DIRAC
Detection and Identification of Rare Audiovisual Cues

Insperata accident magis saepe quam quae speres.
(Things you do not expect happen more often than
things you do expect) Plautus (ca 200(B.C.)

Project no: 027787

# DIRAC

## Detection and Identification of Rare Audio-visual Cues

Integrated Project
IST - Priority 2

DELIVERABLE NO: D5.5
Example/Demo of Hierarchical Multimodal Fusion

*Date of deliverable:* 30.06.2007
*Actual submission date*: 09.08.2007

Start date of project: 01.01.2006 Duration: 60 months

*Organization name of lead contractor for this deliverable*: **IDIAP Research Institute**

Revision [1]

| Project co-funded by the European Commission within the Sixth Framework Program (2002-2006) | | |
|---|---|---|
| Dissemination Level | | |
| PU | Public | X |
| PP | Restricted to other program participants (including the Commission Services) | |
| RE | Restricted to a group specified by the consortium (including the Commission Services) | |
| CO | Confidential, only for members of the consortium (including the Commission Services) | |

DIRAC

Detection and Identification of Rare Audiovisual Cues

Insperata accident magis saepe quam quae speres.
(Things you do not expect happen more often than
things you do expect) Plautus (ca 200(B.C.)

# D5.5 Example/Demo of Hierarchical Multimodal Fusion

IDIAP Research Institute (IDIAP)
The Hebrew University of Jerusalem (HUJI)

***Abstract:***
In this work we describe three different applications of hierarchical processing for audio and visual data. The deliverable is divided into three parts. In part 1 an object hierarchy for combining models form different category levels is presented. In part 2 relational object models for sub-ordinate class recognition are described. Finally in part 3 hierarchical combination of acoustic features for large vocabulary continuous speech recognition is presented.

## Table of Content

*In Proc. 11th IEEE International Conference on Computer Vision, Rio-de-Janeiro Brazil, October 2007.*

# Exploiting Object Hierarchy: Combining Models from Different Category Levels

Alon Zweig Daphna Weinshall
School of Computer Science and Engineering
Hebrew university of Jerusalem, Israel 91904
{zweiga, daphna}@cs.huji.ac.il

## Abstract

*We investigated the computational properties of natural object hierarchy in the context of constellation object class models, and its utility for object class recognition. We first observed an interesting computational property of the object hierarchy: comparing the recognition rate when using models of objects at different levels, the higher more inclusive levels (e.g., Closed-Frame Vehicles or Vehicles) exhibit higher recall but lower precision when compared with the class specific level (e.g., bus). These inherent differences suggest that combining object classifiers from different hierarchical levels into a single classifier may improve classification, as it appears like these models capture different aspects of the object. We describe a method to combine these classifiers, and analyze the conditions under which improvement can be guaranteed. When given a small sample of a new object class, we describe a method to transfer knowledge across the tree hierarchy, between related objects. Finally, we describe extensive experiments using object hierarchies obtained from publicly available datasets, and show that the combined classifiers significantly improve recognition results.*

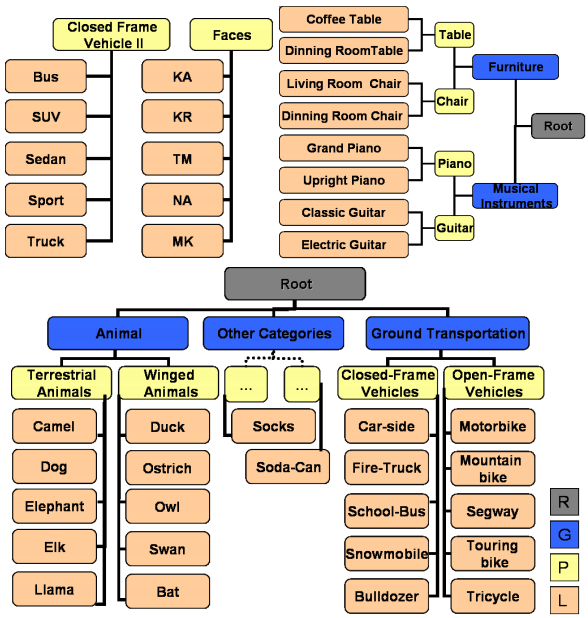


Figure 1. The four hierarchies used in our experiments. Categories at different levels of the tree are labeled (and color-coded) as follows: 'L' denotes the Leaf level, 'P' denotes the Parent level, 'G' denotes the Grandparent level, and 'R' the level of all objects. Our largest hierarchy (lowest diagram above) contains object classes from the CalTech256 database [8]. The facial hierarchy contains objects from [12].

## 1. Introduction

Human cognition relies on a hierarchal representation of objects in the world (see examples in Fig. 1), in the process of recognizing and referring to objects. How can we use such hierarchical structure to improve object recognition and categorization? This question has been addressed in a number of recent papers, mostly pursuing different directions to exploit this hierarchy when confronted with new categories (the small sample problem). The directions under study included the transfer of knowledge from known to new categories using Bayesian priors [11], sharing parts between objects at different levels of the hierarchy and improving generalization [13, 3], learning distance functions using related classes [6, 9], and transferring features [10, 4] or structure [5] from known classes to new ones. Often, when working on object class recognition, objects are represented by parts (or features) learned directly from example images of each object category, where relations between the parts (geometrical and possibly other) may sometimes be captured by graphical models trained using the same data.

We address the question posed above from a somewhat different point of view. We observe that the natural object hierarchy offers at our disposal a rich family of classifiers, for each category in each node of the hierarchy tree; the similarity between these classifiers varies, possibly in some relation to the distance between them on the object tree, but they all share some common characteristics. For example, if we build these classifiers with a discriminative algorithm that uses the background images of the CalTech256

database [8], then all the classifiers are trained to distinguish a certain isolated object from a background of clutter. Such commonalities may permit the combination of different classifiers to improve the recognition of specific object classes. We expect this improvement to be more pronounced when using related objects, and in particular objects from higher (inclusive) levels in the hierarchy tree.

The idea of combining classifiers has been extensively studied in the machine learning community under different frameworks, including committee machines, ensemble average, or boosting. In light of the so-called bias/variance dilemma [7], the ensemble average of a number of classifiers should improve generalization, as long as the classifiers are both accurate and diverse. One common way to obtain such a collection of classifiers is to train different classifiers with different samples of the training data. But note that the object recognition classifiers, trained to recognize different objects at different levels of the object hierarchy tree, may be viewed as just such - classifiers trained on different resamples of the training data. Viewed this way, we may expect to see improvement when combining any set of classifiers, even those trained to recognize very distinct objects. At the same time, for obvious reasons, we expect to see larger improvement when combining classifiers trained to recognize similar objects (those closer to each other on the tree), as compared to very different ones.

