Perception Strategies in Hierarchical Vision Systems

Lior Wolf Stan Bileschi Ethan Meyers
Center for Biological and Computational Learning
The McGovern Institute for Brain Research
Massachusetts Institute of Technology
Cambridge, MA
{liorwolf,bileschi,emeyers}@mit.edu

Abstract

Flat appearance-based systems, which combine clever image representations with standard classifiers, might be the most effective way to recognize objects using current technologies. In the future, however, it seems probable that hierarchical representations might have better performance. In such systems, the image representation consists of a sequence of sets of features, where each subsequent set is computed based on the previous sets.

The main contributions of this paper are to: (1) pose the question "what is the best way to employ discriminative methods for hierarchical image representations?"; (2) enumerate some of the alternative hierarchies while drawing connections to recent work by brain researchers; (3) study experimentally the different alternatives. As we will show, the strategy used can make a substantial difference.

1. Introduction

To date, there is no satisfactory explanation for the role of feedback connections in the primate visual system. These abundant feedback connections dominate the visual areas in a 10 to 1 ratio over the feed-forward connections [8]. A clearer understanding of their function could give insights into how to build better computer vision systems.

Theories for the role of feedback connections have usually focused on attentional mechanisms and hypothesis verification loops. In Hawkins [14], it is proposed that feedback connections are involved in prediction-verification recursions wherein the generation of a predicted representation (top-down) is matched with the upcoming representation (bottom-up). Computational implementations of these hypothesis-verification theories have had limited success when compared to appearance-based recognition systems, although some exceptions exist (*e.g.* [6]).

In Reverse Hierarchy Theory (RHT), proposed by Hochstein and Ahissar [16], information is first processed automatically bottom-up, while perception progresses in the opposite direction. RHT can account for many seemingly disparate psychological phenomena found in humans, such as the different modes of visual search, and the properties of vision at a glance compared to vision with scrutiny. However, in biological systems, verifying this or any other feedback theories remains extremely difficult.

In computational systems, feedback is often used in the training phase of neural networks and graphical models, however, the authors are unaware of any work showing how feedback can be useful for modern discriminative object recognition systems, such as those employing SVM or AdaBoost for classification. Currently, these discriminative systems seem to outperform other technologies but still fall far behind human performance. Adding feedback is one promising direction to close the gap.

Here we create computational implementations of different hierarchal architectures to see if an improvement can be made over the performance of simpler discriminative object classification systems. In our framework, we equate the cognitive notion of perception (from RHT) with the computational output of classifiers trained to respond strongly to that class. We describe five computational hierarchies and experimentally evaluate their relative performance.

2. Background: Hierarchical vision systems

Before appearance-based models became standard practice, hierarchal representations were common in computer vision. Objects were often represented as compositional hierarchies of atomic forms, and object searches were performed by detecting for appropriate groups of these atoms in oftentimes combinatorial sized search spaces (*e.g.* [20, 11]). With the advent of advanced statistical learning tools and more powerful computers, the best performances were next reported by shallow architectures using learning directly on top of relatively simple image representations.

Recently, hierarchal systems, now with multiple levels of learning, have regained their popularity (*e.g.* [18, 23]).

Along with the additional power and flexibility available to hierarchal systems come problems inherent to their design. It remains unknown how best to factor the vision problem: should object parts be represented explicitly, at what stages should learning be employed, etc. In [28], a generative model was trained using both the appearance of target objects, and the gist of the scenes in which these objects were likely to appear. While factoring the conditional of a generative model by layers in a hierarchal representation is a well defined problem, the best way to combine these layers in a discriminative framework is less clear.

Part-based systems These are hierarchal architectures which explicitly represent spatially localized structures related to the task. For instance, the part-based face detector detailed by Heisele et. al. [15] employs detectors for the eyes, nose and mouth. The part based system of Ullman et. al. [30] is similar in that sub-parts of the image are represented explicitly by independent detectors within the system. However, in this system, instead of hand selection, mutual information is used to select which object parts will be represented. Other part-based object detection systems which automatically learn, via clustering or otherwise, which object parts to explicitly represent include [31] and [19]. In [4], feedback is used in order to resolve ambiguities in the part detections by considering inter-part relations.

