidence, within the context of a single categorical search task, that it can successfully identify the visual features used to represent common object categories. As such, this work creates a strong theoretical bridge between the attention (search guidance) and recognition (category verification) literatures.

3.5 Behavioral methods

3.5.1 Participants

Twenty-six Stony Brook University undergraduates participated in a categorical search task. Sample size was determined based on a previous study using a similar method [98]. All participants reported normal or corrected-to-normal visual acuity and color vision, and that English was their native language. All also provided informed consent prior to participation in accordance with Stony Brook University's Committee on Research Involving Human Subjects.

3.5.2 Stimuli and Apparatus

Images of common objects were obtained from ImageNet (http://www.image-net.org) and various web sources. All images were closely cropped using a rectangular marquee to depict only the object and a minimal amount of background. Because object typicality can affect categorization and search [103][97], targets were selected to be typical members of their category at the subordinate, basic, and superordinate levels. We did this by having 45 participants complete a preliminary norming task in which 240 images (5 exemplars from each of 48 subordinate categories) were rated for both typicality and image agreement [145] at each hierarchical level using a 1 (high typicality/image agreement) to 7 (low typicality/image agreement) scale. The three most typical exemplars of each category were used as targets in the search task. Their mean typicality and image agreement was 2.29 and 2.31, respectively, and Table 3.1 lists these category names. In total there were 68 categories spanning 3 hierarchical levels; 4 superordinate-level categories, each having 4 basic-level categories, with each of these having 3 subordinate-level categories.

Eye position during the search task was sampled at 1000 Hz using an Eyelink 1000 eyetracker (SR Research) with default saccade detection settings. Calibrations were only accepted if the average spatial error was less than 0.5°, and the maximum error was less than 1°. Head position and viewing distance were fixed at

Superordinate	Basic	Subordinate
Vehicle	Car	Police Car
		Taxi
		Race Car
	Boat	Sail Boat
		Cruise Ship
		Speed Boat
	Plane	Passenger Airliner
		Biplane
		Fighter Jet
	Truck	18 Wheeler
		Fire Truck
		Pickup Truck
Furniture	Cabinet	Kitchen Cabinet
		Filing Cabinet
		China Cabinet
	Chair	Folding Chair
		Office Chair
		Dining Room Chair
	Bed	Twin Bed
		Canopy Bed
		Bunk Bed
	Table	Coffee Table
		Dining Room Table
		End Table
Clothing	Pants	Jeans
		Dress Pants
		Pajama Pants
	Shirt	Dress Shirt
		T-shirt
		Long Sleeve Shirt
	Hat	Baseball Hat
		Knit Hat
		Cowboy Hat
	Jacket	Winter Jacket
		Windbreaker
		Trench Coat
Dessert	Ice Cream	Chocolate Ice Cream
		Mint Choc. Chip Ice Cream
		Strawberry Ice Cream
	Pie	Pecan Pie
		Blueberry Pie
		Lemon Meringue Pie
	Cookie	Oreo
		Chocolate Chip Cookie
		Sugar Cookie
	Cake	Chocolate Cake
		Wedding Cake
		Bundt Cake

Table 3.1. Object categories grouped by hierarchical level.

65 cm using a chinrest for the duration of the experiment. Stimuli were presented on a flat-screen CRT monitor set to a resolution of 1024 x 768 pixels and a refresh rate of 100 Hz. Text was drawn in 18-point Tahoma font and image patches subtended $\sim 2.5^\circ$ of visual angle. Trials were initiated using a button on the front of a gamepad controller and judgments were made by pressing the left and right triggers.

3.5.3 Search procedure

A category name was displayed for 2500 ms, followed by a central fixation cross for 500 ms and finally a six-item search display (Figure 3.4). Items in the search display were image patches of objects arranged on a circle having a radius of 8°. There were 288 trials, half target-present and half target-absent. Target-present trials depicted a target and five distractor objects chosen from random non-target categories. Each participant saw one of the three selected exemplars for a given target category twice at each hierarchical level, with exemplars counterbalanced across participants. Half of the target-absent trials depicted six distractors; the other half depicted five distractors and one lure. Lures are needed to encourage encoding at the cued level (see [151]. The lure was a categorical sibling of the cued target, drawn from target images one level above in the category hierarchy (e.g., a police car when cued with "taxi", or a truck when cued with "car"). Lures at the superordinate level were drawn from other non-target categories, making them indistinguishable from the distractor objects.

3.6 Behavioral results

Error rates differed between hierarchy conditions, F(5,21) = 15.19, $p < .001$, $\eta^2 = .378$. Post-hoc tests (LSD corrected) showed that accuracy on target-present trials at the superordinate level (M = 84.9%, 95% CI [81.1, 88.7]) was lower than at the basic level (M = 91.6%, 95% CI [89.5, 93.7]) and the subordinate level (M = 92.3%, 95% CI [90, 94.6]), $ps < .001$. These additional misses are consistent with previous work [98] and reflect participants occasionally failing to recognize a target as a member of the cued superordinate category [103]. On target-absent

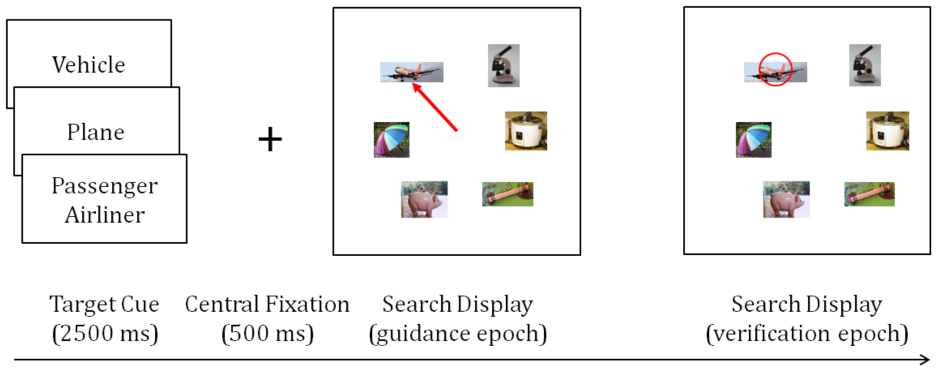

Vehicle
Plane
Passenger Airliner
+
Target Cue (2500 ms)
Central Fixation (500 ms)
Search Display (guidance epoch)
Search Display (verification epoch)

Figure 3.4. Procedure for the categorical search task. A target was designated by category name at one of three hierarchical levels, followed after a delay by a six-item target-present/absent search display. The target guidance and verification epochs used for analysis are indicated by the red graphics (not shown to participants) superimposed over the search display.

trials, accuracy was lower at the subordinate level (M = 89%, 95% CI [87, 91]) compared to the basic (M = 96%, 95% CI [94.7, 97.3]) and superordinate (M = 95.4%, 95% CI [93.2, 97.5]) levels, *ps* $< .001$. This increase in false positives was due to lures at the subordinate level being occasionally mistaken for the cued target category. Neither pattern of errors compromises our conclusions. Only correct trials were included in the subsequent analyses.

As in previous work [34][140][98], search performance was divided into target guidance and verification epochs and analyzed separately. Target guidance was defined in two ways: the time between search display onset and the participants first fixation on the target (time-to-target), and the proportion of trials in which the target was the first object fixated during search (immediate fixations). Target verification was defined as the time between a participants first fixation on the target and their correct target-present manual judgment.

Analyses of the initial guidance epoch of search revealed significant differences in time-to-target between conditions, F(2,24) = 22.08, $p < .001$, $\eta^2 = .508$. Targets cued at the subordinate level were fixated sooner on average than targets cued at the basic level, which were fixated sooner than targets cued at the superordinate level (*ps* $\leq .021$, Figure 3A, dark bars). This same trend held for immediate target fixations, F(2,24) = 13.31, $p < .001$, $\eta^2 = .456$ (Figure 3B, dark bars), a more conservative measure of guidance. Subordinate-level targets were first fixated more often than basic-level targets ($p < .001$), and basic-level targets were first fixated more often than superordinate-level targets ($p < .001$). Initial saccade latency did not reliably differ between cueing conditions (p = .452), suggesting that these differences were not due to a speed-accuracy tradeoff. Differences between conditions were also found during the verification epoch of search, F(2,24) = 5.71, p = .006, $\eta^2 = .215$. As shown in Figure 3.5C (dark bars), these differences took the form of a BSE; targets cued at the basic level were verified faster than those cued at the subordinate level (p =.01) and superordinate level (p = .004). These findings not only extend previous work in showing that the hierarchical level in which a target is cued differentially affects target guidance and verification processes [98], they create a challenging guidance and verification behavioral ground truth against which our generative model of category representation can be evaluated.

3.7 Model methods

Two distinct effects of category hierarchy were found in the behavioral data: a subordinate-level advantage in target guidance and a basic-level advantage in target verification. We explain both of these behavioral patterns using a single unsupervised generative model that extracts features from images of category exemplars and then reduces the dimensionality of this representation to obtain what we refer to as Category-Consistent Features (CCFs). Figure 3.6 provides an overview of this model.

3.7.1 Feature extraction

Using the identical category hierarchy from the behavioral experiment, we built from ImageNet and Google Images an image dataset for model training. This consisted of 100 exemplars for each of the 48 subordinate-level categories (4,800 images in total; see Figure 1 for tiny views of these images), with each exemplar being an image patch closely cropped around the depicted object. Exemplars for basic-level and superordinate-level categories were obtained by combining the subordinate children exemplars under the "parent" categories. For example, the basic-level boat category had 300 exemplars consisting of 100 speed boats, 100 sail

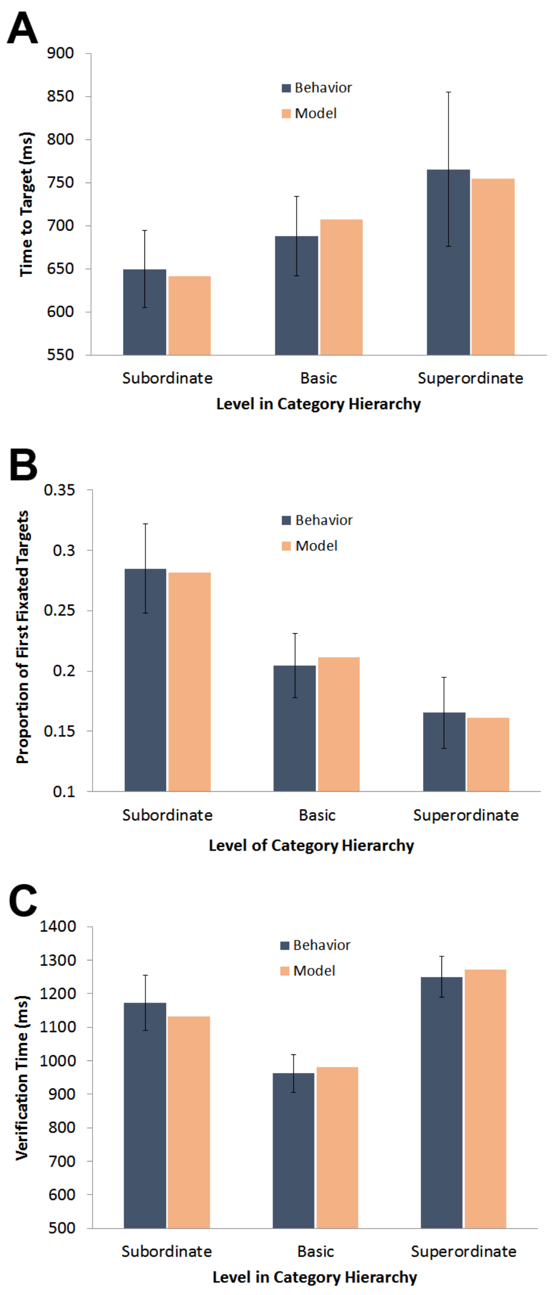


Figure 3.5. Behavioral results (dark bars) for the categorical search experiment plotted with the CCF model output (light bars) for (A) time to the first fixation on the target, (B) proportion of immediate fixations on the target, and (C) time from first fixation on the target until the correct target-present button press decision. Error bars indicate 95% confidence intervals.

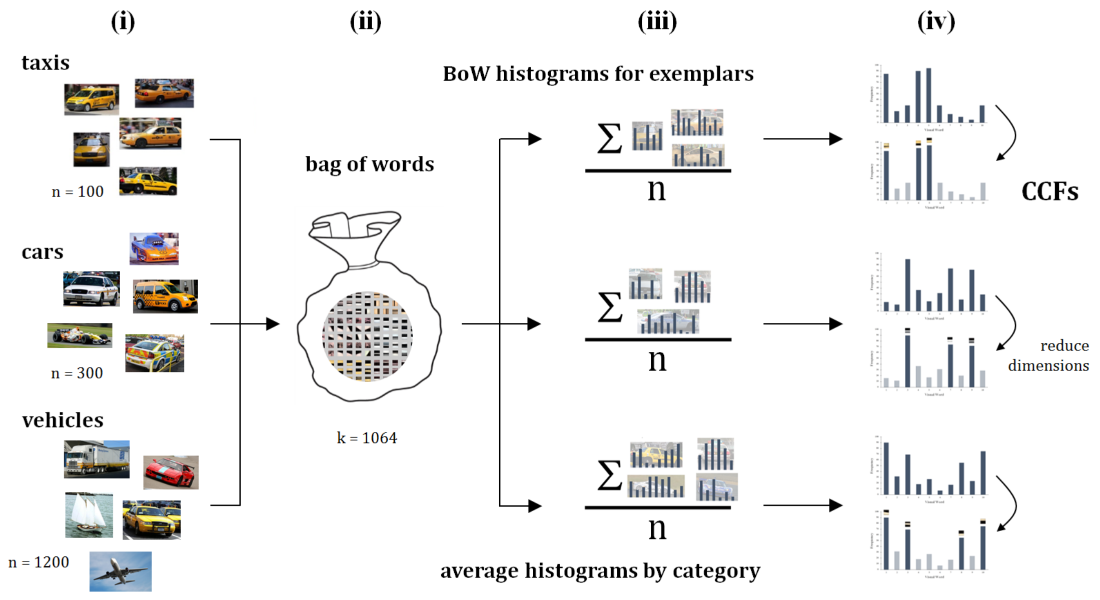


Figure 3.6. An overview of the Category-Consistent Features Model. (i) 100 images of object exemplars were collected for 48 subordinate-level categories. These exemplars were combined to create 16 basic-level categories (each with 300 exemplars) and 4 superordinate-level categories (each with 1200 exemplars). (ii) SIFT and color histogram features were extracted from each exemplar and the Bag-of-Words (BoW) method was used to create from these a common feature space consisting of 1064 "visual words". (iii) 1064-bin BoW histograms were obtained for each exemplar, where the bins correspond to the visual words and bin height indicates the frequency of each feature in the exemplar image. BoW histograms were averaged by category to obtain 68 averaged histograms, each now having a mean frequency and variability associated with each visual word. (iv) Features in these averaged histograms having too low of a frequency or too high of a variability were excluded, resulting in a lower-dimensional feature representation of each category that we refer to as category-consistent features (CCFs)those highly informative features that are present both frequently and consistently across the exemplars of a category.

boats, and 100 cruise ships, and the superordinate-level vehicle category had 1,200 exemplars consisting of the 300 exemplars from each of the boat, car, truck, and plane siblings.

The first step in representing an object category is the extraction of features from exemplars. Two types of features were used: the Scale Invariant Feature Transform (SIFT) and a color histogram feature. SIFT features capture the structure of gradients in images using 16 spatially distributed histograms of scaled and normalized oriented-edge energy [94]. The color histogram feature [158] captures the distribution of hue in an image, represented in the current implementation by 64 bins of Hue in 360° HSV color space. Using dense sampling (and discarding samples from uniform regions), we extracted 5 scales of SIFT descriptors from patches of 12x12, 24x24, 36x36, 48x48, and 60x60 pixels, and color histogram features from a fixed-size 20x20 pixel patch surrounding the center positions of every SIFT descriptor in each of the 4,800 exemplars. Color histograms were pooled over patches within exemplars to create a single 64-bin color histogram for each. However, to compare SIFT features between exemplars it is necessary to find a common feature space, and for this we used the Bag-of-Words (BoW) method [44]. The SIFT features extracted from each exemplar were put into a metaphorical bag, and k-means clustering was performed on this bag to obtain a common vocabulary of 1,000 visual words (k = 1000). The 64 hue features from the color histogram were concatenated to this vocabulary, yielding a 1064-dimensional feature space in which each of the 4,800 exemplars could be represented as a BoW histogram, where the bins of the histogram correspond to the 1,064 visual word features and the height of each bin indicates the frequency of that feature in a given exemplar.