Thus the main conceptual contribution of this paper is to identify the object hierarchy as a source of classifier variability, which is induced by different and inherently meaningful resampling of the training data. We then go ahead and describe a framework to combine the classifiers linearly, using each classifier's probabilistic output and the corresponding LRT (Loglikelihood Ratio Test) value. This approach is somewhat different, and possibly more powerful, than the traditional ensemble of classifiers, since each object classifier builds a different representation of the data based on the training subset that it sees. The approach differs from boosting and bagging in that the data resampling is not based on some bootstrapping procedure, but on supervised information given to the system via the object hierarchy tree. Clearly our approach could be augmented with boosting to further improve results.

The rest of this paper is organized as follows. First, we review in Section 2 the object class constellation model used to obtain each object classifier, and describe how we combine these classifiers. We also discuss the theory underlying our approach. In Section 4 we show that a number of intuitive object hierarchies (described in Section 3), provided by a human teacher, reveal consistent and sensible computational characteristics. Specifically, classifiers built for more specific objects (such as 'bus') - corresponding to the lowest level in the object tree, are characterized by high precision (or high specificity) and low recall (low sensitivity), while classifiers built for more general classes of objects at the next levels up the tree (such as 'vehicle') show the opposite - low specificity and high sensitivity. In general, the specificity of object classifiers decreases as we ascend the object tree. Then, In Section 5, we describe how to use the hierarchy to transfer knowledge between classes, as a way to address the small sample problem. Finally, we investigate in Section 6 combined classifiers as a general framework for the construction of object recognizers. We show empirically that combined classifiers improve performance significantly (over all constituent classifiers), and that typically three-level combinations perform better still than two-level combinations.

## 2. Combining Object Models

**Our object class model** To learn object models, we use the method described in [2], because its computational efficiency allows us to consider models (and images) with many features. The algorithm learns a generative relational part-based object model, modeling appearance, location and scale. Location and scale are relative to the unknown object location and scale, as captured by a star-like Bayesian network. The model's parameters are discriminatively optimized using an extended boosting process. This model has been shown to achieve competitive recognition results on standard benchmark datasets, approaching the state-of-the-art in object class recognition. Thus we believe that the results and improvements we show are general, and can be replicated with other, conceptually similar, part-based models.

Based on this model and some simplifying assumptions, the likelihood ratio test function is approximated (using the MAP interpretation of the model) by

$$F(\mathbf{x}) = \max_{C} \sum_{k=1}^{P} \max_{u \in Q(\mathbf{x})} \log p(u|C, \theta^k) - \nu \qquad (1)$$

with $P$ parts, threshold $\nu$, $C$ denoting the object's location and scale, and $Q(\mathbf{x})$ the set of extracted image features.

**Classifier combination rule** In our experiments we combined 2, 3 and 4 object classifiers. For each classifier, we used its LRT value from (1), obtaining a 2-, 3- or 4-dimensional vector respectively. We then trained a Support Vector Machine classifier with the same training data represented in this new vector space, using a linear kernel.

**The bias/variance dilemma** Let $\mathbf{x} \in \mathcal{X}$ denote the image, $F(\mathbf{x})$ the LRT output of the classifier from (1), and $D(\mathbf{x})$ denote the binary random variable assigning 1 to class images, and -1 to background images. It can be readily

shown that the mean-square error between classifier $F$ and the desired output $D$ can be decomposed as follows:

$$\begin{aligned} E[(F(\mathbf{x}) - E[D(\mathbf{x})])^2] &= B(F(\mathbf{x})) + V(F(\mathbf{x})) \\ B(F(\mathbf{x})) &= (E[F(\mathbf{x})] - E[D(\mathbf{x})])^2 \\ V(F(\mathbf{x})) &= E[(F(\mathbf{x}) - E[F(\mathbf{x})])^2] \end{aligned}$$

where $B(F(\mathbf{x}))$ denotes the classifier's bias, and $V(F(\mathbf{x}))$ its variance. It can also be shown that when considering an ensemble average of such classifiers, the bias of the new classifier remains the same as the bias of $F(\mathbf{x})$, but its variance is reduced. As a result, the mean square error of the new classifier is also reduced.

**The sensitivity/specificity tradeoff** We now analyze the case of two classifier combination, where one classifier has high sensitivity and low specificity and the other has low sensitivity and high specificity. Recall that this is the computational property that distinguishes object classifiers from lower levels of the object hierarchy tree and classifiers from higher levels of the tree (see Section 4), and thus this analysis is revealing.

Given the function $F(\mathbf{x})$ from (1), define the classifier $F^*(\mathbf{x}) = sign(F(\mathbf{x}))$[1]. Let $F_1(\mathbf{x}), F_2(\mathbf{x})$ denote two classification functions, and $G(\mathbf{x}) = \frac{F_1(\mathbf{x})+F_2(\mathbf{x})}{2}$ its ensemble average. Let $G^*(\mathbf{x}) = sign(G(\mathbf{x}))$ denote the corresponding classifier, and let $G^{**}(\mathbf{x}) = \frac{F_1^*(\mathbf{x})+F_2^*(\mathbf{x})}{2}$ denote another related classifier.

We compute the error probability $P_E$ of classifier $G^*(\mathbf{x})$:

$$\begin{aligned} &4P_E(G^*(\mathbf{x})) = E[(G^*(\mathbf{x}) - D(\mathbf{x}))^2] \qquad (2) \\ &= E[((G^*(\mathbf{x}) - G^{**}(\mathbf{x})) + (G^{**}(\mathbf{x}) - D(\mathbf{x})))^2] \\ &= E[(G^* - G^{**})^2] + E[(\tfrac{1}{2}(F_1^* - D))^2] \\ &+E[(\tfrac{1}{2}(F_2^* - D))^2] + 2E[(\tfrac{1}{2}(F_1^* - D))(\tfrac{1}{2}(F_2^* - D))] \\ &+2E[(G^* - G^{**})]\{E[\tfrac{1}{2}(F_1^* - D)] + E[\tfrac{1}{2}(F_2^* - D)]\} \end{aligned}$$

Note that

$$E[(\tfrac{1}{2}(F_i^*(\mathbf{x}) - D(\mathbf{x})))^2] = P_E(F_i^*)$$

and

$$\begin{aligned} E[\tfrac{1}{2}(F_i^*(\mathbf{x}) - D(\mathbf{x}))] &= P[F_i^*(\mathbf{x}) = 1, D(\mathbf{x}) = -1] \\ &-P[F_i^*(\mathbf{x}) = -1, D(\mathbf{x}) = 1] = \Delta S(F_i^*) \end{aligned}$$

where $\Delta S(F_i^*)$ denotes the classifier's preference to either recall (sensitivity) or precision (a measure typically similar to specificity)[2], i.e., its sensitivity minus its specificity.