Classifiers as features Several systems have used discriminative classifier outputs as learned intermediate visual features, *e.g.* [26, 1]. The intuition is that classifiers trained to perform tasks in one domain may learn to implicitly represent patterns and invariances in a host of related domains. The subsequent use of these classifier outputs as input to a higher level of the hierarchy allows for the transfer domain knowledge. Another option is to learn an intermediate level representation without label information. In this case, it is necessary to use an unsupervised algorithm, such as clustering or parametric models, as in [19, 17] or [31].

The convolutional network The system of LeCun et. al. [18] is an example of a multi-layered neural network designed and trained via back-propagation to perform object detection and recognition tasks. It can be seen as a type of visual hierarchy in which learning occurs at each level. Rather than explicitly representing parts, a hierarchal object representation is induced through the clever network architecture. Although feedback is used during the training phase, during testing the convolutional network is completely feed-forward, leading to very rapid performance.

The standard model This system provides a quantitative model of the feed-forward pathway of the ventral stream in visual cortex. Recently, a machine vision implementation of the architecture was tested on real-world image databases, showing promising results [24, 3]. So far, the modeling effort has focused on the feed-forward architecture of the visual stream. Strong evidence suggests that feed-forward pathways are responsible for the first 100 msec of visual perception [21, 25] and that the basic steps for recognition are completed in this time, including tuned responses of neurons in IT. Of course, a complete model of vertebrate vision must take into account image sequences, as well as top-down signals, attentional effects and the structures mediating them (*e.g.* the extensive back-projections present throughout cortex).

Reverse Hierarchies Hochstein and Ahissar's Reverse Hierarchy Theory, (RHT) [16] proposes that visual information initially travels through the feed-forward visual hierarchy, and that perception begins at the higher levels, reaching the lower areas via feedback connections, forming a reverse hierarchy. They point to three lines of evidence to support this theory. Firstly, the gist of a scene, consisting of broad category information, is consciously perceived first, and detailed information is only consciously perceived later when vision with scrutiny is engaged. The ability to perceive the contents of rapidly presented images, as seen in the Rapid Serial Visual Presentation (RSVP) paradigm [22], while at the same time being blind to details of the images, as seen in repetition blindness and change blindness phenomena, is further support. Since the receptive field sizes needed to process the gist of a scene are only seen in neurons of higher visual areas, while cells coding for precise details are found in lower visual areas, these phenomena suggest that perception starts at the highest levels and travels backward to the lower levels. The second piece of evidence comes from the visual search research, where it has been well established that there are two modes of search, one mode that is rapid and parallel and one mode that is slower and serial [29]. The rapid parallel detection of objects often uses complex features such as circles and faces that are only available in higher visual areas, suggesting again that high level areas have consciousness first [27]. The final line of evidence comes from visual learning research, where it has been shown that visual tasks that are learned more rapidly tend to generalize to other similar spatial stimulus conditions, while harder tasks that take a long time to learn tend not to generalize outside of the specific conditions where they are learned [12]. Hochstein and Ahissar hypothesize that learning that has greater generalization occurs in neurons with large receptive fields and thus must be occurring in higher level areas, and since this generalizable learning occurs earlier than specific learning, the authors conclude that learning occurs in higher areas first.

3. Alternative classification strategies

Given a hierarchical image representation, there are several alternative ways it can be classified. If each level of representation is given as a vector, one can concatenate all

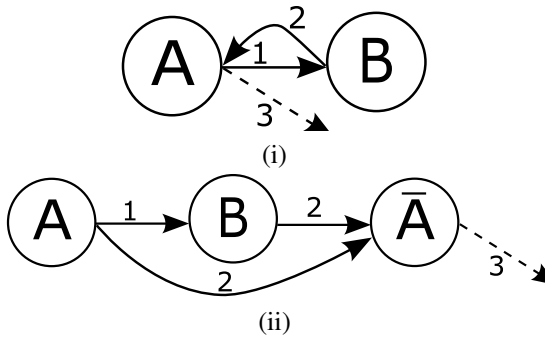


Figure 1. (i) A simple feedback system. (ii) The unfolded version of the same system. Since we are not concerned with the actual location of the computation we view system (i) and system (ii) as being equivalent.

of the representations to a single long vector. Another option is to classify each level separately and then combine the classification results. Below we will name and describe five such options. In our experiments we focus on two level hierarchies since modern representations with more than two levels are rare. It is straight-forward to enumerate more strategies for hierarchies with three or more levels. Below, we clarify some of the basic concepts we use.