3.7.2 Category-consistent features (CCFs)

Having put all the category exemplars in a common feature space, the next step is to find those features that are most representative of each target category. This process begins by averaging over category the BoW exemplar histograms to obtain what might be called a proto-type for each category [130], although we avoid using this theoretically-laden term so as not to associate a proto-type with a particular step in the computation of CCFs. Each averaged category histogram captures the mean frequency that each of the 1,064 features appeared in the category exemplars, along with the variance for each of these means (see Figure 3.7 for a partial averaged histogram for the taxi category, and Figure 3.8A for a visualization of every complete histogram contributing to the taxi category averaged histogram).

Although methods abound in the computer vision literature for selecting features (e.g., [40][156], most of these are tailored to finding features that discriminate between categories of objects for the purpose of classification. This makes them poorly aligned with our generative approach. Alternatively, feature selection un-

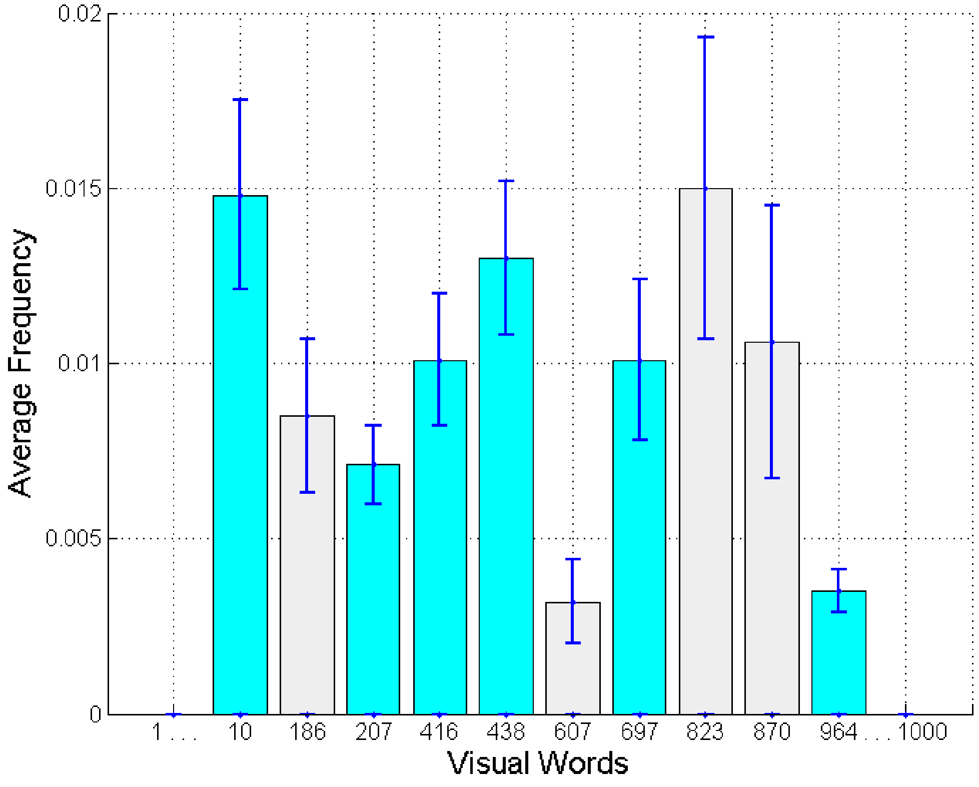


Figure 3.7. A partial plot (10 of the 1000 SIFT-feature bins) of the histogram obtained by averaging the 100 BoW histograms for the exemplars of the taxi category. Following this averaging each visual word bin has both a mean frequency and a variance (error bars indicate standard error). Blue bars indicate some of the visual words from the averaged histogram that would ultimately become CCFs for the taxi category representation. Gray bars indicate examples of visual words that were excluded from the CCF representation due to either excessive variability or too low of a mean frequency. Note that frequencies are generally low because bin frequency sums to one for representation as a probability distribution. The average bin frequency is therefore 1/1000 or 0.001, considerably smaller than all of the bin frequencies shown.

der the CCF model is grounded in signal detection theory [55]. We assume that features having a high frequency and a low variance are more important than the rest, and use these simple measures to prune away the others. Specifically, features having a high mean frequency over the category exemplars are identified using the interquartile range rule: X' = X ¿ 1.5x(Q3 (X) Q1 (X)), where X is the average frequency of the 1,000 SIFT features or 64 color features (performed separately) for a given category histogram, and Q1 and Q3 are the first and third quartiles, respectively. For each of these frequently occurring features we then compute the inverse of its coefficient of variation by dividing its mean frequency by its standard deviation, a commonly used method for quantifying a scale-invariant signal-to-noise ratio (SNR; [137]).

Finally, we weight each feature in the above set by its SNR, then perform k-means clustering, with k=2, on these feature weights to find a category-specific threshold to separate the important features from the less important features. The CCFs for a given category are those features falling above this threshold. CCFs are therefore the features that occur both frequently and reliably across the exemplars of a category, with each category having different CCFs in this 1064-dimensional feature space. These CCFs, and not the noisier category histogram formed by simply averaging exemplar histograms, are what we believe constitutes the learned visual representation of an object category (see Figure 3.8A for the CCFs from the taxi category, and how they compare to the corresponding averaged category histogram from Figure 3.8B).

3.8 Model results

Can the CCF model capture the patterns of target guidance and verification observed in behavior? We show that these two very different patterns can be modeled as different properties of the same CCF category representations.

3.8.1 Test guidance

The behavioral data showed that target guidance got weaker as targets were cued at higher levels in the category hierarchy. Guidance was strongest following a subordinate-level cue, weaker following a basic-level cue, and weakest following a superordinate-level cue. How does the CCF model explain target guidance, and its change across hierarchical level?

According to the CCF model, target guidance is proportional to the number of CCFs used to represent a target category. The logic underlying this prediction is straightforward. To the extent that CCFs are the important features in the

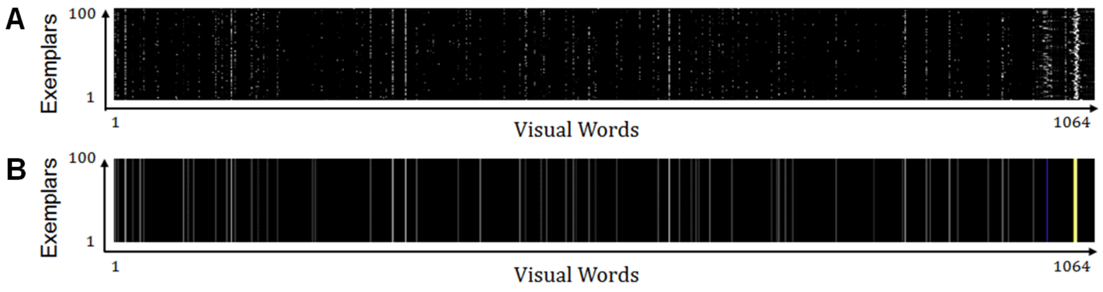


Figure 3.8. Histograms of all the exemplars from the taxi category are "stacked" into a rectangular block. Each row shows the entire set of 1064 visual words corresponding to one exemplar, and each column shows the values for a given visual word for each of the 100 exemplars. (A) Averaged histograms of the type plotted in Figure SM2 (blue and gray bins). Brighter pixels denote a higher frequency of features. Note that stacking the histograms makes it possible to visualize feature commonalities across exemplars, with features common to most exemplars appearing as bright vertical bars. Broken vertical bars indicate features appearing inconsistently across exemplars, and dark regions indicate features that are unimportant in the representation of the taxi category. (B) Stacked CCF histograms (only the blue bins from Figure SM2). Vertical bars indicate the CCFs selected from the features in (A), with brightness indicating the mean feature value for a given visual word divided by its standard deviation. Bins 1001-1064 correspond to the hue visual words, where yellows appear prominently across the taxi exemplars. Note that the process of selecting CCFs serves to remove noise from the averaged histograms, thereby accentuating the features most important for the category representation.

representation of a category, more CCFs mean a better and more specific category representation (see also [139]). A target category having a larger number of CCFs would therefore be represented with a higher degree of specificity and, consequently, fixated more efficiently than a target having a sparser "template" [140]. As shown in Figure 3.9 (dark bars), the number of CCFs per category indeed varied with hierarchical level; the subordinate-level categories had the most CCFs, followed by the basic-level and finally the superordinate-level categories. This too was predicted. Subordinate-level categories have more details in common that can be represented and selected as CCFs, whereas at the higher levels greater variability between exemplars cause features to be excluded as CCFs, resulting in a smaller total number.

Figure 3.5A shows that the effect of hierarchical level on target guidance can be captured simply by the mean numbers of CCFs extracted for the 48 subordinate-level target categories, the 16 basic-level categories, and the 4 categories at the

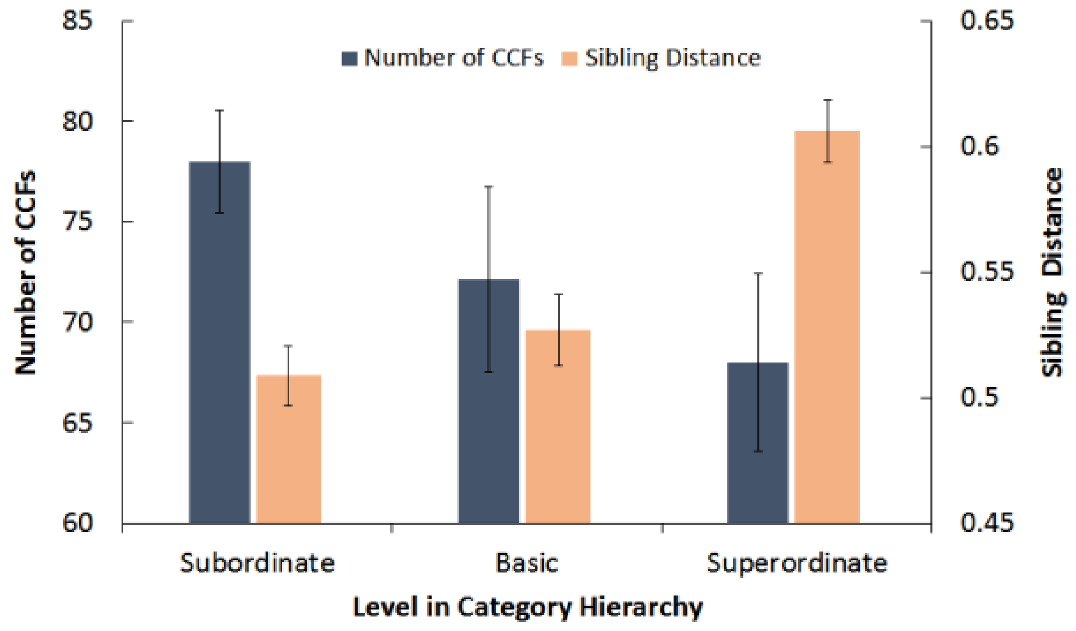


Figure 3.9. Results from the CCF model showing the mean number of category-consistent features (dark bars) and mean sibling distances (light bars) by hierarchical level. Error bars indicate standard error of the mean, computed by treating the number of categories at each level as the number of sample observations (n).

superordinate level. Specifically, the light bars in Figure 3.5A plot 1/CCF# to capture the increase in time-to-target with movement up the category hierarchy, while Figure 3B plots the raw numbers of CCFs to capture the downward trend in immediate target fixations. After linearly transforming the number of CCF data to put it into the same scales as the behavior, the models behavior fell within the 95% confidence intervals surrounding all six of the behavioral means. This finding has implications for search theory. It suggests that the stronger target guidance reported for exemplar search (e.g., targets cued by picture preview) compared to categorical search (e.g., [140] may be due, not to a qualitative difference in underlying processes, but rather a quantitative difference in the number of good features in the target representation used to create the priority map that ultimately guides search [183]. Many strong guiding features can be extracted when the opportunity exists to preview the specific target exemplar, but strong guidance in a categorical search task requires a target category represented by many CCFs.

3.8.2 Target verification

To the extent that more CCFs enable greater specificity in the target representation the converse is also true. Movement up the category hierarchy incurs a cost reflecting decreasing numbers of CCFs, with superordinate-level categories receiving the greatest cost, subordinate-level categories the least, and basic-level categories falling in between. We show that target verification can be modeled by combining this trend with a second and opposing trend, one based on the distance to neighboring categories.

3.8.2.1 Sibling Distance

In the context of a categorical search task, target verification refers to the time between first fixation on the target and the correct target-present judgment. The CCF model predicts that this time is proportional to the distance between the CCFs of the target category and the features of the target's categorical siblings, where siblings are defined as categories sharing the same parent (one level up in the category hierarchy). This logic is also straightforward. Verification difficulty should depend on the distance between the target category and the most similar non-target categories in a test set; as this distance increases, target verification should become easier. This follows from the fact that smaller distances create the potential for feature overlap between categories, and to the extent this happens one category might become confused with another. In the present context, these least distant and most similar non-target exemplars would be the categorical siblings of the target. If the target was a police car the non-target objects creating the greatest potential for confusion would be exemplars of race cars and taxis, with these objects largely determining the verification difficulty of the target. Indeed, these siblings were the same objects used as categorical lures in order to obtain our behavioral demonstration of a basic-level advantage.

To model the distance between a target and its categorical siblings we took the CCF histogram for each sibling and found the mean chi-squared distance between it and the BoW histogram for every exemplar under the parent category. We denote the full set of BoW features as F = {1,...,1064}, and the CCFs for target category k as F', such that $k \in \{1, ..., 68\}$ and F'_k is a subset of F, $F'_k \subseteq F$. Chi-squared distance is defined by:

$$\chi^2(x, y) = \frac{1}{2} \sum_i \frac{(\phi_i(x) - \phi_i(y))^2}{\phi_i(x) + \phi_i(y)}, \tag{3.1}$$

where x and y are the two histograms to be compared, and i is the value at the ith bin of the 1064-bin feature histogram. Note, however, that following the dimensionality reduction that occurred in selecting the CCFs the sibling CCF histograms may no longer be in the same feature space as the BoW histograms for the exemplars. To compute the above-described distances we therefore must put the CCF and BoW histograms back into a common feature space, and we do this by adopting the following algorithm. For comparisons between a given CCF histogram of category k and its own BoW histogram exemplars, chi-squared distances were computed for only those bins in the BoW histograms for which there were corresponding bins in the CCF histogram, such that $i \in F'_k$. For comparisons between exemplar histograms from category j and histograms from a sibling category, k, chi-squared distances were limited to the feature space formed by the union of the two CCF histograms, such that $i \in \bigcup(F'_j, F'_k)$ for Eq. 3.1.

To clarify with an example, consider only two sibling categories, A and B, each having non-identical CCF bins ($F'_A \neq F'_B$) forming CCF histograms $\mu(A)$ and $\mu(B)$ based on exemplars An and B_n, where n describes all of the exemplars for a given category (either 100, 300, or 1200 for the subordinate, basic, or superordinate categories, respectfully, used in this study). We compute the chi-squared distances between $\mu(A)$ and the BoW histograms obtained for each of A's exemplars, A_n, for which there are corresponding bins in F'_A. If we denote this distance between the CCF histogram of A and all the A exemplar histograms as $d_{A,A}$, then $d_{A,A} = \chi^2(\mu(A), A_n | i \in F'_A))$. We also compute the chi-squared distances between $\mu(A)$ and the BoW histograms obtained for each of B's exemplars, $d_{A,B}$, with these comparisons now limited to the bins forming the union of the F'_A and F'_B CCF histograms, such that $d_{A,B} = \chi^2(\mu(A), B_n | i \in \cup(F'_A, F'_B))$. Doing the same for $\mu(B)$ and the BoW histograms of the B exemplars and the A exemplars (based on the union of CCFs F'_A and F'_B), gives us $d_{B,B}$ and $d_{B,A}$, respectively. Finally, taking the mean over the hundreds of distances in the sets $d_{A,A}$, $d_{A,B}$, $d_{B,B}$, and $d_{B,A}$, we obtain an estimate of the distance between sibling categories A and B, which we refer to as sibling distance.