[1]For convenience, we define the sign function as $sign(F) = 1$ if $F \geq 0$, and $sign(F) = -1$ if $F < 0$.

[2]Notation reminder: *recall* and *sensitivity* denote the rate of true positives, *specificity* denotes the rate of true negatives, and *precision* denotes the fraction of true positives among all examples identified as positive.

Henceforth we shall call $\Delta S(F_i^*)$ the 'recall/precision primacy'. Finally,

$$\begin{aligned} E[(G^* - G^{**})^2] &= P(F_1^*=1,F_2^*=-1) + P(F_1^*=-1,F_2^*=1) \\ &\leq P_E(F_1^*) + P_E(F_2^*) \end{aligned}$$

(with equality only when $F_1^*, F_2^*$ err on disjoint sets of examples).

Putting all the above together, we get

$$\begin{aligned} 4P_E(G^*) \leq &2P_E(F_1^*) + 2P_E(F_2^*) + 2\Delta S(F_1^*)\Delta S(F_2^*)] \\ &+2E[(G^* - G^{**})](\Delta S(F_1^*) + \Delta S(F_2^*)) \\ &-2\rho(\tfrac{1}{2}(F_1^* - D), \tfrac{1}{2}(F_2^* - D)) \qquad (3) \end{aligned}$$

where $\rho(X, Y) = E[XY] - E[X]E[Y]$ denotes the non-normalized correlation coefficient of $X, Y$.

We can now state our main result:

**Result:** Assume that $F_1^*, F_2^*$ are two classifiers with opposite recall/precision primacy, i.e. and w.l.o.g., $\Delta S(F_1^*) \geq 0$ and $\Delta S(F_2^*) \leq 0$; thus $\Delta S(F_1^*) \cdot \Delta S(F_2^*) \leq 0$. Assume further that the magnitude of their primacy is similar, i.e., $|\Delta S(F_1^*)| \approx |\Delta S(F_2^*)|$, and that their correlation with respect to the data is small, i.e., $|\rho(\frac{1}{2}(F_1^* - D), (\frac{1}{2}(F_2^* - D)))| < |E[\frac{1}{2}(F_1^* - D)]E[\frac{1}{2}(F_2^* - D)]|$. Then it follows from (3) that

$$P_E(G^*) \leq \frac{P_E(F_1^*) + P_E(F_2^*)}{2}$$

In other words, the error probability of the combined classifier is smaller (usually significantly so) than the mean error probability of the constituent classifiers.

In practice, we see that the combined error is typically smaller than the minimal error of the constituent classifiers with opposite recall/precision primacy, see Section 6.

## 3. Datasets and General Experimental Setup

### Datasets

In our experiments, we used an extensive data set containing various objects that can be found in natural scenes. As much as possible, classes were taken from standard benchmark datasets, with a few exceptions (to be detailed shortly). We organized these objects into four natural hierarchies. Examples from the object classes and background images can be viewed in Fig. 2. A summary of the hierarchies is provided in Fig. 1.

In the rest of this paper we use the following notation to refer to object classes at different levels in the hierarchy (see Fig. 1): specific object classes, like 'Elk' and 'Tricycle', are labeled 'L' (for Leaf). More inclusive categories, like 'Terrestrial Animals', are labeled 'P' (for Parent). Categories at

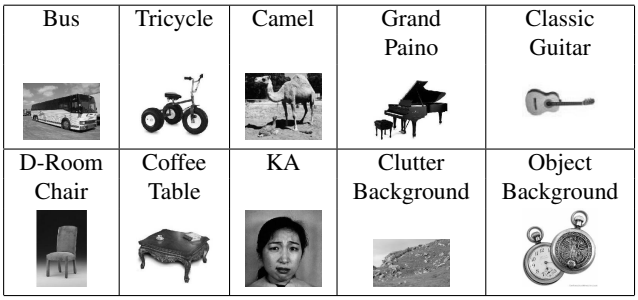


Figure 2. Examples taken from the object classes and background images, used to train and test our different Category level models, see Fig. 1.

the next level up, like 'Animals', are labeled 'G' (for Grandparent). Finally, the category of all objects is denoted 'R' (for Root).

For the discriminative learning procedure (see Section 2) and in order to evaluate the recognition results, we used two types of background. When using classes from the CalTech256 dataset [8] (the lowest hierarchy in Fig. 1), we used their Clutter Background images as well. With the remaining 3 smaller hierarchies and to achieve greater variability (and additional challenges) in the conditions of our experiments, we used our own Object Background dataset, containing various images of objects different from the learnt objects. This background was manually collected using Google, PicSearch and online catalogues.

**CalTech256 Object Hierarchy** This hierarchy includes pictures of various objects from the CalTech256 dataset [8], see Fig. 1. We chose objects that can be naturally organized into a sensible hierarchy: 'Animals' - including 'Terrestrial Animals' and 'Winged Animals', and 'Ground Transportation' - including 'Open-Frame Vehicles' and 'Closed-Frame Vehicles'. Models were learnt for objects from the 'Terrestrial Animals' and 'Open-Frame Vehicles' classes; other objects were used for training 'G' and 'R' level models. The CalTech256 Clutter Background was used with this dataset. We note that our category affiliations may not be identical to those used in [8], in an attempt to emphasize visual similarities over functional (thus ignoring, for example, the motorized vs. un-motorized distinction); we also used different names for the inclusive categories. The images in the Ground-Transportation Category were flipped to achieve uniform orientation (all vehicles pointing rightwards).

**Closed-frame VehicleII Hierarchy** This hierarchy contains 5 classes of common vehicles (more so than those in the CalTech256 database), contrasted with pictures from the 'Object Background'. Pictures (for the vehicle classes and background) were chosen manually from Google and PicSearch, showing vehicles at similar canonical orientation.