High level vs. low level Strictly speaking, representation B will be higher in the hierarchy than representation A if computing B requires computing A. When processing takes place information is lost. Hence, it is always the case that higher levels of hierarchies are less informative than lower levels. However, these representations are often more explicit, resulting in a more straightforward classification. On the other hand, simply concatenating several representations will always result in no less information than any individual representation. Yet, it is the ease in which the classifier can access the relevant information that determines the overall accuracy, and having many irrelevant variables can diminish performance.

Feedback In a *simple feedforward system*, each level of the hierarchy is used only to produce the next level. Once level $l-1$ produces level l, its role is completed, and it is never used again. In systems that employ feedback, information from higher levels is passed back to lower levels and the combination of both is then being considered.

An example for a feedback system might be a system where representation A produces image representation B, which in turn through some process updates representation A. Finally, further processing is applied and an output is produced (see figure 1(i)). In this work we do not consider the location in which the feedback takes place. Instead we separate the initial representation (A) from the later representation ($\bar{A}$). Our view of the system would resemble figure 1(ii): representation A is computed, and representation B is computed from it. Representation $\bar{A}$ is then computed from A and from B, more processing is then done to produce the final system's output.

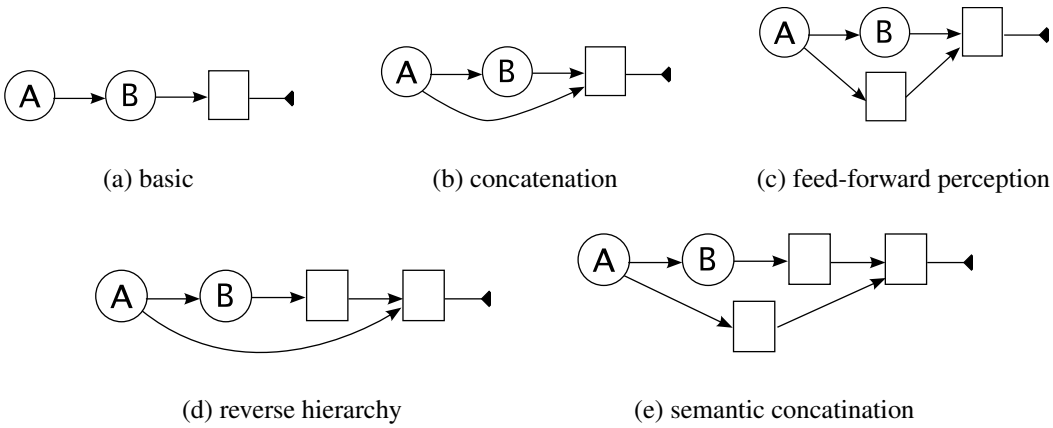
We assume that the feedback system and the unfolded system are equivalent. We describe our systems as unrolled systems, which resemble feed-forward systems. However, these system are not simple feed-forward systems (as defined above), and since they can be implemented as feedback systems, they can be studied as a model of these.

One can imagine the image representation $\bar{A}$ to be similar in nature to A, yet improved by incorporating higher level information that is available in representation B. For example, A might contain edge information, and by employing higher level information that is available in representation B, closed contours are being emphasized in $\bar{A}$. In a more elaborate example, B is classified (*e.g.* in a car detection system – to a car or not a car) and the results of this classification are used to emphasize particular properties in A (such as the object boundaries in the car example).

We have conducted experiments akin to these two examples. Unfortunately, the result only demonstrated the difficulties in implementing a robust feedback system. In one experiment we tried to emphasize elements of the $C1$ image representation [24] that contribute to long continuous boundaries as detected by our continuity detector [5]. The hope was that the modified $C1$ would have better recognition capabilities than the original $C1$. In another experiment, we tried to emphasize the most informative $C1$ elements in a car detection task using feedback. For this experiment, we used a boosting classifier on-top of weak classifiers, each corresponding to a single $C1$ unit. The units which contributed positively to the classification were made larger in value and projected back to the original edge map (which is called S1 in [24]). Finally, a new $C1$ representation was then computed. Both of these experiments produced disappointing results: the modified $C1$ did not significantly outperform the original. Perhaps, with further study better results could be achieved.

In this paper, we are much more restricted in the kind of systems we allow. We assume that the modified representation $\bar{A}$ is to be delivered to a classifier. Since we do not know how to properly modify A to create $\bar{A}$, we build a classifier directly on top of the inputs that contribute to $\bar{A}$. Hence, if $\bar{A}$ is computed based on A and B, instead of classifying $\bar{A}$, we train a classifier directly on top of a combination of A and B. Since our classifiers are limited in power, and since the number of training examples is not high, such a solution is not optimal. However, it does bypass the need to engineer a modified representation explicitly, which is a task for which our technological capabilities are limited.