To the extent that smaller sibling distances mean more difficult category verification decisions, the CCF model predicts a verification benefit for target categories designated at higher levels in the hierarchy. Computing sibling distances for all 64 target categories, then averaging within hierarchical level, we found that subordinate-level categories were closest to their sibling exemplars and that superordinate-level categories had the largest mean sibling distance (Fig 3.9, light bars). Verification times for race cars should therefore be relatively long due to the proximity of this category to taxi and police car exemplars, whereas shorter verification times are predicted for vehicles because of this categorys greater mean distance to Oreo cookies and other sibling exemplars. The basic-level categories again fall between these two, enjoying neither a verification cost nor a benefit.

3.8.2.2 Basic-level Superiority Effect

Rather than the predicted speed-up in target verification times with movement up the hierarchy, we found instead the often-observed basic-level advantage; faster verification for targets cued at the basic level compared to the subordinate or superordinate levels. However, and consistent with early explanations [103], we explain this BSE as a tradeoff between two interacting processes, specificity, which we relate to the number of CCFs existing for a given category, and distinctiveness, which we relate to the distance between the CCFs of a given category and the features of its sibling exemplars. Indeed, the countervailing trends illustrated in Figure 3.9 reflect these opposing specificity and distinctiveness processes. To model the net impact of these interacting processes on target verification time we simply multiple one by the other. Specifically, for each target category we multiply its number of CCFs by its sibling distance to obtain a (unit-less) estimate of that categorys verification difficulty. These results, averaged by hierarchical level and linearly transformed into the behavioral scale, are shown in Figure 3.5C (light bars). As was the case for target guidance, model estimates once again fell within the 95% confidence intervals surrounding the behavioral means. In a control experiment we also showed that randomly selecting the same numbers of visual word features failed to produce the BSE observed in behavior, thereby validating the CCF modelany features will not do, these features have to be CCFs (see Figure

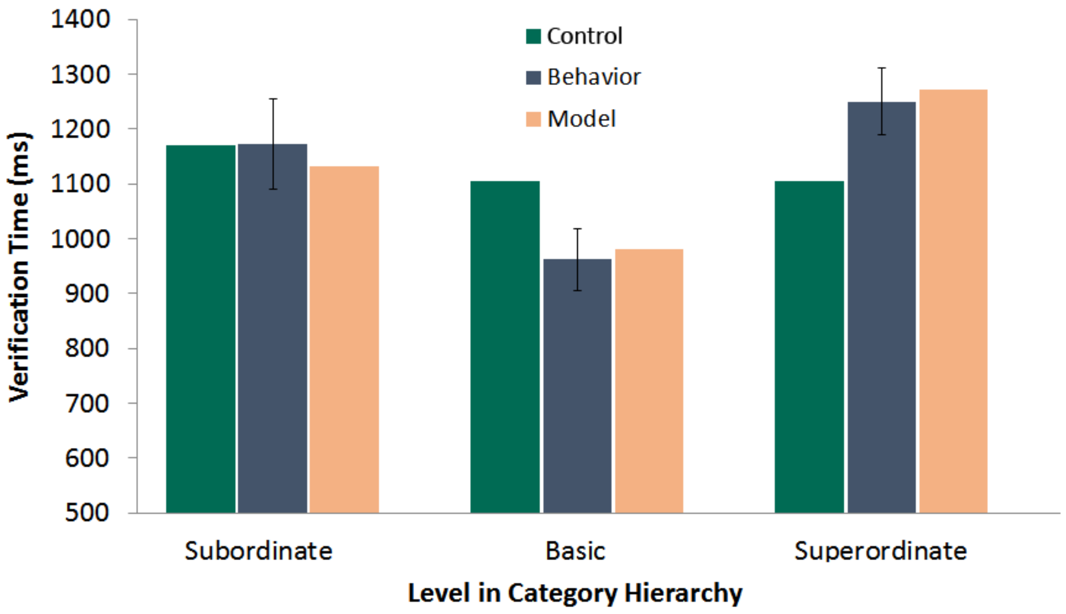


Figure 3.10. Results from a control experiment testing whether any features from the visual dictionary, regardless of whether they were CCFs, would be as good as the CCF model in characterizing the behavior data. The number of CCF features were found for each of the 68 categories, then the same number of features were randomly selected from the category's averaged BoW histogram (Figure SM2 and SM3A). Sibling distances for each category were then computed based on these non-CCFs and multiplied by the respective CCF number to get a category verification estimate, just as in the CCF model. Averaging these estimates by hierarchical level gives the data in green (leftmost bars), which show a much poorer fit to the behavioral data and CCF model (re-plotted from Figure 3C). This experiment serves as a validation of the CCF approach; only CCFs were able to capture the BSE observed in behavior, the same numbers of randomly selected features could not.

3.10).

Although categories at the subordinate level have the most CCFs (a specificity benefit), they also have the smallest sibling distance (a distinctiveness cost). This results in an intermediate degree of verification difficulty. Superordinate-level categories have the opposite relationship, relatively few CCFs (a specificity cost) but a large sibling distance (a distinctiveness benefit). This, again, results in an intermediate degree of verification difficulty. The basic-level categories occupy a privileged position in the hierarchy that avoids these two extremes. They have a relatively high number of CCFs while also being relatively distant from their sibling exemplars. This favorable trade-off between distinctiveness and specificity produces the BSE, faster verification at the basic level relative to the levels above and below.

3.8.3 Predicting search behavior using the CCF model

The above-described analyses demonstrated that the CCF model captured trends observed in target guidance and verification across the superordinate, basic, and subordinate levels, but can this model also predict behavior occurring within each of these categorical levels? As a first step towards answering this question, we conducted a trial-by-trial analysis to predict how strongly the cued target category would guide search to an exemplar of that target.

For each target-present trial, we computed the chi-squared distance between the CCF representation of the target category and the target exemplar appearing in the search display, then correlated these distances with the time-to-target measure of search guidance obtained for every trial. To evaluate the CCF model predictions we used the leave-one-out method to derive a Subject model, which indicates how well the mean target guidance of n-1 subjects predicts the guidance of the subject left out. This analysis provides an upper limit on the predictive success of the CCF model, as correlations higher than the Subject model would not be expected given subject variability in their guidance behavior.

Figure 3.11 plots time-to-target correlations for the CCF model and the corresponding Subject model at each hierarchical level. Paired-group t-tests revealed that correlations did not reliably differ between the CCF and Subject model at the subordinate ($p = .078$) or basic ($p = .334$) levels, although correlations were significantly different at the superordinate level ($p < .001$). The poor correlation at the superordinate level is consistent with the absence of guidance reflected in the chance-level proportion of immediate target fixations at this level (Figure 3.5B). These findings suggest that the CCF model not only predicted the fine-grained search behavior occurring on individual trials, these predictions at the subordinate and basic levels were as good as could be expected given agreement in the participants behavior.

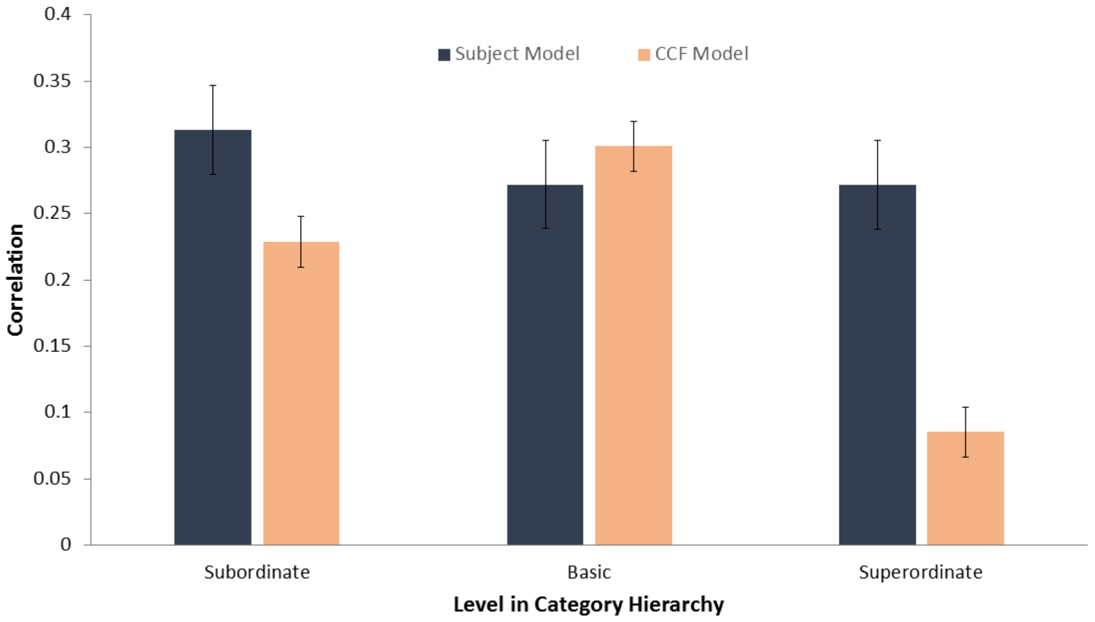


Figure 3.11. Correlations between the trial-by-trial CCF model predictions and time-to-target (light bars), averaged by level (Fisher z-transform) and plotted with correlations from a corresponding Subject model (dark bars) that captures agreement among participants in their guidance behavior. Data were from all correct trials in which the target was fixated. Error bars indicate one standard error.

3.9 Discussion

Categories determine how we interact with the world. Understanding the forces that shape category representation is therefore essential to understanding behavior in every domain of psychological science. We introduce a computational model of category representation, one that accepts image exemplars of common object categories and finds the features appearing frequently and consistently within each categorys exemplarsreferred to here as category-consistent features (CCFs).

We validated the CCF model through comparison to behavior in a categorical search task. Categorical search is important, and has diverse applications. Each time a security screener searches for a weapon, or a radiologist searches for a tumor, they are engaging in categorical search. Categorical search is also unique in that this single task enables study of the representations used to guide attention to categorically-defined targets and the representations underlying the recognition of these objects as members of the target category. We manipulated the hierarchical level in which categorical targets were cued and found that these attention and recognition processes were expressed in very different behavioral patterns. One pattern was a subordinate-level advantage in target guidance; targets cued at the subordinate level were preferentially fixated compared to targets cued at the basic or superordinate levels. Another pattern was a basic-level advantage in target verification; fixated objects were verified faster as members of the target category following a basic-level cue compared to subordinate or superordinate-level cues.

Under the CCF model, both patterns depend on the number of CCFs extracted from exemplars at each hierarchical level. Target guidance weakens with movement up the category hierarchy due to exemplar variability at the higher levels restricting the formation of CCFs, resulting in less effective target templates for guiding search [113]. The CCF model advances existing search and visual working memory theory by making explicit the processes of extracting visual features from image exemplars of real-world categories and consolidating these features into lower-dimensional category representations (CCFs) that can be used to guide search. It also provides a theory for understanding effects of category hierarchy [98] and target specificity [140] on search behavior; search is guided more efficiently to targets specified lower in the category hierarchy because these objects would usually be represented using more CCFs. Target verification was modeled as a multiplicative interaction between CCF number and sibling distancea measure of similarity between the CCFs of a target category and the features of its sibling exemplars. In doing this, the CCF model appealed to the core principles of specificity and distinctiveness that have been guiding categorization research for decades [103]. The number of CCFs maps onto the idea of specificity. Subordinate-level categories are the most specific because they give rise to many CCFs. Sibling distance maps onto the idea of distinctiveness. Verification suffers with movement down the hierarchy because target representations start to share too many features with their closest categorical neighbors. The CCF model advances categorization theory by making these core principles computationally explicit and applicable to real-world object categories.

Of potentially even broader theoretical significance is the question of whether search and categorization share the same target representation; are the visual features used to guide overt attention to a categorical target in a search display the same as those used to categorize the target once it is fixated? The CCF model suggests that this is the case, and to the extent that this suggestion is supported through converging evidence [182] a strong theoretical bridge will be built between the attention and categorization literatures. Future work will also strengthen the bridge to the computer vision and computational neuroscience literatures by attempting to learn CCFs using a deep convolutional neural network (CNN). Supervision is a powerful learning tool [72], and combining it with the generative extraction of features from exemplars may lead to significant advances in the understanding of category representation.

The CCF model makes possible the rigorous study of how visual object categories can be learned and represented from the vast numbers of diverse image exemplars accumulated throughout our everyday experience. Recent decades have seen scientific doors to the real world open for many psychological processes. The CCF model opens another such door into categorization.

Chapter 4

Biologically plausible deep learning models

4.1 Deep learning in computer vision

Deep learning is a relatively new application of neural networks within the field of machine learning, which was made popular since the breakthrough work by Krizhevsky and colleagues in 2012 [79]. In that seminal work, the authors designed a 8-layer convolutional neural network (CNN) that was trained and tested on the Large Scale Visual Recognition Challenge 2012 (ILSVRC2012, also known as the ImageNet dataset) [138], which contained 1.2 million images from 1000 object categories. This 8-layer CNN, which we denote as AlexNet from here on, was trained in a supervised fashion using back-propagation, achieved a state-of-the-art top-1 and top-5 error rate of 36.7% and 15.4% on the classification-task validation set of ILSVRC2012 (with an ensemble of seven models and multiple testing crops), while the second place entry had a top-1 error rate of 45.7%. The winning margin was so large that it caught the attention of the computer vision and machine learning fields, and made deep CNN extremely popular both inside and outside these fields ever since. For the scope of this thesis, we focus our discussion on the feed-forward CNNs in the task of image classification.

While the publication of AlexNet made deep CNNs immensely popular, the concept of CNN was not new. CNN has been studied and applied in handwriting digit recognition in 1998 by LeCun and colleagues [84]. In the work, a 5-layer CNN (LeNet, two convolutional layers and three fully-connected layers) was proposed to learn and classify handwritten digits from the MNIST dataset [1], and achieved an error rate of 0.8%. AlexNet's design is heavily based on the LeNet architecture, where the convolutional layers perform automatic feature learning and the fully-connected layers perform feature combination and classification, but made it deeper: from LeNet's two convolutional layers into five, which is where the word "deep" CNN comes from. In additional to the deeper architecture of AlexNet, it also incorporated other techniques such as local response normalization, rectified linear units (ReLU) [105], overlapping pooling, dropout [146], and various ways of data augmentation. The combinations of these techniques, and a deep convolutional neural network architecture, have made it possible for the deep CNNs to be applied to a wide range of tasks.

Since the introduction of AlexNet, many new works have been proposed with deeper, and more complex network architectures that continue to improve the image classification error rate on the ImageNet dataset. Shortly after AlexNet, the research group at University of Oxford showed that the depth of a network is an important recipe for improving the classification accuracy, and proposed the "VGG" series of CNNs that have depth ranging from 11 to 19 layers and the corresponding model parameters ranging from 133 to 144 million, improving the ImageNet image classification top-1 and top-5 error rate down to 24.4% and 7.1%, respectively [143]. While going deeper with CNNs is important for achieving better accuracy, a deeper network also requires more memory on the hardware. The major contribution of this work is therefore to design a deep network that is relatively more scalable, and able to fit in GPU memories for training. This was made possible by using primarily 3×3 sized convolutional units (or kernels), as oppose to AlexNet that has units of size 11×11, 5×5, and 3×3. The rational behind the VGG design principles was that by stacking two 3×3 convolutional units one after another (stride one), the output has a receptive field that is equivalent to using a single 5×5 unit, but with reduced number of parameters: a 3×3 unit followed by another 3×3 unit at the next layer requires $9 + 9 = 18$ parameters, less than a single 5×5 unit that has 25 parameters. This allows the network to have a large number of layers containing just 3×3 units (deep), with an increasing number of

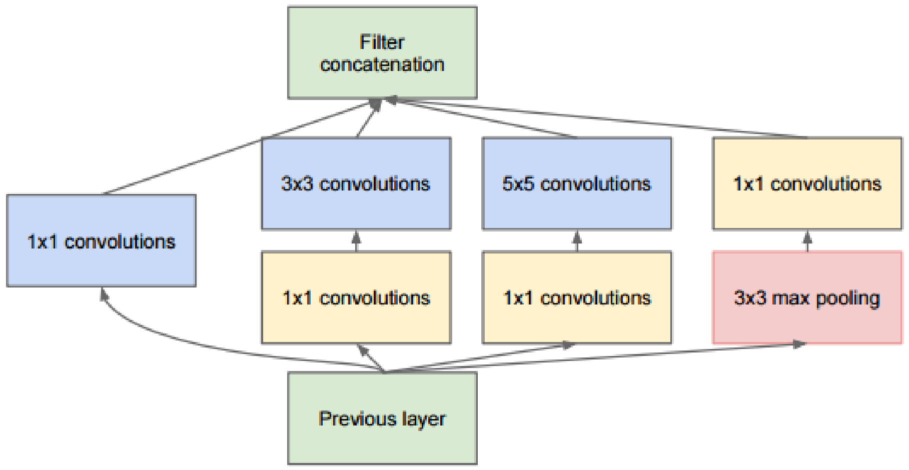


Figure 4.1. Inception module as part of GoogLeNet, image is taken from [150].

convolutional units at each layer toward the output layer. While the deep VGG networks were able to fit in a GPU memory, it still contains a massive number of parameters (133 to 144 million, as compared to 61 million of AlexNet), which was subsequently addressed by a model developed by Google.