**Faces Hierarchy** This hierarchy contains pictures of 5 individuals taken from [12], with varying facial expressions.

**Basic-Level Hierarchy** This hierarchy was built to match standard hierarchies favored in the cognitive science literature, representing canonical categorization levels (basic-level, sub-ordinate, and super-ordinate). Pictures were obtained from the CalTech101 subset of [8], or collected using mainly online shopping catalogues.

### General experimental setup

In general, we *always* tested recognition performance for specific object categories from level 'L' (leaf) of all the respective hierarchies. Thus, for example, when comparing three models such as 'Llama', 'Animal', and 'Terrestrial Animal', they were all tested on the recognition of Llama pictures.

For each hierarchy, all models were trained with the same background images but different object images. All tests were done with the same background and test images, and the same algorithm parameters. Each experiment was repeated 60-100 times, with new random samples of train and test images. Since the Equal Error Rate (denoted EER) of the ROC is not well suited when the number of positive examples is much smaller than the number of negative examples [1], we used the preferred EER of the Recall Precision Curve (RPC).

To compare performance, we report two measures: (i) Precision and Recall of each classifier; (ii) EER of the RPC curve for each classifier, computed by varying the threshold of the optimal linear SVM classifier.

## 4. Object Hierarchy

We study the computational properties distinguishing objects from different levels in the object hierarchy tree, revealing opposite recall/precision primacy - high precision for the lowest level (specific) models, and high recall for higher level (inclusive) models. With sufficiently large samples per object, and given that we always test the recognition of object classes from level 'L', not surprisingly object models from level 'L' show superior recognition performance. In accordance, we see a decrease in performance as we use models ascending the object hierarchy tree.

**Experimental setup** We tested the recognition of each specific object from level 'L' by 5 types of models learnt using object categories from different levels in the hierarchy tree, see Table 1. To assure fair comparison, all models saw the same train images of the 'L' object they were tested on; an illustrative example of this procedure is shown in Table 1. In different experiments we varied the number of train images per 'L' object: 5, 10, 15, 20, 25 and 30. Three

hierarchies were used: CalTech256 Object, Closed-frame VehicleII, and Faces.

| Exp | Category training set L P G R DB | Example: Llama |
|---|---|---|
| 1 | 1 | Llama |
| 2 | 5 | Llama, Camel, Dog, Elk Elephant |
| 3 | 5 | Llama, Duck, Owl Swan, Ostrich |
| 4 | 5 | Llama, Soda-can, Sock Segway, Motorbike |
| 5 | 5 | Llama, Segway, Tricycle Motorbike, Mountainbike |

Table 1. This table shows the 5 different models learnt and evaluated on the recognition of 'L' level objects. 'DB' refers to a 'G'-level category from a Different Branch of the tree. Examples are shown for the Llama as 'L' level object. In each different experiment, each 'L' class provided the same fixed amount of pictures to the training set (5, 10, 15, 20, 25 or 30.)

With all the 3 hierarchies, we used test data composed of images of the target object and images of the relevant background in equal proportion. None of the test images was used for training. With the CalTech256 Object Hierarchy, where the models are learnt using the Clutter Background, we also conducted additional experiments, using for test data images from the target 'L' object mixed with images of a different 'L' object.

### 4.1. Results

Recall and Precision are shown in Fig. 3, comparing recognition when using 'L' and 'P' level models (left), and 'L' and 'G' level models (right). In the first comparison (Leaf vs. Parent), only 4 representative examples from the 3 hierarchies are shown; very similar results were obtained with all other objects. In the second comparison (Leaf vs. Grandparent), all objects from the CalTech256 Object Hierarchy are shown. These graphs clearly show the Recall/Precision Primacy effect, where 'L' models show high precision and low recall in recognition, while 'P' and 'G' models show high recall and low precision. This happens for all objects, regardless of the number of training examples.

The Recall Precision Curve and its corresponding EER are shown in Fig. 4 for two representative examples. Not surprisingly, we see that with sufficient training examples per 'L' object (as for the Dog class), the 'L' model performs best, and performance deteriorates as we ascend the object hierarchy. As the sample per class decreases, the advantage of the 'L' model over the 'P' model decreases, eventually the 'P' model might outperform the 'L' model. Once again, similar phenomenon is observed for all objects.

Looking more closely at these results, we see that the

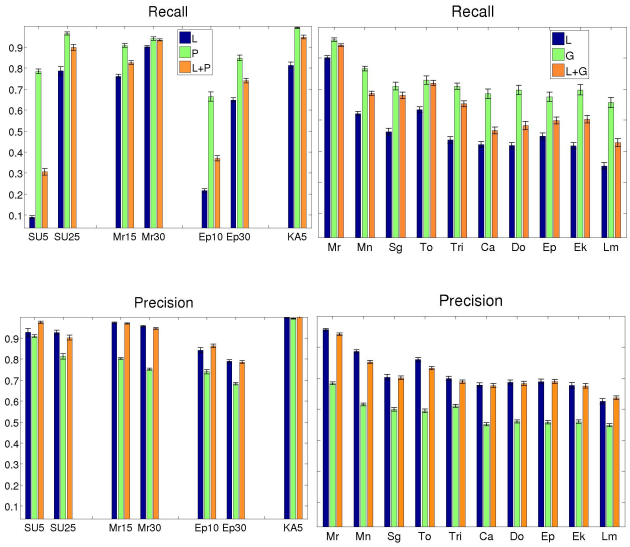


Figure 3. Recall and Precision of classifiers. Left column: four representative object classes, recognized with the 'L', 'P' and the combined 'L+P' models. 'SU' refers to the SUV class, 'Mr' - Motorbike, 'Ep' - Elephant, 'KA'- one of the female faces. The numbers indicate the size of the train set (e.g., 'SU5' refers to the SUV model trained with 5 SUV examples). Right column: all learnt object classes from the CalTech256 Object Hierarchy trained with 30 images per object class, and recognized with the 'L', 'G' and the combined 'L+G' models.

recall/precision primacy difference occurs regardless of the overall recognition rate. Specifically, for the Elephant with 10 training examples, we see from Fig. 4 that the 'P' model performs better than the 'L' model, and vice versa for the Elephant with 30 training examples. Still, Fig. 3 shows the same Recall/Precision primacy in both cases.

## 5. Using hierarchy to transfer knowledge

We study here how to transfer information between related objects, located nearby in the object hierarchy tree, to handle the problem of small sample or the appearance of new objects.