Perception Another simplification that we employ is the equating of perception with classifier-output. Since perception is computed from the data it is worthwhile discriminating between processing (going up the representation hi-



(a) basic (b) concatenation (c) feed-forward perception

(d) reverse hierarchy (e) semantic concatination

Figure 2. These figures illustrate alternative hierarchy architectures (this is not a full list). A circle represents an image representations, a square represents classification, the small full triangle represents the output. For example, Ⓐ can refer to the C1 image representation, while Ⓑ can refer to the higher level C2, a square can represent the output (raw distances from hyperplanes) of a 102 one-vs-all linear SVM classifier trained on the 101 objects (plus background) dataset. (a) Basic feedforward architecture. (b) Concatenation of the layers followed by a classifier. (c) A feedforward perception architecture in which the perception of the lower level representation is used in combination with the higher level representation to create the final perception. (d) Our interpretation of the reverse hierarchies [16]. The perception of the high level features is classified together with the low level features to create the final perception. (e) The semantic concatenation architecture, in which the final perception is a result of combining the perceptions of all of the previous levels together.

erarchies) and perception. In our terminology, perception is a type of computation which is learned from the data and which produces information which is aimed at being directly indicative of the objects' identity. In other words, through training, we expect the perception vector to be very highly correlated with the label information. In figure 2, representations are marked as circles, while perceptions are marked as boxes.

For multiple class data sets, we employ one vs. all architectures and the perception vector is as large as the number of classes. Note that we record the raw classifier output (a real number) rather than a discrete label. This "perception" can then be combined with other perceptions or with vector image representations simply by concatenation. For binary tasks, we create many classifiers, each using only part of the training data, and use the output of all of these classifiers to create a long perception vector (somewhat similarly to random forests [7]).

A crucial point is that for some hierarchies we train a first set of classifiers to generate perceptions and then train another set of classifiers on top of the output of the first set. Ideally, we would use one set of training examples to compute the first set of classifiers, and another set of training examples to compute the second set. In our experiments we do not use a separate training set for the following reasons: (1) when data are expensive, splitting the training set is wasteful; (2) Since we try to check if classification of classifiers' output is beneficial on a separate test set, the diminished performance one gets by using the same training data twice, can only lead to a more conservative conclusion; (3) We use classifiers with good generalization ability, limiting the effect of reusing training data.

3.1. Five alternative strategies

Below we present five alternatives strategies for classifying the information in a hierarchical representation. These alternatives are illustrated in figure 2.

basic (a) The most basic strategy to classify a hierarchical image representation is to simply classify one of the levels of the hierarchy. Usually, one would classify the highest level available [24]. Another alternative is to use the level in the hierarchy that produces the best results.

concatenation (b) Another simple strategy is to concatenate the features from all of the layers into one long vector, and use this vector for classification. This is the method used in [5] and it seems to be quite effective.

feedforward perception (c) By feedforward perception we simply mean that the perception of the low level image description is used in concert with the raw high-level description to generate the final classification. An example for such a system would be a car detection system which employs edge information along side with edge statistics. The edge information would be classified to be a car boundary or not, and this boundary perception would be incorporated with the edge statistics of a window to generate the final classification. In our experiments, we compare to a simplified version where the low level perception are global to the entire window (not one per pixel). This allows us to make direct comparisons between different hierarchies, without

biasing the results by searching the large space of parameters which a more engineered solution requires.

reverse hierarchy (d) RHT asserts that higher level representations are being perceived before low level ones. The initial perception is based on the rough high level information (or gist). This perception is verified while details are filled in by a second perception process that takes into account the initial perception and the low level representation.

In our experiments, we concentrate on classification accuracy, and not on the fine details of recognizing the object (for example, we do not try to recognize the object's boundaries or the parts of the object). This enables us to establish common performance measurements between the varying strategies. The high level feature is being classified to create a perception vector which is concatenated to the low level image representation and then classified.

semantic concatenation (e) This strategy is usually referred to as stacking [32]. Here the perception vectors are computed for each level of the hierarchy separately, and are then combined by a simple concatenation. The concatenated vector is classified to produce the final output.