The next major step was taken by Google, where GoogLeNet was developed and was the winner of the ILSVRC 2014 challenge with a top-5 error rate of 6.67% [150]. In this work, a 22-layer deep CNN was proposed that contains just 5 million parameters, and further improved the ImageNet validation set classification top-5 error rate down to 6.7%. Their design is based on stacking nine specialized modules called “Inception”, each containing parallel convolutional units with sizes 1×1, 3×3, 5×5, and a within-module max-pooling. As shown in Figure 4.1, each Inception module also contains a 1×1 layer prior to each of the 3×3 and 5×5 layers, that act as dimensionality reduction blocks. Other design novelties include the use of auxiliary loss-layers at the 3rd and the 6th inception module, and the omission of the fully-connected layers at the top level of the network. The auxiliary loss-layers help reduce the vanishing gradient problem that is prone to happen in deeper networks, and the omission of the fully-connected layer is the primary reason for the low number of parameters in GoogLeNet, as the fully-connected layers contain over 95% of the total model parameters such as in AlexNet.

As going deeper has been consistently shown to be the most effective way

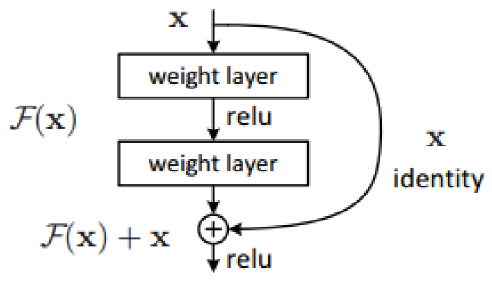


Figure 4.2. A residual learning block, image is taken from [60].

in improving the large scale image classification accuracy, an ultra deep network with 152 layers (ResNet) was proposed by He and colleagues and achieved top-5 error rate of just 3.57%, that rivals human performance [60]. In this work, the authors showed that while going deeper is important, simply stacking more 3×3 convolutional layers does not work: a 56-layer network has higher training and testing error than a 20-layer network. Instead, they proposed to perform "residual learning": instead of hoping the stacks of layers would learn to map to the objective better, learning to map to the residual is easier. As shown in Figure 4.2, the idea of residual learning is that if $F(x) + x$ is already similar to the input x, then the weight layers $F(x)$ can be easily updated to near zero. This is made possible by the bypass connection that passes a copy of input directly to combine (using addition) with the outputs of the convolutional weight layers. Without the bypass identity connection, the convolutional layers are forced to make updates according to the error gradient, and the vanishing gradient problem can easily arise if there are too many layers; but with the bypass connection that passes a copy of the input by skipping the intermediate convolutional layers, the layers can just be geared to zero if the input is already close to the desired output. Therefore, the residual learning network not only further advanced the image classification accuracy to surpass human-level performance (top-5 error of 5.1% as reported in [138]), but is also the first work to demonstrate successful application of bypass connections within a CNN.

4.2 Deep learning in biological vision sciences

The success of deep CNNs in computer vision and machine learning has also caught the attention of the biological vision community, most prominently in the area of computational neuroscience [176][75]. The feed-forward deep CNNs have architectures that loosely resemble the ventral-stream processing of the brain (the primary object recognition pathway), and several works have studied the similarities of object representations between deep CNNs and the Inferior Temporal (IT) cortical representations.

Khaligh-Razavi and Kriegeskorte conducted a study to evaluate the representational similarities between IT in primates and a variety of computational models, that include neuroscientific models such as HMAX, VisNet, and computer vision models based on SIFT, GIST, PHOG, PHOW, and a deep CNN [73]. HMAX is a family of biologically-plausible ventral stream models that stands for Hierarchical Model and X, due to its use of max-pooling operations between simple and complex cell units [127][153], with the most recognized model extension from [141]. VisNet is yet another hierarchical model of the ventral visual pathway [129], organized into four layers of self-organizing maps. SIFT [94], GIST [112], PHOG and PHOW [22][21] are local and/or global image feature descriptors for object and scene recognition in computer vision. The authors of [73] compared the brain responses in primate (monkey and human) IT using fMRI measurements from object images that broadly categorized into animate and inanimate categories, with the aforementioned models using representational similarity analysis [77], and found that feature representations learned from a deep CNN (AlexNet) best explained the primate IT data, and also concluded that models that require supervision explain IT data better than the unsupervised models.

In another effort, Cadieu and colleagues ([31]) attempted to predict firing rates of IT neurons from two rhesus macaques (single and multi-unit), and compared to two deep CNNs (AlexNet, and [181]), the hierarchical modular optimization CNN (HMO) [177], an instantiation of the HMAX model [104], a V1-like model [119], and a V2-like model [51]. They used an image set of 1960 images with objects that are arbitrarily transformed in position, scale, and rotation, on a variety of backgrounds. With these images, single and multi-unit V4 and IT cortical neuronal firing rates were recorded from two macaques during passive fixation of each image, for 100 ms of presentation time. These recordings, presumed to constitute the primate neural representation for object recognition, was then compared to the models' feature representations of the same 1960 images using kernel analysis [101][24] and representational similarity analysis. The results indicate that the two deep CNN models are competitive in representational performances with the IT cortex, and outperforms V4 and all other models.

There are also works showing tremendous success in directly predicting the neuronal firing rates from the layers of a deep CNN, such as the use of AlexNet in [11], and the use of HMO CNN in [177]. While these recent studies have shown promising results in modeling the neuronal firing rates with deep CNNs, works that model behavioral data with deep CNNs have been lacking; furthermore, the prominent deep CNN model that has been utilized by the vast majority of these modeling studies is AlexNet, which was designed for the purpose of winning the ImageNet challenge and not for biological plausibility. In the next section, a deep CNN that is designed according to the human ventral stream neurophysiology is introduced (VsNet), along with a convolutional version of the HMAX model (Deep-HMAX), and applied to modelling categorical search performance by using them to extract CCFs. As a result, we show that deep CNNs that are more biologically plausible (VsNet and deep-HMAX) are able to predict human behavior better than the widely adopted AlexNet.

4.3 Ventral-stream convolutional neural network (VsNet)

Deep CNNs, AlexNet in particular, have been widely used recently as the model architecture for modeling various aspects of the primate ventral-stream as discussed previously. However, AlexNet was designed for the sole purpose of obtaining the lowest error rate in a large scale classification challenge (ILSVRC2012), as well as networks with different architectures such as the VGG net, GoogLeNet, and the ResNet. In the context of neuronal firing rate and/or behavior modeling, would these networks be as good as a network that's designed with a more biologically-informed architecture, given that they weren't designed with the brain in mind? We argue that in order to model aspects of the primate ventral-stream in object recognition, it is important for the model to be designed as closely to the current understanding of the ventral-stream as possible. In the following, we introduce a ventral-stream inspired deep CNN called the VsNet, that has five convolutional modules with the modules representing V1, V2, V4-like, inferior temporal areas TEO and TE, respectively, followed by three fully-connected layers at the top. Figure 4.3 shows the visualization of the VsNet architecture.

The architecture of the VsNet incorporates several aspects of the human ventral stream that have yet to be captured in existing ventral-stream models and deep CNNs. These ventral-stream aspects include: each VsNet convolutional module contains receptive field (RF) sizes that conform to each ventral-stream area; the units within each VsNet convolutional module contain a range of RF sizes, as the neurons within each ventral-stream area covers a range of RFs; bypass connections between convolutional models as according to the known ventral-stream neurophysiology; the V4-like module, formed by combining hV4 with Lateral Occipital area 1 and 2 (LO1/2), as LO1/2 have been shown to be selective to object contour, segmentation, and shape [83]; and the number of filters within each VsNet convolutional module, that correspond to estimated number of neurons with unique selectivities in each brain area.

4.3.1 Convolutional module receptive field sizes

In order to design each convolutional module of the VsNet to resemble the five major ventral-stream areas: V1, V2, hV4, TEO, and TE, the first task is to determine the overall RF sizes of each ventral-stream area from previous works, so that the sizes of the convolutional units within each VsNet convolutional module can be designed appropriately. Table 4.1 summarises several works that report RF-size estimates (in degrees) of each ventral-stream area, where numbers in bold are human estimates, and the non-bold numbers are from macaques. The RF sizes of each VsNet convolutional module are designed from the human estimates of Table 4.1, other than the TE module, as there are very few studies on measuring human TE RF sizes. Furthermore, while inferior temporal area TE is known to

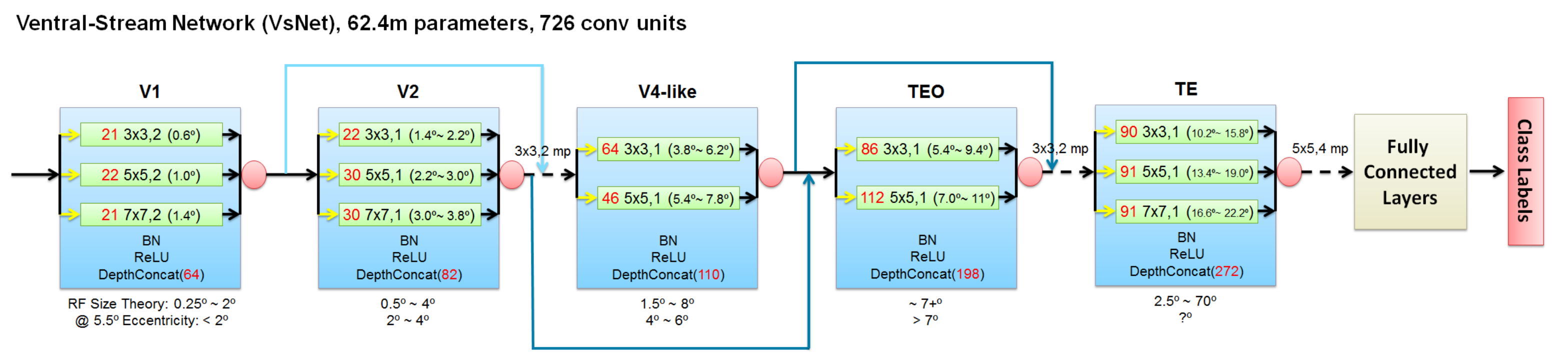


Figure 4.3. The visualization of our ventral-stream network (VsNet). Each convolutional module is colored in blue, with yellow arrows representing dimensionality reductions using 1×1 convolutional filters (similar to GoogLeNet's usage of 1×1 filters within each Inception module); green boxes representing within-module convolutional units and its specification of number of units (red), followed by the RF-size of the units (3×3, 5×5, or 7×7), followed by the stride size, ends with the receptive field size in terms of degrees (5 pixels $= 1°$); red circles represent concatenation at the depth dimension, and blue arrows represent bypass connections. Every convolutional module is also batch-normalized (BN, [63])and is followed by rectified-linear units (ReLU). Max-pooling is indicated as "mp" if performed at the end of a convolutional module, the specification convention is the same as the green convolutional blocks. VsNet has a total of 62.4 million parameters, comparable to AlexNet's 61 million; a total of 726 convolutional units (excluding the dimensionality reduction 1×1 units) compared to 1152 convolutional units in AlexNet (a more recent version [78]). The architecture of the fully-connected layers is the same as that of AlexNet's fully-connected layers.

	V1	V2	V4	TEO	TE
[23]	n/a	n/a	∼6.5	>7	>22
*[71]	**<2**	**2∼4**	**4∼6**	**>7**	n/a
[144]	**.25∼1**	**1∼2.5**	**2.5∼8**	n/a	n/a
[135]	.5∼1.5	.5∼4	1∼20	2∼25	2.5∼70
[59]	**.5∼1**	**.5∼1.5**	**1.5∼4.5**	n/a	n/a
[74]	∼1	∼1.4	∼4.8	∼5.8	∼12
[68]	n/a	n/a	n/a	**∼10**	**∼30**

Table 4.1. A summary of the RF-size estimates from select literatures, in terms of degree visual angle. Numbers in bold are human estimates, otherwise the estimates were taken from macaques. The * indicates that the RF-size estimates were taken at 5.5° eccentricity, while the others include a range of eccentricities, or pooled from different literatures of different of eccentricities.

have very large RF sizes in general, studies have found that TE neurons cover a large range of RF sizes that can be as small as a few degrees or as large as 30° or more [135][128]; [71] points out that the RF-size estimates in human visual cortex are strikingly similar to those in the homologous visual areas in monkeys, therefore we took the TE RF-size estimates from [135] as the TE convolutional module's RF-size constraint, for having the largest estimated RF-size range (first line of text under each convolutional module in Figure 4.1).

At another level of detail, neurons are known to be more densely populated at a smaller eccentricity, and that RF sizes tend to increase with eccentricity [51][165], therefore we also obtained the RF-size estimates of neurons in each ventral-stream area at 5.5° eccentricity, reported in [71], as basis for the within-convolutional module unit allocations (second line of text under each convolutional module in Figure 4.1). With these RF-size estimates from the literatures, we design the convolutional units within each module of the VsNet to have RF-sizes that conform with the largest range of human RF-size estimates of each area in Table 4.1, where all the RF-sizes are based on $1° = 5$ pixels. This is to allow sufficient flexibility in the possible filters that can be learned by each module of the VsNet.

4.3.2 Variable RF sizes within an area

The convolutional layers of most feed-forward deep CNNs contain filters of a single size [79][143][60][181], which is very restrictive and unlike the biological system, where neurons within each ventral-stream area range in RF size. The easiest way of making a convolutional layer contain filters of different RF sizes is to design a layer with parallel convolutional filters, such that a set of filters are a certain size i.e. 3×3, and another set of filters are of a different size i.e. 5×5. These filters convolve with the input and the output feature maps can be concatenated in the depth dimension, so that the convolutional units on the subsequent layer receive filter responses that come from a combination of different RF sizes. This design is very similar to GoogLeNet's inception module [150], as shown in Figure 4.1. [150] also mentions the importance of applying 1×1 filters at the beginning of each Inception module, which are beneficial for gathering responses that may have clusters of information [89], as well as an effective dimensionality reduction method for better memory conservation. Given the resemblance of our parallel units within each convolutional module, we also adopt the use of 1×1 as the first convolutional stage of each VsNet's convolutional module, shown as yellow arrows in Figure 4.3.

Our parallel convolutional units within each convolutional moduel consists of 3×3, 5×5, and/or 7×7 units, with strides two in the V1 module and stride one for the remaining four convolutional modules, where stride is the term that specifies the filter's step-size during a convolution. As shown in Figure 4.3, the RF-size of the parallel units in the V1 module are all of a fixed size, because they all receive the same RGB input image from the input layer; after the V1 module, the parallel units within the subsequent modules begin to cover an increasing range of RF sizes, due to the output of the V1 module being a combination of three RF sizes. 7×7 units and max-pooling operations are added selectively, as part of making the RF-size range of each VsNet convolutional module consistent with the homologous human ventral-stream areas.

4.3.3 Bypass connections

The neuroanatomy of the ventral-stream shows that in addition to the feed-forward projections, there are also bypass projections between different areas of the ventral-stream [74]. While the actual anatomical network is highly complex, there are major bypasses between area V2 and TEO, and V4 to TE [152], as well as weaker ones from the foveal region of V1 to V4 [56]. Given these evidences from the neuroanatomy of the ventral-stream, bypass connections are built into part of the VsNet architecture accordingly.