**Experimental setup** We tested the recognition of each specific object from level 'L' by 7 types of models learnt using object categories from different (more inclusive) levels in the hierarchy tree, see Table 2, which also shows an illustrative example of this procedure. The test background set consisted of 75 background images, while the test object set consisted of 30 images. Only the Basic-Level Hierarchy was used.

### 5.1. Results

Fig. 5 shows the results. Clearly the 'P' class model transfers information most effectively (seen in the superi-

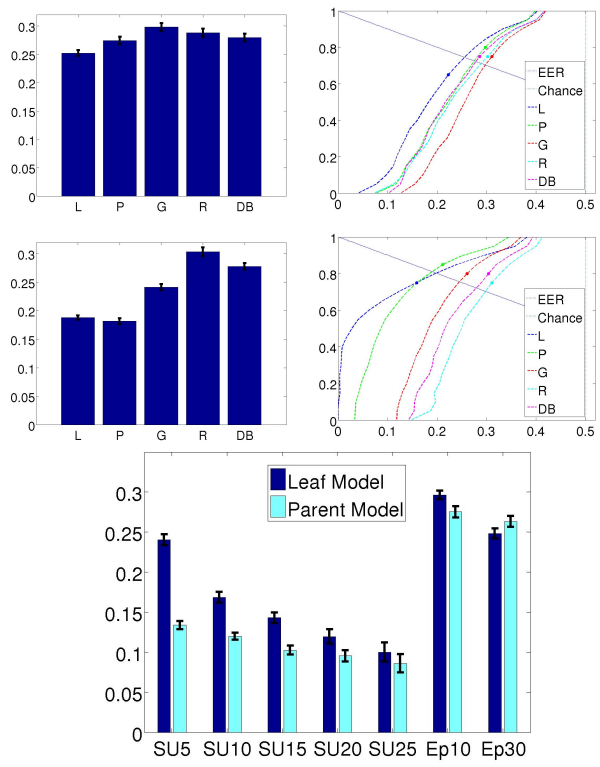


Figure 4. Performance of the different category level models. Top & middle: left column - the EER of the RPC, right column - full RPC curve where the point of the original classifier is highlighted . Once again, 'L' denotes the Leaf category, 'P' - Parent, 'G' - GrandParent, 'R' - Root, and 'DB' - Different Branch. Top: performance of all models tested on the 'Dog' class from the Caltech256 hierarchy. Middle: same for the 'Mountain-Bike' class from the Caltech256 hierarchy. Bottom: performance of the 'L' and 'P models on the 'SUV' class from the 'Closed-Frame VehiclesII' hierarchy and on the Elephant class from the 'Terrestrial Animals'. EER scores are shown as a function of the size of the training set - increasing from 5 up to 25 for the SUV, sizes 10 and 30 for the Elephant. Note the decrease in the performance superiority of the SUV 'P' model over the 'L' model till it is insignificant, as the train size increases. Note the opposite superiority of 'L' vs. 'P' models when comparing the two Elephant class models.

ority of Exp. 3 over Exp. 5-7), and improves performance over the small sample case (seen in the superiority of Exp. 4 over Exp. 2).

### 5.2. Discussion

The results above show a clear hierarchy structure, where models which are learnt from nearby objects (brothers) in the object hierarchy tree can substantially improve the recognition results of each other. It shows the possibility for the success of a learning-to-learn scheme - where fewer examples of the goal object class are used in the learning process, augmented by examples from different related classes.

| Ex | Object training set | | | | | Example: |
|---|---|---|---|---|---|---|
| | L | P | G | DB | BG | Classic Guitar |
| 1 | 35 | | | | | Classic Guitar |
| 2 | 1 | | | | | Classic Guitar |
| 3 | | 30 | | | | Electric Guitar |
| 4 | 1 | 30 | | | | Classic, Electric Guitar |
| 5 | | | 30 | | | Grand Piano |
| 6 | | | | 30 | | Living Room Chair |
| 7 | | | | | 30 | Background |

Table 2. The models learnt in the different experiments on the transfer of information between classes. 'L' refers to the Leaf level, 'P' to the Parent, 'G' to the Grandparent, 'DB' to a Different Branch of the tree, and 'BG' to the background.

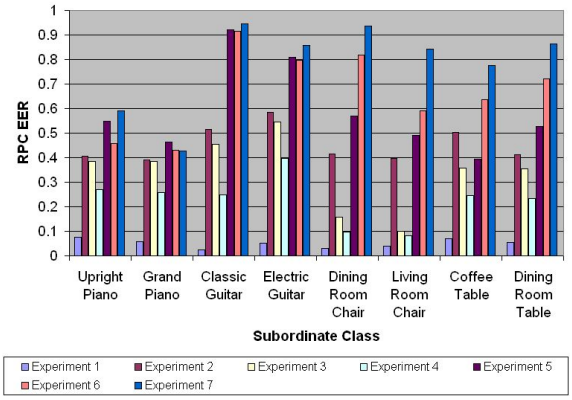


Figure 5. Transfer of knowledge between related classes. Description of the different experiments is given in Table 2.

## 6. Using hierarchy to improve classification via combination

We now ask whether the combination of two or more object model classifiers, which are based on different category levels, can improve the performance of the original classifiers.

**Experimental setup** We used the same setup, same data, and same learnt models as used in Section 4. We tested different combinations by combining two or more models from different category levels as described in Section 2. The different combinations that we studied are summarized in Table 3.

For comparison, we used a naïve alternative method, which learned directly an object model using the same set of images as used by the combined model. Each image in this set was initially weighted to reflect its true weight on the combined model. For example, when combining two models such as 'Llama' and 'Terrestrial Animals', we note that the Llama images provided all the training set for the Llama model, and only 20% of the training set for the 'Ter-

| Exp | Example:<br>Llama as Leaf Level |
|---|---|
| L+P | 'Llama' + 'Terrestrial Animals' |
| L+G | 'Llama' + 'Animals' |
| L+R | 'Llama' + 'Tree Root' |
| L+DB | 'Llama' + 'Open-Frame Vehicles' |
| P+G | 'Terrestrial Animals' + 'Animals' |
| L+P+G | 'Llama' + 'Terrestrial Animals' + 'Animals' |
| L+P+G+R | 'Llama' + 'Terrestrial Animals' + 'Animals' + 'Tree Root' |

Table 3. The different combinations we studied: '+' denotes a combination of two models. 'L' refers to the Leaf level, 'P' - Parent, 'G' - Grandparent, 'R' - Root, and 'DB' - Different Branch.