This simple strategy resembles voting, but with important differences. First, while simple voting cannot take place between two classifiers, semantic concatenation is successful in such cases. Second, by our definition of perception, the classifier does not output a label but rather a real valued number. Similar to some voting techniques, in the final decision each element of the perception vectors is weighted differently.

4. Experimental study

We cannot offer a theory that can predict which hierarchy is the most appropriate for any given task. Similar to other situations where one has to choose between learning algorithms/parameters, performance measures on a validation data set might be the prevailing indicator. Below we provide experimental results on several vision data sets. These experiments show that the type of hierarchy used may have a large effect on the resulting accuracy.

In the experiments reported, when combining feature matrices X_1 and X_2, we normalize matrix X_2 by multiplying it by a constant such that the matrix norms of the kernel matrices $K_1 = X_1^\top X_1$ and $K_2 = X_2^\top X_2$ are the same. This way we ensure that their contribution to the final kernel used by the SVM, $K = K_1 + K_2$, is the same. Hence, even if $X1$ contains 3000 rows, each being a different C2 feature, and $X2$ contains 102 rows that are the output of one vs. all classifiers, their contribution would be similar.

4.1. Toy data set – polygon identification

The random process described in figure 3 is used to create a data set of polygons with three to ten vertices. The goal is to classify each polygon image by the number of the vertices each polygon has. Since the vertex location is random and the images are decimated down to 28×28 pixels, this is not a simple task to solve using an appearance-based model. For example, it is hard to distinguish between polygons of 8-10 vertices.

Method		Recognition rate
Edge	(a)	32.67 ± 1.1
Distance Map	(a)	35.25 ± 1.7
Both concatenated	(b)	48.33 ± 1.9
Feedforward perception	(c)	39.37 ± 1.5
Reverse hierarchy	(d)	34.50 ± 1.6
Semantic concatenation	(e)	51.37 ± 1.7

Table 1. Recognition rates in percent for several hierarchical strategies applied to the polygon dataset. The mean and standard-deviation were computed over 20 independent trials. The distance map representation is preferable to the edge map. Combining them provides the best results, but these vary depending on the combination method. Semantic concatenation (e) does best, followed by simple concatenation (b). Methods (c) and (d) do not perform well.

For this experiment, the edge image is used as the low level feature, and the distance transform is used as the high level image representation. The edge image was computed by applying the canny edge detector [9]. The distance transform is computed based on the edge image, and is clearly higher in the computational hierarchy.

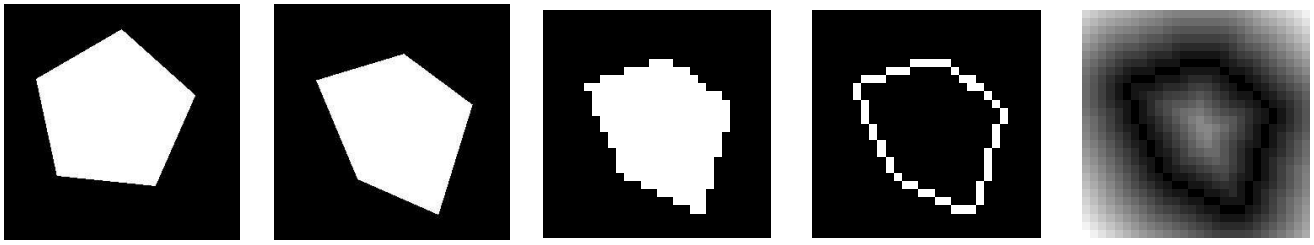
The results are given in table 1. The success rates are averaged over 20 repeats of random experiments. For each experiment, 70 random images per class were used for training (a total of 560 training images) and 130 random images per class were used for testing.

4.2. The 101 object data set

Here, results are given for the popular 101 classes dataset [13]. The results reported are the average recognition rate obtained in 10 independent trials of 15 random training and 50 different random testing images from each object class; (some classes contained less than 65 images in which case all the images not used for training were used in the test set). Using the identical protocol[1], where the errors per class are averaged, Berg reports a performance of 45% on the non-duplicated dataset [2], and Serre reports an average performance of 35% using $10,000$ "global" C2 features [24]. However, by using the code of [24] combined with a variance normalization step, we get a performance of $36.86\% \pm 1.64\%$ using only $3,000$ of the C2 features[2].

[1] In this borrowed protocol there are 102 classes: background is a class and faces and faces-easy are two separate classes.