There are a total of three bypass connections for VsNet that includes V1 (the first convolutional module) to hV4+LO1+LO2 (the third), V2 (the second) to TEO (the fourth), and hV4+LO1+LO2 (the third) to TE (the fifth). As mentioned previously, we have two types of bypasses that include a weak bypass and a normal bypass. Normal bypass is done by simply making a copy out of the output (identity matrix) that is combined with the output of the bypassed module, where the combining operation is done by concatenating the two matrices along the depth dimension. For the weak bypass, the same copy of the output is obtained but an additional dimensionality reduction operation is performed using 1×1 filters to "weaken" the projected information. In the case of V1 to hV4, the matrix is reduced to half of its original dimension. It is worth noting that while most deep CNNs do not adopt any bypass connections between different convolutional layers, bypass connections are the major design contribution in the architecture of residual-learning network [60]. The major difference between the bypass connections of [60] and VsNet is how the current information is combined with the bypassed output: [60] combines the two by summing them together, and we combine the two by concatenating them along the depth dimension. The intuition for the two approaches differ in that summing causes residual learning to occur, while concatenating simply hands both information as is to the next processing stage. Since bypass connections in the biological sense seem to be more in line with passing information via shortcuts, we therefore implement our bypass connections as concatenations.

4.3.4 The V4-like convolutional module

While building the specifics of each human ventral-stream area into the architecture of a deep CNN is a non-trivial task, modeling the V4 part of the pathway is particularly challenging in several ways. The first major issue is that V4 is a ventral-stream area of monkeys, and numerous studies have attempted but were unable to reach a consensus on the location of the human homologous area (hV4) to that of V4 in monkeys [167][170][58][27]. Secondly, the measured surface area based on the assumed location of hV4 is very small (an important part for the next section), due to the conservative identification of the hV4 area boundary by some works [27][83].

As neurons in monkey V4 are known to be selective to color, shape, and boundary conformation [32], current identification of neurons from hV4 are mostly selective to color stimuli, but the more object related selectivity are found in the ventral-occipital area 1 and 2 (VO1/2), anterior to hV4 on the fusiform gyrus [27]. In another work, [83] found two higher level retinotopic visual areas, lateral occipital area 1 (LO1) and 2 (LO2) in the human lateral occipital cortex. While the authors argue that LO1/2 are two new visual areas that are not directly homologous to the macaque V4 area, they found LO1/2 to be essential parts of the object-selective process, and are selective to complex shape and boundary stimuli that are important for segmentation, grouping, region extraction, and boundary and border related extractions [83]. Furthermore, [161] found that deformed monkey V4 overlaps extensively with the human LOC area. Therefore, we combine hV4 and LO1/2 into a single V4-like module as the third convolutional module of the VsNet, as their biological counterparts are nearby anatomical areas with complementary visual selectivities.

4.3.5 Number of convolutional units per module

The final biologically-plausible design decision of the VsNet is the allocation of convolutional units per module, and we base this on the average surface area of each corresponding human ventral-stream area. We obtain the average anatomical surface areas from 15 subjects for V1, V2, hV4+LO1+LO (mm^2) as 2323, 2102, 2322 from [83]. We were unable to find the surface area estimates of human TEO and TE, therefore we attempt to estimate the surface area of TEO and TE from the monkey homologous areas with the following. First, [161] compared the surface areas of monkey and human and found an that they are generally larger in human than in monkey, but this enlargement is not uniform over all visual areas: human V1 and V2 are roughly twice larger, while human's entire cerebral cortex is about 10 times larger. This suggests that human and monkey shares similar evolutionary pathway with visual systems that process low-level features, but humans developed much more complex recognition ability from the much larger higher-level visual areas. Second, [117] states that the monkey posterior IT (PIT) area corresponds to the human TEO, and the monkey central and anterior IT (CIT/AIT) largely correspond to the human TE area; given the monkey PIT and CIT+AIT surface areas as 390 and 380 mm^2, respectively. Given the warping analysis from [161], we apply a multiplier of 9 toward both of these surface area measurements which results to 3510 and 3420 mm^2 as our human TEO and TE surface area estimates.

With these human ventral-stream surface area estimates, a naive approach would be to directly allocate convolutional units of each VsNet convolutional module in proportion with these estimates. However, the selectivity of early visual areas (i.e. V1 and V2) are much less spatially-invariant than the higher visual areas [32]; in other words, neurons with very similar selectivities are tiled across the visual field in V1 and V2 such that each neuron is specific to a spatial location within the visual input, while this tiling or "duplication" of neurons is much less in the higher level visual areas, which have integrated information from different spatial locations into a more object-centric processing. The tiling of similar neurons is in fact very similar to performing convolution over an image with a filter, just that the biological system uses duplicated neurons instead of performing convolution via sliding a filter across the image. Therefore, the number of convolutional units per layer corresponds to the number of uniquely selective filters within each convolutional module, and that requires both an estimate of anatomical surface areas, and the duplication factor of actual neurons of each ventral-stream area.

To estimate the duplication factor, we make a simple assumption: if a ventral-stream area's average RF size covers the entire visual field, then there is no need for any duplication of the neurons to capture all the information from the visual field, resulting in a duplication factor of 1; if an area's average RF size covers a quarter of the visual field, then there needs to be four of the same neurons in order to cover the visual field completely (duplication factor = 4). Therefore, the number of convolutional units per module can be determined by the following formulas:

$$\#\text{\_Conv\_Units} \propto \frac{\text{Surface\_Area}}{\text{Dup\_Factor}} \quad , \quad \text{Dup\_Factor} = log(\frac{\text{Visual\_Area}}{\text{RF\_Size}}) \tag{4.1}$$

Both visual field area and RF size values are in degrees squared, due to the the 2D input nature of images (width and height), and we take the log of the raw duplication factor in order to scale down the increment between areas (for hardware memory constraints) while keeping the same general trend. In our experiments, the inputs to the network are 224×224 random crops from the original image; with $1^\circ = 5$ pixels, the input roughly equals a $45^\circ \times 45^\circ$ visual area. Next, we take the average RF-size at 5.5° eccentricity of each ventral-stream area (1°, 3°, 5°, 7°, and 12° from [74]) for computing the duplication factors of each area. Then, plugging the surface area estimates obtained previously with these duplication factors, and then setting the first convolutional module (V1) to have 64 total units, we obtain our final VsNet architecture with 64, 82, 110, 198, and 272 convolutional units (excluding the 1×1 filters for dimensionality reduction) for each of the five convolutional modules, respectively. For the unit allocation of each parallel portion within each convolutional module, we allot more units to the ones with RF sizes closer in range to the actual RF-size estimates.

4.4 Deep-HMAX

As mentioned in Section 4.2, HMAX is a family of object recognition models that was designed to be biologically-plausible. For comparison purposes, we also designed a convolutional version of an instantiation of HMAX model, with the architectures closely following the specifications from [141] that we denote as Deep-HMAX. A baseline AlexNet ([78]) was also implemented as part of the model evaluations.

In porting HMAX into a CNN, all the simple and complex cells are replaced with convolutional layers, and the bypass connections that follow the same copy-and-concatenate operation as in VsNet. We took the liberty to design the number of units per layer and the corresponding RF sizes to somewhat resemble the original design in [141], that seems most reasonable and able to fit into the memory of a 12 Gb Titan-X GPU. Figure 4.4 shows the visualization of the architecture of Deep-HMAX, with the same naming format and conventions as Figure 4.3. The Deep-HMAX is compared to the VsNet in the next section.

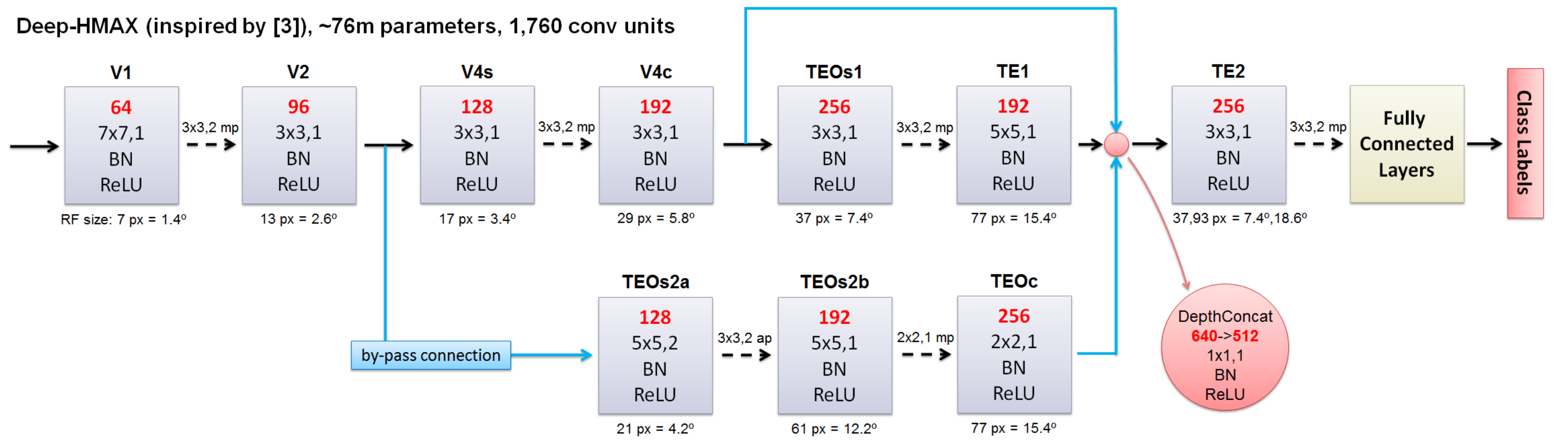


Figure 4.4. The architecture of the convolutional version of an HMAX instantiation [141], which we denote as Deep-HMAX. There are 10 serial and parallel convolutional layers that form a depth of 7 layers. All the simple and complex cells of the original HMAX are replaced with convolutional layers. Following the design principles of the HMAX idea, max-pooling operations are performed between each simple and complex cells. The fully-connected layers contains the same design as the fully-connected layers of AlexNet.

4.5 Model comparison

To evaluate VsNet, we compare VsNet to the convolutional version of the HMAX model that we refer to as Deep-HMAX, as well as the AlexNet (see Figure 4.5 for the alexnet architecture used in this experiment)that has been popularly used in recent computational neuroscience works [31][11]. The CNN models were implemented and trained using Torch7 on an Intel Xeon X3470 at 2.93 Ghz with 32 Gb, with a single Titan X GPU. The comparison is done under a representational similarity analysis (RSA) framework [76].

RSA is a framework for comparing the representations of models that can potentially be vastly different, such as a computational model to the brain recordings, by generating a representational dissimilarity matrix (RDM) for each model and correlating them. A RDM is a matrix containing dissimilarity values of a model's representation over all pairs of conditions, and the dissimilarity measure can be correlations, Euclidean distance, or some other distances depending on the type of model representation. In this case, we employ χ^2-distance as the dissimilarity measure for generating the RDMs for each of the 3 models in question.

Before we can obtain the representations of each model, the models are pretrained with ImageNet. For our baseline alexnet model, Figure 4.5 illustrates the architecture that we have employed for the experiments. The only difference between our baseline to the literature is the addition of batch normalization operation at the end of each convolutional layer, that has been shown to further reduce overfitting and drastically improve the training time [63]. In order to make all three models (AlexNet, Deep-HMAX, and VsNet) as comparable as possible, we allotted all models with 64 units at the first convolutional layer, and the same 256 units for the Deep-HMAX as AlexNet at the highest convolutional layer. As VsNet's convolutional unit allocations were computed automatically from Equation 4.1, we could not manually set the last module of VsNet to also contain a total of 256 units like the other models; interestingly, the resulting number of units of the last module turn out to be only slightly more than the other models (272 vs 256).

All three models are first trained using ImageNet as to obtain a reliable set of features. Then, the networks are fine-tuned with our hierarchical 68-category dataset, in order to be utilized for the subsequent CCF extraction step for modeling

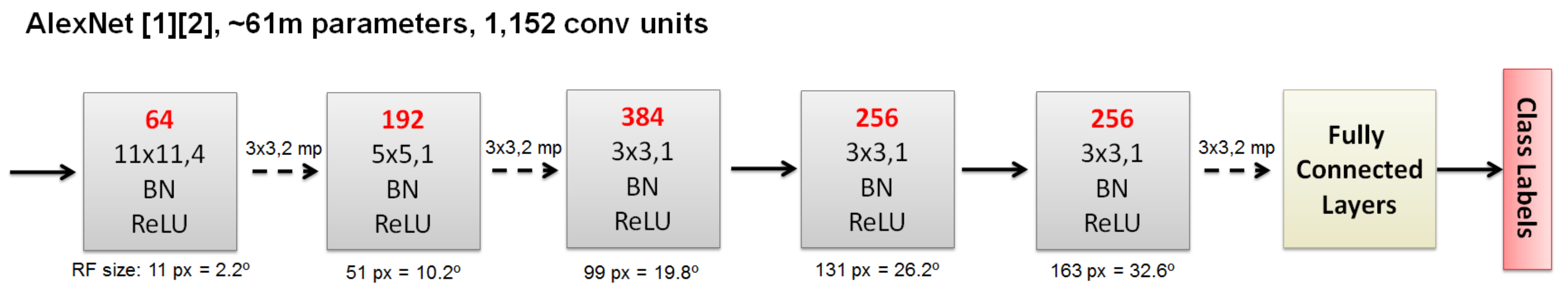


Figure 4.5. The architecture of the baseline AlexNet model [78]. The only difference from this model to [78] is the addition of batch-normalization at each convolutional layer. We also worked out the RF sizes in both pixels and degrees for each of the convolutional layer for comparison purposes.

categorical search performances. The fine-tuning step is formulated into a multi-task training setting, as there an object has class labels at the three hierarchical levels, i.e. passenger airplane, airplane, and vehicle.

4.5.1 ImageNet training and multi-task fine-tuning

The three networks were first trained with ImageNet's training and validation set (ILSVRC2012) for obtaining an initial set of features. We implemented the networks and performed all subsequent experiments using Torch7 framework [41], the He initialization method [60] was adopted for all networks, and utilized the training code from [9]. In summary, all training and validation images were previously resized to have the shortest side as 256 pixels, while keeping the original aspect ratio. The standard data augmentation methods of random crops (224×224) and random horizontal flips were employed. The center crop was use for computing the validation accuracies at the end of each training epoch. The training batch-size for AlexNet, Deep-HMAX, and VsNet was 128, 64, and 60, respectively. All training were done using 4-cores with image data stored on a SSD, and took roughly 4 days to complete 60 training epochs for all three networks.

Table 4.2 summarizes the validation set performances of the three models at the end of the training. As our focus of this experiment is not in achieving the highest possible ImageNet classification accuracy, we used a simple testing method of just the center 224×224 crops of the validation set images, in contrast to other works that obtain validation accuracies using an ensemble of models, multiple crops, and multi-resolution [143][150][60][181].

Table 4.2 summarizes the validation set performances of the three models at the end of the 60th training epoch. The results show that VsNet achieved the lowest errors and the baseline AlexNet had the highest errors. This is interesting because a deeper architecture has been shown outperform the shallower networks [143], but while Deep-HMAX is an improvement from AlexNet, it was worse than the shallower VsNet. Furthermore, VsNet has the least amount of total convolutional units compared to the other two models, meaning VsNet was able to produce the best representations with the least amount of learned filters. The improvements most certainly came from parts or all of the biologically-plausible design decisions

ImageNet	AlexNet	Deep-HMAX	VsNet
Top-1 Error	42.30%	40.32%	38.25%
Top-5 Error	19.36%	17.63%	16.08%

Table 4.2. Validation set performance for the three models at the end of the 60th training epoch. All results were obtained using the center 224×224 crops. While a deeper model such as Deep-HMAX was able to achieve better performance than AlexNet, VsNet turned out to perform the best even with just five convolutional modules and the least amount of convolutional units. This suggests that the bypass connections and the parallel units within each module that provides flexibility in RF-sizes helped significantly in a model's performance on image classification.

described in Section 4.3.1 through Section 4.3.5, and worth investigating into more details in the future. The results in Table 4.2 also indicates that this pre-training step of obtaining meaningful features were successfully performed, due to the validation errors that seem more than reasonable.