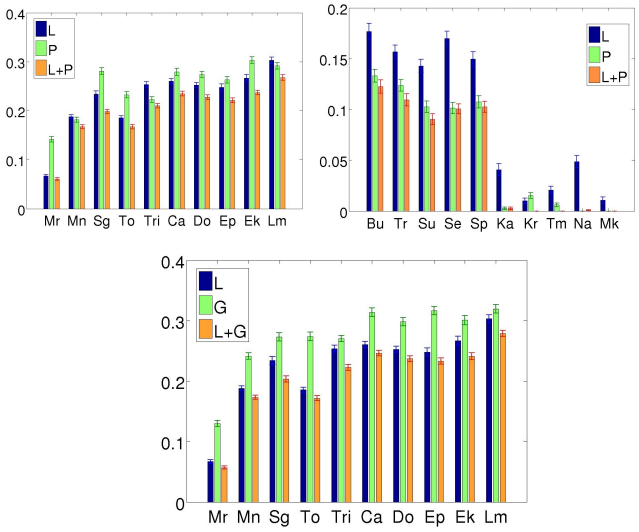


Figure 6. Comparison of combined and original classifiers. 'L' denotes a leaf model, 'P' - parent model, 'G' - grandparent model, 'L+P' leaf/parent combined model and 'L+G' leaf/grandparent combined model. Top: object models recognized with the 'L', 'P' and the combined 'L+P' models. Bottom: object models recognized with the 'L', 'G' and the combined 'L+G' models. Top-Left: CalTech256 Object Hierarchy, 'Open-frame Vehicle' and 'Terrestrial Animal', with 30 training images per object class. Specifically, 'Mr' denotes the Motorbike class, 'Mn' - Mountain-Bike , 'Sg' - Segway, 'To' - Touring-Bike, 'Tri' - Tricycle, 'Ca' - Camel, 'Do' - Dog, 'Ep' - Elephant, 'Ek' - Elk, 'Lm' - Llama. Top-Right: 'Closed-Frame VehicleII' Hierarchy with 15 training images per object class, and 'Faces' Hierarchy with 5 training images per object class. Bottom: CalTech256 Object Hierarchies, with 30 training images per object class

restrial Animals' model; thus in the training of the combined model, the Llama training set received total weight of 0.6, while the remaining 4 classes received total weight of 0.4. The background train set remained unchanged.

### 6.1. Results

Fig. 6 shows the EER recognition results for all 20 objects, from the 'P' classes of 'Terrestrial Animals', 'Open-

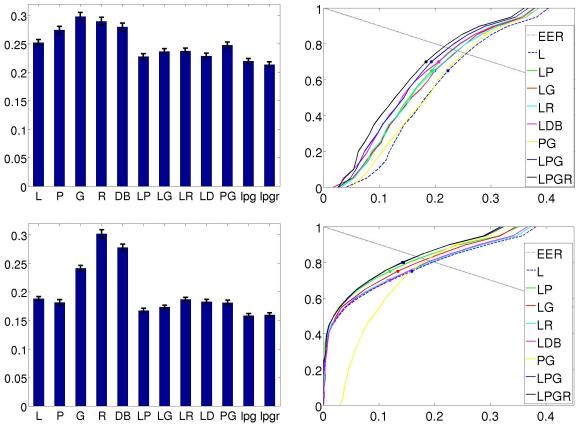


Figure 7. All different combinations of classifiers. Left: the EER of the RPC. Right: full RPC (Recall Precision Curves). Top: results when recognizing the 'Dogs' class. Bottom: results with the 'Mountain Bike' class.

Frame Vehicles', 'Closed-Frame VehiclesII', and 'Faces'. We show recognition results with 3 models - 'L', 'P', and the combined 'L+P', fixing the number of training examples to 30, 30, 15, and 5 for each object in the 4 'P' classes respectively. We also show recognition results with 3 models - 'L', 'G', and the combined 'L+P', with 30 training examples and two classes of the CalTech256 Object Hierarchy.

Clearly, almost always, the combined model performed better than both constituent models. This happened for all objects and all training conditions, regardless of which of the constituent models was initially superior. The only exception occurred in the experiments with only 5 training images per object class (small sample). Moreover, in all experiments the combined model improved significantly the weak measure (either Recall or Precision) of each of the constituent models, as demonstrated in Fig. 3.

Fig. 7 shows results with the 7 different classifier combinations, listed in Table 3, for two object classes. These are representative results - similar results were obtained with all other classes. Note that the two-level combinations that obtain the highest performance are either the 'L+P' or 'L+G'. Not surprisingly, therefore, the best results are obtained with the three- and four-level combinations ('L+P+G' and 'L+P+G+R' respectively).

Fig. 8 shows the EER of the RPC in the second test condition, when test examples included an equal number of images from the target object and another unrelated distractor object (instead of the standard background images). Not surprisingly, when the 'L' model was combined with a model whose training set included pictures of the distractor object, the performance of the combined model was reduced. However, interestingly enough, this reduction is rather slight (see Fig. 8). This decrease remains slight even when the 'L' model is combined with a very poor classifier (under these conditions), like the 'R' or 'DB' ones. Thus

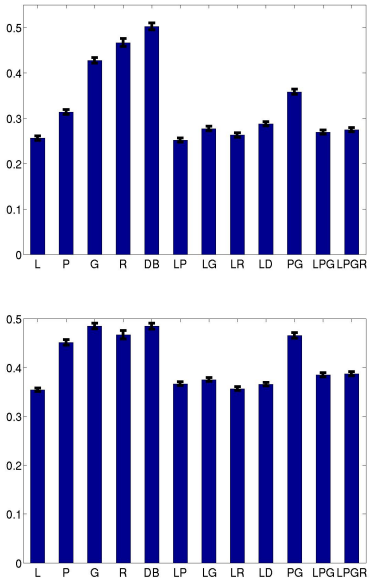


Figure 8. The EER of the RPC when the test examples included images of the target object - the 'Mountain Bikes' - and another object (instead of the standard background images). Top: models tested against 'Camel'. Bottom: models tested against 'Tricycle'.

these results seem to suggest that the improvement obtained by the combined classifier against the standard background is not accomplished at the high cost of reducing the discriminability of the new classifier against other, possibly related, objects.