[2] Public code is available at `http://cbcl.mit.edu/software-datasets`.



(i) (ii) (iii) (iv) (v)

Figure 3. The process for creating the toy data. (i) A perfect polygon is created with a given number of vertices. It is enclosed inside a circle of radius 100 pixels and is rotated by a random rotation. (ii) Each vertex is shifted independently by random amount of up to 40 pixels in the x and y directions. (iii) The image is decimated to 28×28 pixels. Two image descriptors are then used: (iv) the edge image, which is computed by using a canny edge detector applied to the decimated image, and (v) the distance transform of the edge image.

Method		Recognition rate
C1	(a)	30.93 ± 1.2
C2	(a)	36.86 ± 1.6
Both concatenated	(b)	44.18 ± 0.8
Feedforward perception	(c)	43.37 ± 1.0
Reverse hierarchy	(d)	45.81 ± 1.1
Semantic concatenation	(e)	45.16 ± 1.0

Table 2. Percentage recognition rate for the 101 dataset where C1 was used as the low level representation and C2 was used as the high level one. The results for (b) are given without the matrix normalization step in order to match the results given in [5]. Normalized results are very similar. (e) and (d) do best followed by (b) and (c).

Method		Recognition rate
C1 + Cont + Circ + Par	(a)	42.56 ± 1.0
C2	(a)	36.86 ± 1.6
Both concatenated	(b)	48.26 ± 0.9
Feedforward perception	(c)	50.14 ± 1.2
Reverse hierarchy	(d)	46.97 ± 0.9
Semantic concatenation	(e)	51.18 ± 1.2

Table 3. Recognition rate in percent for the 101 dataset, where the low level representation consisted of a combination of C1, Continuity, Circularity and Parallelism. Semantic concatenation (e) does best followed by (c). (b) and (d) perform worse.

To test the performance of the hierarchies on this dataset, we preformed two sets of experiments. In the first experiment we used their C1 features as the low level representation, and their C2 features as the high level representation. As can be seen in table 2 using the right hierarchies significantly improves performance.

The second set of experiments also includes the three novel image representations presented in [5]: Continuity, Circularity and Parallelism. They are used in concert with C1 as the low-level of the hierarchy, as we believe such representations to exist in the lower visual areas. As the high-level representation we again use C2. The results (table 3) show that by combining the strategy of semantic concatenation (e) with these image representations one gets the best result on the 101 dataset reported so far.

4.3. Street Scene experiments

In this experiment the hierarchy architectures were tested on 3 binary object detection experiments using the car, pedestrian, and bicycle objects from the StreetScenes database [3, 5]. Each database consists of labeled positive and negative grayscale examples of the class. The results of these experiments, in terms of the statistics of the ROC curves produced, are listed in table 4. Specifically, the mean and standard deviation of the equal error rate and the true positive rate when the false positive rate is set to 1% are listed. Independent trials consisted of dividing the data into one third for testing, and the rest for training.

The first two rows show the result of directly applying a classifier to the raw representations (*i.e.*, hierarchy a) and consequently match the results published in [5]. In this case, as in all others, the classifier used is gentleBoost, a variant of AdaBoost, which was run to convergence. The concatenation architecture (b), in which the two feature vectors are simply concatenated for each example and then classified, performed better than either feature mode alone for each class. In order to build the other three hierarchy types, it was necessary to build multiple intermediate classifiers, in order to build a perception vector. In these cases, for each necessary group of classifiers, n such intermediate classifiers were constructed by training on random subsets of the examples,a strategy similar to random forests (but without replacement). Each subset contained 20% of the training examples. For the results shown here, n was set arbitrarily to 80, but the results seem to be stable for a wide range of parameter values. Looking at the results, there does not seem to be a clear advantage to either forward, (c), or reverse, (d), hierarchies, but semantic concatination, (e), seems to consistently perform well.