Next, we fine-tune the three networks with an expanded version of our hierarchical 68-category dataset that was previously introduced in Section 3.7.1. We obtained 1500 initial exemplars for each of the 48 subordinate-level categories from image search on google, yahoo, and bing. GIST descriptors ([112]) were then utilized for removing image duplicates, followed by manual pruning of the images to retain images that are cropped around the depicted object, and removed images with incorrect class labels. These operations resulted in a total of 550 exemplars for each of the 48 subordinate-level categories, where 500 of which are reserved as the training set and the remaining 50 are set aside as the validation set, adds up to a total of 24,000 training and 2,400 validation images. All images are resized such that the shortest side is 256 pixels wide with the original aspect ratio.

To fine-tune on the our extended hierarchical dataset, the models are updates such that they are trained under a multi-task learning regime, since an object contains three class labels at each of the three hierarchical levels, i.e. passenger airplane, airplane, and vehicle. The implementation of multi-task fine-tuning is straight-forward: given a pre-trained network, the highest fully-connected layer is replaced with three branches of newly initialized fully-connected layer, where the three branches connect to three output layers of 48, 16, and 4 classes, respectively. The same softmax layer and negative log-likelihood loss is employed. During a training forward pass, an input image would result into three losses from the three

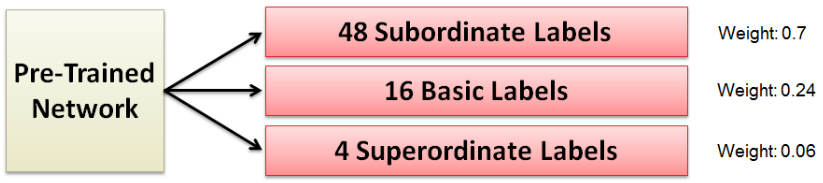


Figure 4.6. An illustration of the three output branches from a pre-trained network, under a multi-task learning setting.

68-Cat		AlexNet	Deep-HMAX	VsNet
Top-1 Error	Sub 48	8.62%	7.58%	6.58%
	Basic 16	4.46%	4.50%	3.71%
	Super 4	1.25%	1.33%	1.46%

Table 4.3. Validation set error for the three models at the end of the 25th training epoch, for the multi-task fine-tuning stage. Only top-1 errors are reported as the top-5 errors are near 0 for all models.

output branches, and we obtain the total loss by performing a weighted-sum with weights 0.7, 0.24, and 0.06 that correspond to the proportion of the categories at each branch. This final loss is then back-propagated for the gradient computation via the conventional back-propagation algorithm. Figure 4.6 illustrates the multi-task fine-tuning procedure with the three output branches.

The fine-tuning was performed with mostly the same training settings as the ImageNet pre-training step, but with just 25 total epochs and a reduced learning rate. Unlike the pre-training step, fine-tuning only took roughly 3 hours to finish on all three models. All models converged extremely well and the validation set performances are reported in Table 4.3, broken down into the three hierarchical levels. The results from Table 4.3 show that VsNet achieved the lowest top-1 error at the subordinate level, but not as good at the basic and superordinate level. However, the error differences between the models are very small which may not signify any indication from these results. The sure conclusion is that all models generalized near perfectly to the unseen validation set of the expanded 68-category dataset, and were successfully fine-tuned without any apparent over-fitting.

4.5.2 Representational similarity analysis

To generate the RDM of a given CNN model for the subsequent RSA step, [31] used the output of the last fully-connected layer of AlexNet (4096-dimensions) as the representation of a given input image. In our case, we would like to investigate the layer-wise differences between the 3 models, therefore we extract the convolutional kernels' responses to an input image instead of the responses from the last fully-connected layer. Given a hierarchical level (subordinate, basic, or superordinate) and an object category, a network's layer-wise representation is the averaged convolutional kernel responses of the layer over the set of images that belong to the category, and a kernel's response is determined as the sum over its feature map values. After obtaining a network's layer-wise representation to all object categories in the experiment, the network's RDM can be computed as the dissimilarity scores (χ^2 distance) of all pairs of object category L_1-normalized representations, such that the RDM has size 48×48, 16×16, and 4×4 for the subordinate, basic, and superordinate object categories, respectively.

Given the RDM generation procedure, a RDM is generated per each layer of a model at each object hierarchical level. Figures 4.7, 4.8, 4.9 show the RDMs for AlexNet, Deep-HMAX, and VsNet for the three hierarchical levels of object categories, respectively. Given that the dataset applied here is the hierarchical object category dataset that contains 48 subordinate, 16 basic, and 4 superordinate categories, the expectation is that the subordinate RDMs should show 4 big blue squares along the diagonal (the 4 superordinate groups), with 4 smaller and darker 3×3 blocks along the diagonal within each square (the 4 basic children categories each with 3 subordinate children categories). Referring to the actual RDMs from Figures 4.7, 4.8, 4.9, the figures show that as the information moves from the lower to the higher layers, the separation of the categories in groups become more visible, with the RDMs of the 5^{th} layer showing the expected block structures. One can also assess the layer-wise representational similarities across the three models by correlating pairs of models' RDMs. Table 4.4 shows the result of the pair-wise RDM similarities at the different object hierarchies, using Spearman's rank order correlation (Spearman's ρ). The table suggests that most layers of the three networks exhibit highly similar representations, except layer 4, showing Deep-HMAX at odds with the other two models. Another consistency that is shown

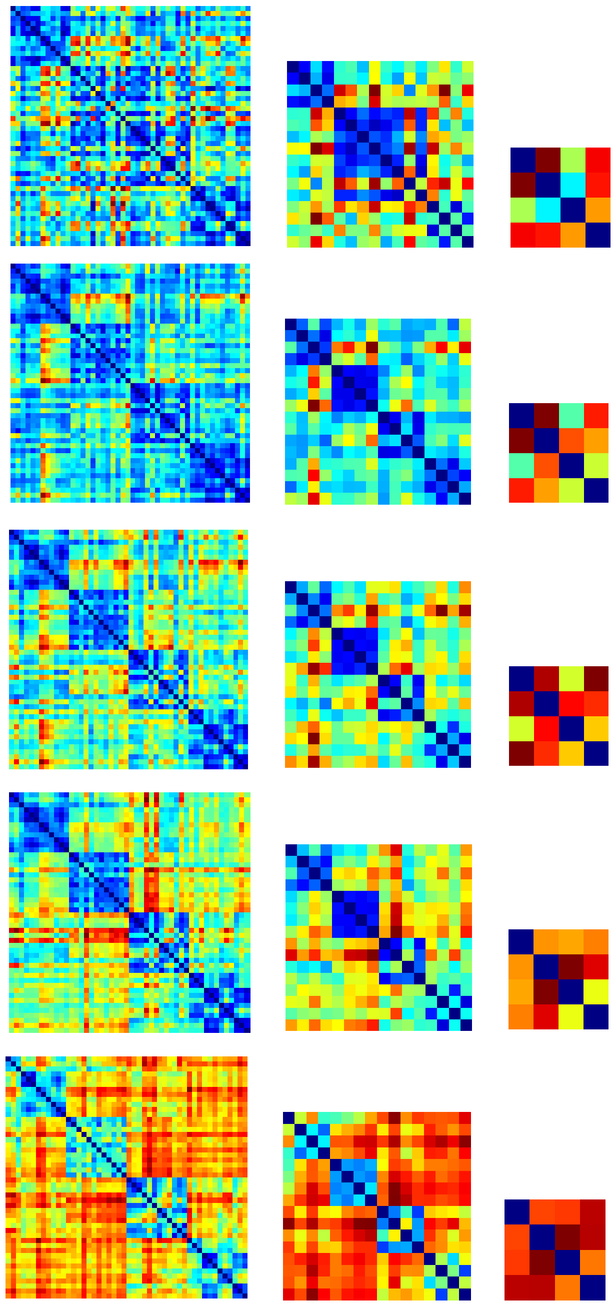


Figure 4.7. The representational dissimilarity matrices (RDMs) of the AlexNet model, over the hierarchical object category dataset. The rows are the RDMs out of individual layer's responses (from top to bottom: layer 1 to layer 5), and the columns are the RDMs of the different hierarchical levels (from left to right: subordinate, basic, and superordinate). The color indicates the strength of dissimilarity: the warmer (red) the color means the more dissimilar, and the colder (blue) the color means more similar.

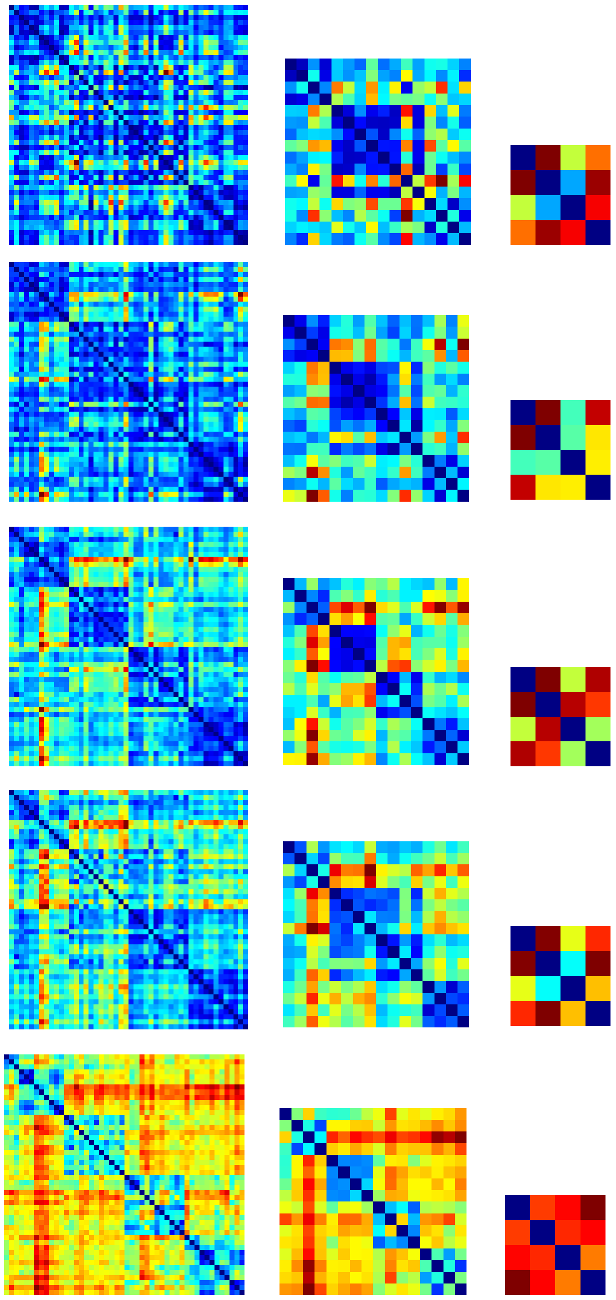


Figure 4.8. The representational dissimilarity matrices (RDMs) of the Deep-HMAX model, over the hierarchical object category dataset. The rows are the RDMs out of individual layer's responses (from top to bottom: v1, v2, v4c, TEOc, and TE2), and the columns are the RDMs of the different hierarchical levels (from left to right: subordinate, basic, and superordinate). The color indicates the strength of dissimilarity: the warmer (red) the color means the more dissimilar, and the colder (blue) the color means more similar.

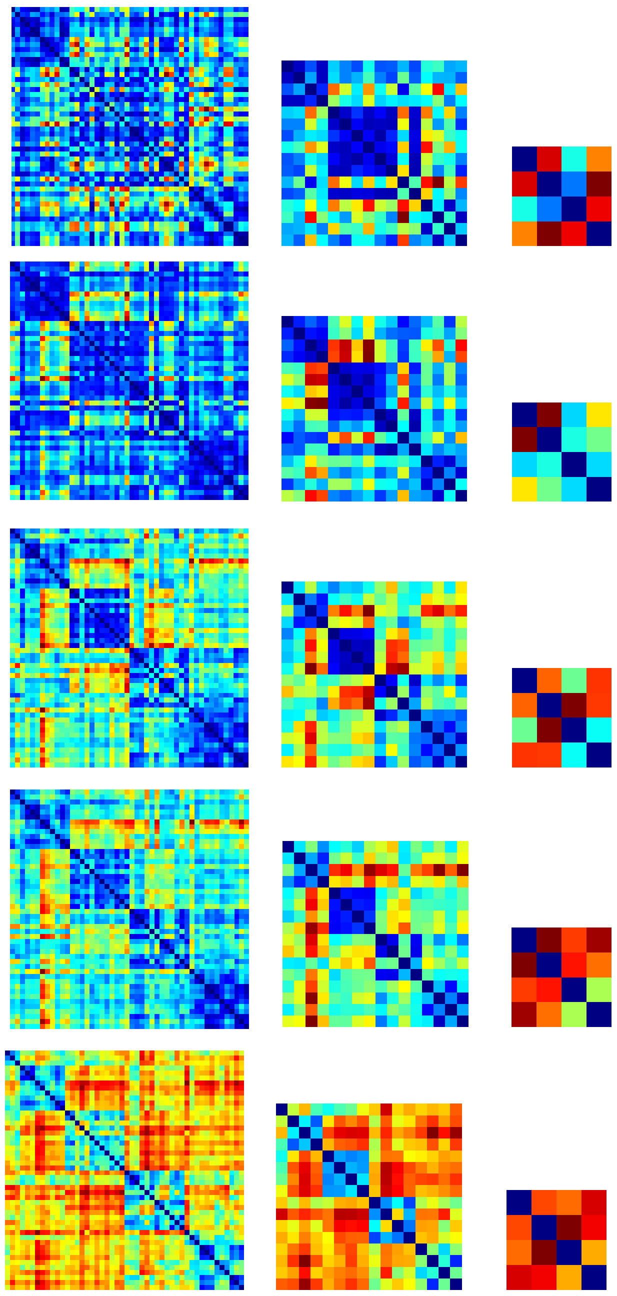


Figure 4.9. The representational dissimilarity matrices (RDMs) of the VsNet model, over the hierarchical object category dataset. The rows are the RDMs out of individual layer's responses (from top to bottom: layer 1 to layer 5), and the columns are the RDMs of the different hierarchical levels (from left to right: subordinate, basic, and superordinate). The color indicates the strength of dissimilarity: the warmer (red) the color means the more dissimilar, and the colder (blue) the color means more similar.

Layer 1		Subordinate	Basic	Superordinate
	AlexNet vs VsNet	0.899	0.947	0.714
	AlexNet vs Deep-HMAX	0.888	0.934	0.829
	Deep-HMAX vs VsNet	0.954	0.961	0.943
Layer 2				
	AlexNet vs VsNet	0.846	0.807	0.943
	AlexNet vs Deep-HMAX	0.843	0.806	0.829
	Deep-HMAX vs VsNet	0.892	0.883	0.943
Layer 3				
	AlexNet vs VsNet	0.885	0.893	0.657
	AlexNet vs Deep-HMAX	0.894	0.938	0.886
	Deep-HMAX vs VsNet	0.935	0.950	0.600
Layer 4				
	AlexNet vs VsNet	0.823	0.797	0.314
	AlexNet vs Deep-HMAX	0.506	0.460	0.029
	Deep-HMAX vs VsNet	0.658	0.709	0.086
Layer 5				
	AlexNet vs VsNet	0.931	0.919	0.886
	AlexNet vs Deep-HMAX	0.860	0.835	0.600
	Deep-HMAX vs VsNet	0.900	0.885	0.543

Table 4.4. Comparing the representational similarities between the layers of different models, in Spearman's ρ. The overall correlations are high, with layer 4 showing larger degree of differences between the three models.

from Table 4.4 is that Deep-HMAX in general has the highest correlation with VsNet, and that VsNet in general has the lowest correlation with AlexNet. This is interesting as both Deep-HMAX and VsNet were designed with the ventral-stream biology in mind, and that their more biologically-informed architectures result in representations that are more similar than the less biologically-informed AlexNet.