Finally, we note that in most cases, the comparison against the naïve model showed superior results for the combined model, while in the other cases the advantage of the naïve model was not significant. On the other hand, the training time of the naïve model was substantially longer. Moreover, this training procedure is not modular, while the combination scheme we described is rather flexible.

## 7. Summary and Discussion

We analyzed the computational properties of constellation object class models, built to describe object categories at different levels of the object hierarchy tree. An interesting observation emerged, when comparing specific object models, trained using images of objects corresponding to the leaves of the hierarchy tree, with models built to describe categories at higher levels of the object hierarchy tree. The first (specific) models exhibit higher precision, while the second (inclusive) models exhibit higher recall. We provided the theoretical analysis showing why this situation should be favorable for the success of a classifier combined from two such constituents (one with higher precision, the other with higher recall), and demonstrated experimentally that significant improvement is indeed achieved in all cases. In our experiments the combined model performed better than all constituents models in almost all cases. The improvement magnitude was larger when the constituent classifiers corresponded to nearby objects in the hierarchy tree, showing that this improvement is not due simply to the larger training set.

In all our experiments, we used a specific part-based model that can be learned rather efficiently, and can therefore handle a relatively large number of parts (or features). Although we did not perform experiments to this effect, we believe that this improvement can be obtained with any object class model, and that the phenomena we have observed do not depend on the specific model we used.

## References

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proc. ECCV*, 2002. 4

[2] A. Bar-Hillel, T. Hertz, and D. Weinshall. Efficient learning of relational object class models. *Proc. ICCV*, 2005. 2

[3] A. Bar-Hillel and D. Weinshall. Subordinate class recognition using relational object models. *Proc. NIPS*, 19, 2006. 1

[4] E. Bart and S. Ullman. Cross-generalization: learning novel classes from a single example by feature replacement. *Proc. CVPR*, pages 672–679, 2005. 1

[5] A. Ferencz, E. Learned-Miller, and J. Malik. Building a classification cascade for visual identification from one example. *Proc. ICCV*, pages 286–293, 2005. 1

[6] M. Fink. Object classification from a single example utilizing class relevance metrics. *Proc. NIPS*, 17, 2004. 1

[7] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Comput.*, 4(1):1–58, 1992. 2

[8] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report UCB/CSD-04-1366, California Institute of Technology, 2007. 1, 2, 4

[9] T. Hertz, A. Bar-Hillel, and D. Weinshall. Learning distance functions for image retrieval. *Proc. CVPR*, 2, 2004. 1

[10] K. Levi, M. Fink, and Y. Weiss. Learning From a Small Number of Training Examples by Exploiting Object Categories. *LCVPR04 workshop on Learning in Computer Vision*, 2004. 1

[11] F. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE PAMI*, 28(4):594–611, 2006. 1

[12] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. *Proc. ICAFGR*, pages 200–205, 1998. 1, 4

[13] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky. Learning hierarchical models of scenes, objects, and parts. *Proc. ICCV*, 2005. 1

# Subordinate class recognition using relational object models*

**Aharon Bar Hillel**
Department of Computer Science
The Hebrew university of Jerusalem
aharonbh@cs.huji.ac.il

**Daphna Weinshall**
Department of Computer Science
The Hebrew university of Jerusalem
daphna@cs.huji.ac.il

## Abstract

We address the problem of sub-ordinate class recognition, like the distinction between different types of motorcycles. Our approach is motivated by observations from cognitive psychology, which identify parts as the defining component of basic level categories (like motorcycles), while sub-ordinate categories are more often defined by part properties (like 'jagged wheels'). Accordingly, we suggest a two-stage algorithm: First, a relational part based object model is learnt using unsegmented object images from the inclusive class (e.g., motorcycles in general). The model is then used to build a class-specific vector representation for images, where each entry corresponds to a model's part. In the second stage we train a standard discriminative classifier to classify subclass instances (e.g., cross motorcycles) based on the class-specific vector representation. We describe extensive experimental results with several subclasses. The proposed algorithm typically gives better results than a competing one-step algorithm, or a two stage algorithm where classification is based on a model of the sub-ordinate class.

## 1 Introduction

Human categorization is fundamentally hierarchical, where categories are organized in tree-like hierarchies. In this organization, higher nodes close to the root describe inclusive classes (like vehicles), intermediate nodes describe more specific categories (like motorcycles), and lower nodes close to the leaves capture fine distinctions between objects (e.g., cross vs. sport motorcycles). Intuitively one could expect such hierarchy to be learnt either bottom-up or top-down (or both), but surprisingly, this is *not* the case. In fact, there is a well defined intermediate level in the hierarchy, called *basic level*, which is learnt first [11]. In addition to learning, this level is more primary than both more specific and more inclusive levels, in terms of many other psychological, anthropological and linguistic measures.

The primary role of basic level categories seems related to the structure of objects in the world. In [13], Tversky & Hemenway promote the hypothesis that the explanation lies in the notion of parts. Their experiments show that basic level categories (like cars and flowers) are often described as a combination of distinctive parts (e.g., stem and petals), which are mostly unique. Higher (super-ordinate and more inclusive) levels are more often described by their function (e.g., 'used for transportation'), while lower (sub-ordinate and more specific) levels are often described by part properties (e.g., red petals) and other fine details. These points are illustrated in Fig. 1.

This computational characterization of human categorization finds parallels in computer vision and machine learning. Specifically, traditional work in pattern recognition focused on discriminating vectors of features, where the features are shared by all objects, with different values. If we make the analogy between features and parts, this level of analysis is appropriate for sub-ordinate categories.

*Appeared in *Advances in Neural Information Processing Systems (NIPS)*, MIT Press, Dec 2006.

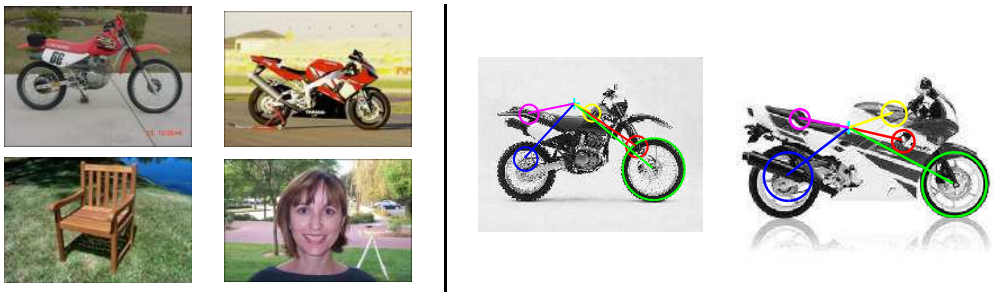

Figure 1: **Left** Examples of sub-ordinate and basic level classifi cation. *Top row:* Two motorcycle subordinate classes, sport (right) and cross (left). As members of the same basic level category, they share the same part structure. *Bottom row:* Objects from different basic level categories, like a chair and a face, lack such natural part correspondence. **Right**. Several parts from a learnt motorcycle model as detected in cross and sport motorcycle images. Based on the part correspondence we can build ordered vectors of part descriptions, and conduct the classifi cation in this shared feature space. (Better seen in color)

In this level different objects share parts but differ in the parts' values (e.g., red petals vs. yellow petals); this is called 'modified parts' in [13].