Another experiment was conducted using the pedestrian

Method		car		pedestrian		bicycle	
		TP@FP=.01	Eq. err	TP@FP=.01	Eq. err	TP@FP=.01	Eq. err
C1 Direct.	(a)	82.5±4.9	5.4±1.0	32.9±5.9	19.2±2.4	62.5±8.0	8.6±2.9
C2 Direct.	(a)	64.1±5.5	7.9±0.9	44.9±5.1	13.5±1.8	49.3±9.7	11.2±2.8
C1+C2 Concatination	(b)	85.4±3.6	4.7±1.0	54.2±6.0	12.7±1.5	69.1±7.2	8.2±1.9
Feedforward perception.	(c)	87.1±3.8	4.5±1.1	46.6±10.3	13.9±1.9	69.9±8.3	7.4±2.1
Reverse hierarchy.	(d)	81.2±5.5	4.8±1.0	60.0±5.3	10.8±1.7	68.7±9.4	7.6±1.8
Semantic concatination	(e)	86.1±4.3	4.8±1.1	63.7±7.8	10.0±1.9	73.6±8.3	6.4±2.0

Table 4. Equal error rate and the true positive rate when the false positive rate is set to 1% for several hierarchical strategies applied to the StreetScenes dataset [5]. The mean and standard-deviation given were computed on 25 independent trails. For each object class, combining the feature modes in a hierarchal manner improves upon any direct classification strategy.

Method		Eq. err	AUC
Continuity	(a)	9.35 ± 2.1	95.49 ± 1.7
HoG	(a)	7.87 ± 1.5	97.02 ± 0.9
Both concatenated	(b)	6.59 ± 1.6	97.73 ± 0.8
Feedf. perception	(c)	7.33 ± 2.4	97.46 ± 1.1
Reverse hierarchy	(d)	6.66 ± 0.9	98.05 ± 0.5
Semantic conc.	(e)	6.34 ± 1.4	98.19 ± 0.6

Table 5. Equal error rate and area under the ROC curve in percent for several hierarchical strategies applied to the polygon dataset. The mean and standard-deviation given were computed on 20 independent trails. HoG is better than the baseline representation. Combining them gives best results. Strategies (b), (d), and (e) give best results, while (c) does not seem to do better than HoG.

data, but this time with the HoG [10] features as the high level representation. The edge continuity feature computed as in [5] was used as the low level information. In 20 independent trials, the data were split cleanly into 60% training, 40% testing. Here SVM was used. The results shown in table 5 suggest that all combination strategies seem better than the baseline, except for the feed-forward perception.

4.4. Analysis of the results

The general trends in the experiments seem to be that: (1) classifying a single level of the hierarchy is suboptimal. (2) using perception when combining multiple levels can be beneficial. (3) perception, in most of the experiments, works well when it is applied first to the most discriminative hierarchical level. Finally, semantic concatenation works well in the case where the hierarchy levels are similar in their discriminative power. Below we provide the details of some sanity checks we performed to verify the results.

Are the results statistically significant? Since the exact same training and testing splits were used for all of the compared algorithms we could verify that the differences are indeed statistically significant. This was done by applying the paired t-test on the set of performance measures returned by the algorithms. In the polygon experiment the semantic concatenation method is significantly better than the others ($p<0.01$, corrected for multiple comparisons). In the C1/C2 101 object dataset, reverse hierarchy and semantic concatenation are significantly better than the other methods, but not significantly different ($p=0.46$). In the experiments of table 3, semantic concatenation is significantly better than feedforward perception ($p=0.005$), and the latter is significantly better than all the rest. For the pedestrian experiment, methods (b) (d) and (e) performed significantly better than the rest ($p<0.01$), but were not significantly different.

Could the source of discrepancy be better normalization? As mentioned above, we normalized the data such that contributions from different sources would be similar. We also performed several experiments examining this normalization. In one experiment we repeated the 101 object experiment using an AdaBoost classifier, which is less sensitive to normalization issues. The results were much worse, in the range of 20% to 30%, but the relative performance of the strategies (a)-(e) remained the same.

In another control experiment we tested concatenation (b) with different scale factors. We used the polygon dataset, and scaled the distance map prior to concatenation with a factor that ranged between 10^{-5} to 10^{5}. None of these experiment resulted in performance higher than 49%.

Maybe adding a classifier on top of a 1-vs-all representation improves results? We tested whether taking a set of features, classifying it with a 1-vs-all classifier to get a vector of perceptions, and then classifying it again improves performance. Intuitively, since the basic 1-vs-all strategy is in the solution space of the two level 1-vs-all, performance may increase. It turns out that it does not. For example, for the C2 feature set on the 101 object dataset, it reduces performance slightly to 36.10% (from 36.86%).