4.5.3 Filter visualizations

What are the learned features from ImageNet pre-training and the subsequent multi-task fine-tuing? The convolutional units of the first layer from all three models are shown in Figure 4.10. The visualizations show that AlexNet's features are sharp high-frequency gabor-like filters and dual-blob color features; Deep-HMAX's edge-related features are mostly entirely localized low-frequency gabor-like filters and uniform or spectrums of color features; VsNet's features non-color features

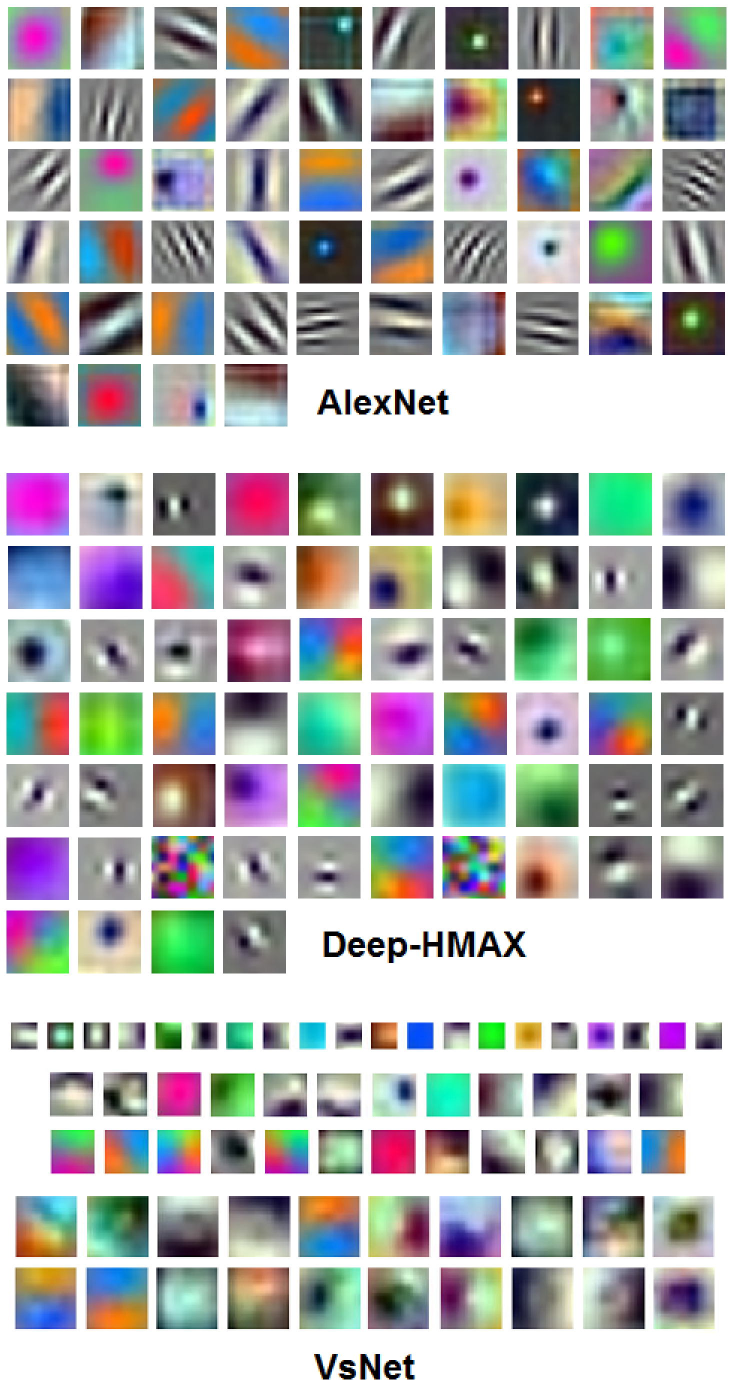


Figure 4.10. Feature visualization from the first convolutional layer of AlexNet, Deep-HMAX, and VsNet. Note that the features of AlexNet were resized from the original size of 11×11; Deep-HMAX features were resized from the original size of 7×7; the original sizes of the VsNet features were 3×3, 5×5, and 7×7. They were enlarged for better visualization purposes.

are even less indicative, they seem to show some edge-like filters but are mostly irregular looking, and the color features seem to be a mix of both AlexNet and Deep-HMAX.

Given the visualized features, the first apparent effect is that AlexNet's features are much sharper and well defined than both Deep-HMAX and VsNet. This suggests that neither Deep-HMAX nor VsNet have fully converged to its optimal model state, and the training regime for those two models can be adjusted and continued that will result in more well-defined and sharper filters and even better classification accuracies. Secondly, the features from the three networks seem to be very different from one another, especially with the non-color selective features. This is most likely due to the architecture differences, but further investigation is needed in order to conclude on the exact cause of the differences between the learned filters.

4.6 Discussion and conclusion

The recent deep learning popularity in both the field of machine learning have lead to a wide range of advances and applications, including computational neuroscience. The vast majority of the computational modeling efforts that utilize deep CNNs have been using AlexNet as the core model of choice [176][75][11], due to its success in ImageNet classification and its hierarchical structure that somewhat resembles the ventral-stream pathway. These efforts have also demonstrated that the primate IT responses under an object recognition task can be very well explained by aspects of AlexNet, while other non CNN models just partially explain the primate data. However, how far should one computational neuroscience go with AlexNet in terms of a model of the ventral-stream? While AlexNet's hierarchical structure is similar to that of the primate ventral-stream organization, AlexNet was designed for the purpose of winning the ImageNet 2012 classification competition, and therefore its model architecture was not designed with the goal of being biologically-plausible in any way. Furthermore, AlexNet's early success has lead to the computer vision community to have made several new CNNs with vastly different architectures with prime examples such as the VGG networks [143], GoogLeNet [150], ResNet [60], and more; should these models be considered as new ventral-stream models, since their image classification accuracies approach or have even surpassed human performance?

As the classification-based CNNs advance, there is no shortage of state-of-the-art models that can be borrowed by the field of computational neuroscience, for the investigation of how well the newer models can explain human data, but this is like studying a bird's aerodynamics and maneuverability with an ever faster jet airplane: it will always explain parts of the bird's movement very well, but not answer all of the questions because the design goals of the jet airplanes are just different from scientists who study the birds. To understand how a bird flies so well, the model of a bird should not be taken from the best jet airplane there is, it should be from utilizing the latest technology to build a model of the bird as closely as possible, and uncover the possible findings from the marriage of the two.

Motivated by the aforementioned bird-airplane analogy, that to model and understand the ventral-stream, we need to actually model the ventral-stream. Therefore, we have proposed a deep CNN with architecture details that were inspired by the human ventral-stream neuroanatomy, which we termed as VsNet. Interestingly, the resulting VsNet architecture seems to be a hybrid of GooLeNet and ResNet, due to the parallel units within VeNet's convolutional modules that are similar to the design of the Inception modules, and the bypass connections that are somewhat related to the bypass connections in ResNet.

In experiments, a baseline model of AlexNet, and a convolutional version of the HMAX model ([141]) that we termed Deep-HMAX were implemented for model comparisons. All three models were analyzed and compared under the framework of RSA, where the layer-wise representations of the three models were correlated and showed that VsNet is most similar to Deep-HMAX, and AlexNet is most different from VsNet. The results imply that models with biologically-informed architectures seem to behave more similarly in terms of layer-wise representations, as compared to the model with architecture that is less designed toward being biologically consistent. These interesting results suggest that building more biologically-informed design principles into the architecture of the latest CNN technology may be helpful in better modeling human data, and these models could potentially perform even better if more aspects of the neurophysiology are built into the next generations of CNNs.

Appendix A

Maximum Likelyhood Estimation and Non Linear Least Squares

A.1 MLE using Nelder-Mead algorithm

Given a nondegenerate simplex S as the convex hull of vertices $\{t_1, t_2, ..., t_{n+1}\} \in \mathbb{R}^n$, the Nelder-Mead method evaluates the objective function at each of the simplex vertices, the evaluation is denoted as $f(t_i)$, $i = 1, ..., n+1$, $f(t) = -\ln \mathcal{L}(t; \mathbf{x})$. For each iteration, the algorithm consists of the following three steps [109]:

1. Order the evaluation values at each vertex, such that $f(t_1) \leq f(t_2) \leq \cdots \leq f(t_{n+1})$.
2. Calculate the centroid of the best n points by $c = \sum_{i=1}^{n} t_i$.
3. Compute the new simplex by finding a new accepted point t' that leads to a better function evaluation. First, evaluate the *reflection* point $f(t_r)$ with respect to c. Iteration terminates if $f(t_1) \leq f(t_r) < f(t_n)$. Else:
   - *Expand* if reflected point is the new best point, such that $f(t_r) < f(t_1)$.
   - *Contract* if the reflected point is the second worse point, such that $f(x_n) \leq f(t_r)$.
   - *Shrink* the new simplex toward current best point t_1 if none of the above resulted in a better function evaluation, by replacing t_i with $\frac{1}{2}(t_1 + t_i)$, for $i = 2, ..., n+1$.

The effect of this algorithm is better understood in the case of $\mathbb{R}^2$, that the simplex is a triangle that flip-flops (reflection, if necessary) its way down the hill in the likelihood function space until convergence. To enforce the bound constraints of $\beta_{1,2} \geq 1$ and $\pi \leq 1$ that involves just inequalities, the accepted point t' is adjusted with the respective lower and upper bound if any of its corresponding parameter values fail to satisfy the constraints.

A.2 Nonlinear least squares

For observations $(x_1, y_1), ..., (x_n, y_n)$ and our parameter vector $\theta = (\alpha_1, \beta_1, c_1, \alpha_2, \beta_2, c_2, \pi)$, the least squares estimator finds the minimizer to the following objective function:

$$F(\theta; \mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\theta) \quad , \qquad r_i(\theta) = \mathcal{W}^2(\theta; x_i) - y_i \tag{A.1}$$

where $r_i(\theta)$ is the residual, and θ is subject to the bound constraints that were specified earlier. The trust-region method iteratively minimizes the objective function starting from an initial starting parameter vector θ'. By setting $\theta = \theta'$, we proceed to minimize a quadratic approximation $Q(s)$ that is the change in the objective function $F(\theta + s) - F(\theta)$. $Q(s)$ is given by:

$$Q(s) \equiv g^T s + \tfrac{1}{2} s^T H s \quad , \qquad \text{subject to} \quad ||s||_2 \leq \tau, \quad \tfrac{1}{\tau} - \tfrac{1}{||s||_2} = 0 \tag{A.2}$$

$s \in N$ is the subspace in the neighborhood N of the trust-region, g and H are the gradient and Hessian at θ. The Steihaug-Toint conjugate-gradient method can be used for each iteration step s with the unconstrained Newton equation $Hs = -g$, setting $\theta = \theta + s$ if $F(\theta + s) < F(\theta)$ and adjusting τ at the end of each iteration [147][154]. In our constrained case, the unconstrained Newton step is replaced with a scaled Newton step $D^{-2}g = 0$ to solve for the following linear system:

$$MDs^N = -g', \qquad M = D^{-1}HD^{-1} + diag(g)J^v \tag{A.3}$$

with $g' = D^{-1}g$ at the k^{th} iteration, and D is the diagonal matrix of vector $|v_k^{-1/2}|$, with J^v denoting the Jacobian of $|v|$. The bound constraints (ub: upper-bound, lb: lower-bound) are used here for computing $v(\theta)$: for the i^{th} observation, $v_i = \theta_i - ub_i$ if $g_i < 0$ and $ub_i < \infty$; $v_i = \theta_i - lb_i$ if $g_i \geq 0$ and $lb_i > -\infty$; $v_i = -1$ if $g_i < 0$ and $ub_i = \infty$; and $v_i = 1$ if $g_i \geq 0$ and $lb_i = -\infty$.

Bibliography

[1] The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`. Accessed: 18-Jun-2016.

[2] The kernel trick. `http://www.cs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf`, 2004. Accessed: 18-Jan-2016.

[3] Linear discriminant analysis. `http://www.tutorial.freehost7.com/human_face_recognition/linear_discriminant_analysis.htm`, 2008. Accessed: 18-Jan-2016.

[4] Cluster analysis. `http://astrostatistics.psu.edu/su09/lecturenotes/clus2.html`, 2009. Accessed: 18-Jan-2016.

[5] Mining for groups using clustering algorithms. `http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/mvoget/cluster/cluster.html`, 2010. Accessed: 18-Jan-2016.

[6] Gaussian mixture model. `http://blog.csdn.net/jwh_bupt/article/details/7663885`, 2012. Accessed: 18-Jan-2016.

[7] What is the difference between lda and pca for dimensionality reduction? `http://sebastianraschka.com/faq/docs/lda-vs-pca.html`, 2013. Accessed: 18-Jan-2016.

[8] Perceptron. `https://github.com/cazala/synaptic/wiki/Architect`, 2015. Accessed: 18-Jan-2016.

[9] Training an object classifier in torch-7 on multiple gpus over imagenet. `https://github.com/soumith/imagenet-multiGPU.torch`, 2015. Accessed: 24-Jun-2016.

[10] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on pattern analysis and machine intelligence*, 2012.

[11] P. Agrawal, D. Stansbury, J. Malik, and J. Gallant. Pixels to voxels: modeling visual representation in the human brain. *arXiv preprint arXiv:1407.5104*, 2014.

[12] J. R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 22:261–295, 1983.

[13] J. R. Anderson. Act: A simple theory of complex cognition. *American*

Psychologist, 51(4):355, 1996.

[14] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[15] F. G. Ashby and W. T. Maddox. Relations between prototype, exemplar, and decision bound models of categorization. *Journal of Mathematical Psychology*, 38:423–466, 1993.

[16] F. G. Ashby and W. T. Maddox. Human category learning. *Annu. Rev. Psychol*, 56:149–178, 2005.

[17] M. R. Beck and M. C. Lohrenz. Measuring search efficiency in complex visual search tasks: global and local clutter. *Journal of experimental psychology: applied*, 16(3):238–250, 2010.

[18] J. Bergen and M. Landy. Computational modeling of visual texture segregation. *Computational models of visual processing*, 1991.

[19] E. Bertin. Global fuctuations and gumbel statistics. *Physical Review Letters*, 2005.

[20] E. Bertin and M. Clusel. Generalised extreme value statistics and sum of correlated variables. *Journal of Pnysics A*, 2006.

[21] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[22] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *International conference on Image and video retrieval*, pages 401–408. ACM, 2007.

[23] D. Boussaoud, R. Desimone, and L. Ungerleider. Visual topography of area teo in the macaque. *Journal of comparative neurology*, 1991.

[24] M. L. Braun. Accurate error bounds for the eigenvalues of the kernel matrix. *Journal of Machine Learning Research*, 7:2303–2328, 2006.

[25] M. J. Bravo and H. Farid. Search for a category target in clutter. *Perception*, 2004.

[26] M. J. Bravo and H. Farid. A scale invariant measure of clutter. *Journal of Vision*, 2008.

[27] A. Brewer, J. Liu, A. Wade, and B. Wandell. Visual field maps and stimulus selectivity in human ventral occipital cortex. *Nature neuroscience*, 2005.

[28] C. Bundesen. A theory of visual attention. *Psychological Review*, 97(4):523–547, 1990.

[29] G. J. Burghouts, A. W. M. Smeulders, and J.-M. Geusebroek. The distribution family of similarity distances. In *Advances in Neural Information Processing Systems*, 2007.

[30] P. Burt and E. H. Adelson. The laplacian pyramid as compact image code. *IEEE Transactions on Communication*, COM-31:532–540, 1983.

[31] C. Cadieu, H. Hong, D. Yamins, N. Pinto, D. Ardila, E. Solomon, N. Majaj, and J. DeCarlo. Deep neural networks rival the representation of primate

it cortex for core visual object recognition. *PLOS Computational Biology*, 2014.

[32] C. Cadieu, M. Kouh, A. Pasupathy, C. Connor, M. Riesenhuber, and T. Poggio. A model of v4 shape selectivity and invariance. *Journal of neurophysiology*, 2007.

[33] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, 2010.

[34] M. S. Castelhano, A. Pollatsek, and K. Cave. Typicality aids search for an unspecified target, but only in identification, and not in attentional guidance. *Psychonomic Bulletin and Review*, 15:795–801, 2008.

[35] J. Chang, D. Wei, and J. Fisher. A video representation using temporal superpixels. In *CVPR*, 2013.

[36] A. Y. C. Chen and J. J. Corso. Propagating multi-class pixel labels throughout video frames. In *Western NY Image Processing Workshop*, 2010.

[37] S. Chin-Parker and B. H. Ross. Diagnosticity and prototypicality in category learning: A comparison of inference learning and classification learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30:216–226, 2004.

[38] C.I.E. *CIE Recommendations on Uniform Color Spaces, Colour-difference Equations, and Psychometric Colour Terms*, Paris: Bureau Central de la CIE, 1978.

[39] A. M. Collins and M. R. Quillian. Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior*, 8(2):240–247, 1969.

[40] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. pattern analysis and machine intelligence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, 2005.

[41] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLEarn, NIPS Workshop*, 2011.

[42] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5), 2002.

[43] J. J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *IEEE Transactions on Medical Imaging*, 2008.

[44] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, European Conference on Computer Vision*, 2004.

[45] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[46] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification.* John Wiley & Sons, 2012.

[47] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, M. Kruse, W. Munk, and H. J. Reitboeck. Coherent oscillations: A mechanism of feature linking in

the visual cortex? *Biological Cybernetics*, 60:121–130, 1988.

[48] P. F. Felzenszwalb and H. D. P. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 2004.

[49] P. E. Forssen. Low and medium level vision using channel representations. *Dissertation No. 858*, ISBN 91-7373-876-X, 2004.

[50] S. Franconeri, D. Bemis, and G. Alvarez. Number estimation relies on a set of segmented objects. *Cognition*, 2009.

[51] J. Freeman and E. P. Simoncelli. Metamers of the ventral stream. *Nature neuroscience*, 14(9):1195–1201, 2011.

[52] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *ICML*, 1996.

[53] J.-M. Geusebroek and A. W. Smeulders. A six-stimulus theory for stochastic texture. *IJCV*, 2005.

[54] R. L. Goldstone. The role of similarity in categorization: Providing a groundwork. *Cognition*, 52(2):125–157, 1994.

[55] D. Green and J. Swets. *Signal detection theory and psychophysics.* New York: Krieger, 1966.

[56] R. Greger and U. Windhorst. *Comprehensive human phyiology, Vol. 1: From cellular mechanisms to integration.* Springer, 1996.

[57] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010.

[58] K. Hansen, K. Kay, and J. Gallant. Topographic organization in and near human visual area v4. *Journal of neuroscience*, 2007.

[59] B. Harvey and S. Dumoulin. The relationship between cortical magnification factor and population receptive field size in human visual cortex: Constancies in cortical architecture. *Journal of neuroscience*, 2011.

[60] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[61] J. M. Henderson, M. Chanceaux, and T. J. Smith. The influence of clutter on real-world scene search: Evidence from search efficiency and eye movements. *Journal of Vision*, 2009.

[62] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. *CVPR*, 2007.

[63] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 2015.

[64] A. Ion, J. Carreira, and C. Sminchisescu. Image segmentation by figure-ground composition into maximal cliques. In *ICCV*, 2011.

[65] L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2001.

[66] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[67] A. J. Izenman. Recent developments in nonparametric density estimation. *Journal of the American Statistical Association*, 1991.

[68] M. Jenkin and L. Harris. *Vision and attention*. Springer, 2013.

[69] A. S. Kaplan and G. L. Murphy. Category learning with minimal prior knowledge. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26:829–846, 2000.

[70] J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schnorr. Globally optimal image partitioning by multicuts. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2011.

[71] S. Kastner, P. Weerd, M. Pinsk, M. Elizondo, R. Desimone, and L. Ungerleider. Modulation of sensory suppression: Implications for receptive field sizes in the human visual cortex. *Journal of neurophysiology*, 2001.

[72] S.-M. Khaligh-Razavi and N. Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS Computational Biology*, 10(11), 2014.

[73] S.-M. Khaligh-Razavi and N. Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS Computational Biology*, 2014.

[74] D. Kravitz, K. Saleem, C. Baker, L. Ungerleider, and M. Mishkin. The ventral visual pathway: an expanded neural framework for the processing of object quality. *Trends in cognitive sciences*, 2013.

[75] N. Kriegeskorte. Deep neural networks: A new framework for modeling biological vision and brain information processing. *Annual Review of Vision Science*, 1:417–446, 2015.

[76] N. Kriegeskorte, M. Mur, and P. Bandettini. Representational similarity analysis connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2008.

[77] N. Kriegeskorte, M. Mur, and P. A. Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4, 2008.

[78] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arVix:1404.5997*, 2014.

[79] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, 2012.

[80] J. K. Kruschke. Alcove: an exemplar-based connectionist model of category learning. *Psychological Review*, 99(1):22, 1992.

[81] K. J. Kurtz. Human category learning: Toward a broader explanatory account. *Psychology of Learning and Motivation*, 63:77–114, 2015.

[82] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal on Optimization*, 1998.

[83] J. Larsson and D. Heeger. Two retinotopic visual areas in human lateral occipital cortex. *Journal of neuroscience*, 2006.
[84] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[85] K. R. Levering and K. J. Kurtz. Observation versus classification in supervised category learning. *Memory & Cognition*, 43:266–282, 2014.
[86] E. Levina and P. Bickel. The earth mover's distance is the mallows distance: some insights from statistics. In *IEEE International Conference on Computer Vision*, 2001.
[87] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013.
[88] A. M. Liberman, K. S. Harris, H. S. Hoffman, and B. C. Griffith. The discrimination of speech sounds within and across phoneme boundaries. *Journal of Experimental Psychology*, 54(5):358, 1957.
[89] M. Lin, Q. Chen, and S. Yan. Network in network. In *International conference on learning representations*, 2014.
[90] T. Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 1998.
[91] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
[92] M. C. Lohrenz, J. G. Trafton, R. M. Beck, and M. L. Gendron. A model of clutter for complex, multivariate geospatial displays. *Human Factors*, 2009.
[93] B. C. Love, D. L. Medin, and T. M. Gureckis. Sustain: a network model of category learning. *Psychological Review*, 111(2):309, 2004.
[94] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
[95] M. L. Mack and A. Oliva. Computational estimation of visual complexity. In *the 12th Annual Object, Perception, Attention, and Memory Conference*, 2004.
[96] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.
[97] J. T. Maxfield, W. Stadler, and G. J. Zelinsky. The effects of target typicality on guidance and verification in categorical search. *Journal of Vision*, 13(9):524, 2014.
[98] J. T. Maxfield and G. J. Zelinsky. Searching through the hierarchy: How level of target categorization affects visual search. *Visual Cognition*, 20(10):1153–1163, 2012.
[99] D. L. Medin and M. M. Schaffer. Context theory of classification learning. *Psychological Review*, 85(3):207, 1978.
[100] C. B. Mervis and E. Rosch. Categorization of natural objects. *Annual Review*

of Psychology, 32(1):89–115, 1981.

[101] G. Montavon, M. L. Braun, and K.-R. MÃller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12:2563–2581, 2011.

[102] G. L. Murphy. *The big book of concepts*. MIT Press, 2002.

[103] G. L. Murphy and H. H. Brownell. Category differentiation in object recognition: Typicality constraints on the basic category advantage. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11:70–84, 1985.

[104] J. Mutch and D. Lowe. Object class recognition and localization using sparse features with limited receptive fields. *International journal of computer vision*, 80(1):45–57, 2008.

[105] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *International conference on machine learning*, 2010.

[106] M. B. Neider and G. J. Zelinsky. Exploring set-size effects in scenes: Identifying the objects of search. *Visual Cognition*, 2008.

[107] M. B. Neider and G. J. Zelinsky. Cutting through the clutter: searching for targets in evolving complex scenes. *Journal of Vision*, 2011.

[108] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 1965.

[109] J. Nocedal and S. Wright. Numerical optimization, series in operations research and financial engineering. *Springer, New York*, 2006.

[110] R. M. Nosofsky. Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, 115(1):39–57, 1986.

[111] R. M. Nosofsky and T. J. Palmeri. An exemplar-based random walk model of speeded classification. *Psychological Review*, 104(2):266, 1997.

[112] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[113] C. Olivers, J. Peters, H. Roos, and P. Roelfsema. Different states in visual working memory: when it guides attention and when it does not. *Trends in Cognitive Sciences*, 15:327–334, 2011.

[114] B. Olshausen, C. Anderson, and D. V. Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of neuroscience*, 1993.

[115] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *ECCV*, 2014.

[116] F. Orabona, G. Metta, and G. Sandini. A proto-object based visual attention model. *Attention in cognitive systems*, 4840:198–215, 2007.

[117] G. Orban, Q. Zhu, and W. Vanduffel. The transition in the ventral stream from feature to real-world entity representations. *Frontiers in psychology*, 2014.

[118] M. Pazzani. The influence of prior knowledge on concept acquisition: Experimental and computational results. *Journal of Experimental Psychology:*

Learning, Memory and Cognition, 17(3):416–432, 1991.

[119] N. Pinto, D. D. Cox, and J. J. DiCarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4(1), 2008.

[120] Z. W. Pylyshyn. Visual indexes, perconceptual objects, and situated vision. *Cognition*, 80(1-2):127–158, 2001.

[121] S. R. Rao, H. Mobahi, A. Y. Yang, S. Sastry, and Y. Ma. Natural image segmentation with adaptive texture and boundary encoding. In *ACCV*, 2009.

[122] T. Regier and P. Kay. Language, thought, and color: Whorf was half right. *Trends in Cognitive Sciences*, 13(10):439–446, 2009.

[123] R. Rensink and J. Enns. Preemption effects in visual search: evidence for low-level grouping. *Psychological Rewview*, 1995.

[124] R. A. Rensink. Seeing, sensing, and scrutinizing. *Vision Research*, 2000.

[125] R. A. Rensink. Seeing seeing. *Psyche*, 2010.

[126] R. A. Rensink, J. K. O'Regan, and J. J. Clark. To see or not to see: The need for attention to perceive changes in scenes. *Psychological Science*, 8(5):368–373, 1997.

[127] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.

[128] E. Rolls, N. Aggelopoulos, and F. Zheng. The receptive fields of inferior temporal cortex neurons in natural scenes. *Journal of neuroscience*, 2003.

[129] E. T. Rolls and T. Milward. A model of invariant object recognition in the visual system: learning rules, activation functions, lateral inhibition, and information-based performance measures. *Neural Computation*, 12(11):2547–2572, 2000.

[130] E. H. Rosch. Natural categories. *Cognitive Psychology*, 4(3):328–350, 1973.

[131] E. H. Rosch. *Principles of categorization*. MIT Press, 1978.

[132] E. H. Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8:382–439, 1976.

[133] R. Rosenholtz, Y. Li, J. Mansfield, and Z. Jin. Feature congestion, a measure of display clutter. *SIGCHI*, 2005.

[134] R. Rosenholtz, Y. Li, and L. Nakano. Measuring visual clutter. *Journal of Vision*, 2007.

[135] G. Rousselet, S. Thorpe, and M. Fabre-Thorpe. How parallel is visual processing in the ventral pathway? *Trends in Cognitive Sciences*, 2004.

[136] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *IEEE International Conference on Computer Vision*, 1998.

[137] J. C. Russ. *The image processing handbook*. CRC press, 2011.

[138] 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.

[139] J. Schmidt, A. MacNamara, G. H. Proudfit, and G. J. Zelinsky. More target

features in visual working memory leads to poorer search guidance: Evidence from contralateral delay activity. *Journal of Vision*, 14(3):8, 2014.

[140] J. Schmidt and G. J. Zelinsky. Search guidance is proportional to the categorical specificity of a target cue. *Quarterly Journal of Experimental Psychology*, 62:1904–1914, 2009.

[141] T. Serre, A. Oliva, and T. Poggio. A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences*, 104(15):6424–6429, 2007.

[142] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[143] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations*, 2015.

[144] A. Smith, A. Williams, and M. Greenlee. Estimating receptive field size from fmri data in human striate and extrastriate visual cortex. *Cerebral cortex*, 2001.

[145] J. G. Snodgrass and M. Vanderwart. A standardized set of 260 pictures: Normed for name agreement, image agreement, familiarity, and visual complexity. *Journal of Experimental Psychology: Human Learning and Memory*, 6:174–215, 1980.

[146] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[147] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 1983.

[148] E. B. Sudderth and M. I. Jordan. Shared segmentation of natural scenes using dependent pitman-yor processes. In *Neural Information Processing Systems*, 2008.

[149] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

[150] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, and S. Reed. Going deeper with convolutions. In *IEEE Conference on computer vision and Pattern Recognition*, 2015.

[151] J. W. Tanaka and M. Taylor. Object categories and expertise: Is the basic level in the eye of the beholder? *Cognitive Psychology*, 23:457–482, 1991.

[152] K. Tanaka. Mechanisms of visual object recognition: Monkey and human studies. *Current opinion in neurobiology*, 1997.

[153] M. Tarr. News on views: pandemonium revisited. *Nature neuroscience*, 2:932–934, 1999.

[154] P. L. Toint. Towards an efficient sparsity exploiting newton method for minimization. *Sparse Matrices and Their Uses*, 1981.

[155] D. Tsai, M. Flagg, and J. Rehg. Motion coherent tracking with multi-label mrf optimization. In *BMVC*, 2010.

[156] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature neuroscience*, 5(7):682–687, 2002.

[157] I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *Computer Vision and Pattern Recognition*, 2005.

[158] J. Van De Weijer and C. Schmid. Coloring local feature extraction. In *European Conference on Computer Vision*, 2006.

[159] R. van den Berg, F. W. Cornelissen, and J. B. T. M. Roerdink. A crowding model of visual clutter. *Journal of Vision*, 2009.

[160] R. van den Berg, J. B. T. M. Roerdink, and F. W. Cornelissen. On the generality of crowding: Visual crowding in size, saturation, and hue compared to orientation. *Journal of Vision*, 7(2):14, 1–11, 2007.

[161] D. Van Essen, J. Lewis, H. Drury, N. Hadjikhani, R. Tootell, M. Bakircioglu, and M. Miller. Mapping visual cortex in monkeys and humans using surface-based atlases. *Vision research*, 2001.

[162] A. Vazquez-Reina, S. Avidan, H. Pfister, and E. Miller. Multiple hypothesis video segmentation from superpixel flows. In *ECCV*, 2010.

[163] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *European Conference on Computer Vision*, 2010.

[164] D. Walther and C. Koch. Modeling attention to salient proto-objects. *Neural Networks*, 2006.

[165] B. Wandell and J. Winawer. Computational neuroimaging and population receptive fields. *Trends in cognitive sciences*, 19(6):349–357, 2015.

[166] J. Wang, Y. Jia, X.-S. Hua, C. Zhang, and L. Quan. Normalized tree partitioning for image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[167] J. Winawer and N. Witthoft. Human v4 and ventral occipital retinotopic maps. *Visual neuroscience*, 2015.

[168] M. Wischnewski, A. Belardinelli, W. X. Schneider, and J. J. Steil. Where to look next? combining static and dynamic proto-objects in a tva-based model of visual attention. *Cognitive Computation*, 2010.

[169] M. Wischnewski, J. J. Steil, L. Kehrer, and W. X. Schneider. Integrating inhomogeneous processing and proto-object formation in a computational model of visual attention. *Human centered robot systems*, 6:93–102, 2009.

[170] N. Witthoft, M. Nguyen, G. Golarai, K. LaRocque, A. Liberman, M. Smith, and K. Grill-Spector. Where is human v4? predicting the location of hv4 and vo1 from cortical folding. *Cerebral cortex*, 2014.

[171] J. M. Wolfe. Visual search. *Attention*, 1998.

[172] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[173] C. Xu and J. Corso. Evaluation of super-voxel methods for early video

processing. In *CVPR*, 2012.

[174] C. Xu, S. Whitt, and J. J. Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *ICCV*, 2013.

[175] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012.

[176] D. L. Yamins and J. J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

[177] D. L. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.

[178] A. Y. Yang, J. Wright, Y.Ma, and S. Sastry. Unsupervised segmentation of natural images via lossy data compression. *CVIU*, 2008.

[179] V. Yanulevskaya and J.-M. Geusebroek. Significance of the Weibull distribution and its sub-models in natural image statistics. In *Int. Conference on Computer Vision Theory and Applications*, 2009.

[180] Z. Yu, O. C. Au, K. Tang, and C. Xu. Nonparametric density estimation on a graph: Learning framework, fast approximation and application in image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[181] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, 2014.

[182] G. J. Zelinsky, H. Adeli, Y. Peng, and D. Samaras. Modelling eye movements in a categorical search task. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 368(1628), 2013.

[183] G. J. Zelinsky and J. W. Bisley. The what, where, and why of priority maps and their interactions with visual working memory. *Annals of the New York Academy of Sciences*, 1339:154–164, 2015.

[184] G. J. Zelinsky, Y. Peng, A. C. Berg, and D. Samaras. Modeling guidance and recognition in categorical search: Bridging human and computer object detection. *Journal of Vision*, 13(3):30, 2013.

[185] G. J. Zelinsky, Y. Peng, and D. Samaras. Eye can read your mind: Decoding gaze fixations to reveal categorical search targets. *Journal of Vision*, 13(14):10, 2013.