5. Conclusions

Our results are too preliminary to tell whether the strategy of reverse-hierarchies is an effective one for object recognition. However, our work, which was inspired by Reverse Hierarchy Theory, has shown that a considerable

performance gain can be achieved by employing less common classification strategies for visual hierarchies. One implication is that feedback, even in the limited form studied here, is helpful for object recognition. For the kind of elaborate feedback systems thought to be present in the human visual system, involving attention, object based segmentation, and possibly verification loops, a performance gain in a discriminative framework is much harder to demonstrate.

Acknowledgments

This report describes research done at the Center for Biological & Computational Learning, which is in the McGovern Institute for Brain Research at MIT, as well as in the Dept. of Brain & Cognitive Sciences, and which is affiliated with the Computer Sciences & Artificial Intelligence Laboratory (CSAIL). This research was sponsored by grants from: Office of Naval Research (DARPA) Contract No. MDA972-04-1-0037, Office of Naval Research (DARPA) Contract No. N00014-02-1-0915, National Science Foundation-NIH (CRCNS) Contract No. EIA-0218506, and National Institutes of Health (Conte) Contract No. 1 P20 MH66239-01A1. Additional support was provided by: Central Research Institute of Electric Power Industry (CRIEPI), Daimler-Chrysler AG, Eastman Kodak Company, Honda Research Institute USA, Inc., Komatsu Ltd., Merrill-Lynch, NEC Fund, Oxygen, Siemens Corporate Research, Inc., Sony, Sumitomo Metal Industries, Toyota Motor Corporation, and the Eugene McDermott Foundation.

References

[1] E. Bart and S. Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *CVPR*, 2005.

[2] C. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.

[3] S. Bieschi and L. Wolf. A unified system for object detection, texture recognition and cotext analysis based on the stadard modle feature set. In *British Machine Vision Conference (BMVC)*, 2005.

[4] S. Bileschi and B. Heisele. Advances in component-based face detection. In *Proc. of Pattern Recog. with Support Vector Machines, SVM 2002*, pages 135–143, 2002.

[5] S. Bileschi and L. Wolf. Image representations beyond histograms of orientations: Nonlinear gestalt descriptors, 2006. Submitted.

[6] V. Blanz, S. Romdhani, and T. Vetter. Face identification across different poses and illuminations with a 3d morphable model. In *Intl. Conf. Face and Gesture Recog.*, 2002.

[7] L. Breiman. Random forests. *J. Machine Learning*, 2001.

[8] J. Bullier. Integrated model of visual processing. *Brain Research Reviews*, 36:96–107, 2001.

[9] J. Canny. A computational approach to edge detection. *PAMI*, 1986.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[11] S. Dickinson, A. Pentland, and A.Rosenfeld. From volumes to views: an approach to 3-d object recognition. *CVGIP: Image Underst.*, 55(2):130–154, 1992.

[12] M. Fahle and T. Poggio. *Perceptual Learning*. MIT Press, Cambridge, USA, 2002.

[13] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR*, 2004.

[14] J. Hawkins. *On Intelligence*. Times Books, 2004.

[15] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: global versus component-based approach. In *ICCV*, 2001.

[16] S. Hochstein and M. Ahissar. View from the top: Hierarchies and reverse hierarchies in the visual system. *J. Neuron*, 36(5):791–804, 2002.

[17] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *PAMI*, 2005.

[18] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004.

[19] B. Leibe, A.Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Stat. Learning in Comp. Vision*, 2004.

[20] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of 3d shapes. In *Proceedings of the Royal Society of London. Series B,*, volume 200, 1978.

[21] D. Perrett and M. Oram. Neurophysiology of shape processing. *Imaging Vis. Comp.*, 11:317–333, 1993.

[22] M. C. Potter. Short-term conceptual memory for pictures. *Journal of Experimental Psychology*, 2:509–522, 1976.

[23] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *CVPR*, 1998.

[24] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005.

[25] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996.

[26] S. Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS 8*, 1996.

[27] F. Tong and K. Nakayama. Robust representations for faces: Evidence from visual search. *Journal of Experimental Psychology: Human Perception and Performance*, 1990.

[28] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2):153–167, July 2003.

[29] A. Treisman. Features and objects: The fourteenth bartlett memorial lectures. *Quartly Journal of Experimental. Psychology*, 40A:201–237, 1998.

[30] S. Ullman and E. Sali. Object classification using a fragment-based representation. In *Biologically Motivated Computer Vision,*, 2000.

[31] M. Weber, W. Welling, and P. Perona. Towards automatic dscovery of object categories. In *CVPR*, 2000.

[32] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.