This discrimination paradigm cannot easily generalize to the classification of basic level objects, mostly because these objects do not share common informative parts, and therefore cannot be efficiently compared using an ordered vector of fixed parts. This problem is partially addressed in a more recent line of work (e.g., [5, 6, 2, 7, 9]), where part-based generative models of objects are learned directly from images. In this paradigm objects are modeled as a set of parts with spatial relations between them. The models are learnt and applied to images, which are represented as unordered feature sets (usually image patches). Learning algorithms developed within this paradigm are typically more complex and less efficient than traditional classifiers learnt in some fixed vector space. However, given the characteristics of human categorization discussed above, this seems to be the correct paradigm to address the classification of basic level categories.

These considerations suggest that sub-ordinate classification should be solved using a two stage method: First we should learn a generative model for the basic category. Using such a model, the object parts should be identified in each image, and their descriptions can be concatenated into an ordered vector. In a second stage, the distinction between subordinate classes can be done by applying standard machine learning tools, like SVM, to the resulting ordered vectors. In this framework, the model learnt in stage 1 is used to solve the correspondence problem: features in the same entry in two different image vectors correspond since they implement the same part. Using this relatively high level representation, the distinction between subordinate categories may be expected to get easier.

Similar notions, of constructing discriminative classifiers on top of generative models, have been recently proposed in the context of object localization [10] and class recognition [7]. The main motivation in these papers was to provide discriminative power to a generative model, optimized by maximum likelihood. Thus the discriminative classifier for a class in [7, 10] uses a generative model of the same class as a representation scheme.[1] In contrast, in this work we use a recent learning algorithm, which already learns a generative relational model of basic categories using a discriminative boosting technique [2]. The new element in our approach is in the learning of a model of one class (the more general basic level category) to allow the efficient discrimination of another class (the more specific sub-ordinates).

Thus our main contribution lies the use of objcet hierarchy, where we represent sub-ordinate classes using models of the more general, basic level class. The approach relies on a specific form of knowledge transfer between classes, and as such it is an instance of the 'learning-to-learn' paradigm. There are several potential benefits to this approach. First and most important is improved accuracy, especially when training data is scarce. For an under-sampled sub-ordinate class, the basic level model can be learnt from a larger sample, leading to a more stable representation for the second stage

[1] An exception to this rule is the Caltech 101 experiment of [7], but there the discriminative classifi ers for all 101 classes relies on the same two arbitrary class models.

SVM and lower error rate. A second advantage becomes apparent when scalability is considered: A system which needs to discriminate between many subordinate classes will have to learn and keep considerably less models (only one for each basic level class) if built according to our proposed approach. Such a system can better cope with new subordinate classes, since learning to identify a new class may rely on existing basic class models.

Typically the learning of generative models from unsegmented images is exponential in the number of parts and features [5, 6]. This significantly limits the richness of the generative model, to a point where it may not contain enough detail to distinguish between subclass instances. Alternatively, rich models can be learnt from images with part segmentations [4, 9], but obtaining such training data requires a lot of human labor. The algorithm we use in this work, presented in [2], learns from unsegmented images, and its complexity is linear in the number of model parts and image features. We can hence learn models with many parts, providing a rich object description. In section 3 we discuss the importance of this property.

We briefly describe the model learning algorithm in Section 2.1. The details of the two-stage method are then described in Section 2.2. In Section 3 we describe experiments with sub-classes from six basic level categories. We compare our proposed approach, called BLP (Basic Level Primacy), to a one-stage approach. We also compare to another two-stage approach, called SLP (Subordinate Level Primacy), in which discrimination is done based on a model of the subordinate class. In most cases, the results support our claim and demonstrate the superiority of the BLP method.

## 2 Algorithms

To learn class models, we use an efficient learning method briefly reviewed in Section 2.1. Section 2.2 describes the techniques we use for subclass recognition.

### 2.1 Efficient learning of object class models

The learning method from [2] learns a generative relational object model, but the model parameters are discriminatively optimized using an extended boosting process. The class model is learnt from a set of object images and a set of background images. Image $I$ is represented using an unordered feature set $F(I)$ with $N_f$ features extracted by the Kadir & Brady feature detector [8]. The feature set usually contains several hundred features in various scales, with considerable overlap. Features are normalized to uniform size, zero mean and unit variance. They are then represented using their first $15$ DCT coefficients, augmented by the image location of the feature and its scale.

The object model is a generative part-based model with $P$ parts (see example in Fig. 2b), where each part is implemented by a single image feature. For each part, its appearance, location and scale are modeled. The appearance of parts is assumed to be independent, while their location and scale are relative to the unknown object location and scale. This dependence is captured by a Bayesian network model, shown in Fig. 2a. It is a star-like model, where the center node is a 3-dimensional hidden node $C = (\vec{C}_l, C_s)$, with the vector $\vec{C}_l$ denoting the unknown object location and the scalar $C_s$ denoting its unknown scale. All the components of the part model, including appearance, relative location and relative log-scale, are modeled using Gaussian distributions with a (scaled) identity covariance matrix.

Based on this model and some simplifying assumptions, the likelihood ratio test classifier is approximated by

$$f(I) = \max_{C} \sum_{k=1}^{P} \max_{x \in F(I)} \log p(x|C, \theta^k) - \nu \tag{1}$$

This classifier compares the first term, which represents the approximated image likelihood, to a threshold $\nu$. The likelihood term approximates the image likelihood using the MAP interpretation of the model in the image, i.e., it is determined by the single best implementation of model parts by image features. This MAP solution can be efficiently found using standard message passing in time linear in the number of parts $P$ and the number of image features $N_f$. However, Maximum Likelihood (ML) parameter optimization cannot be used, since the approximation permits part rep-

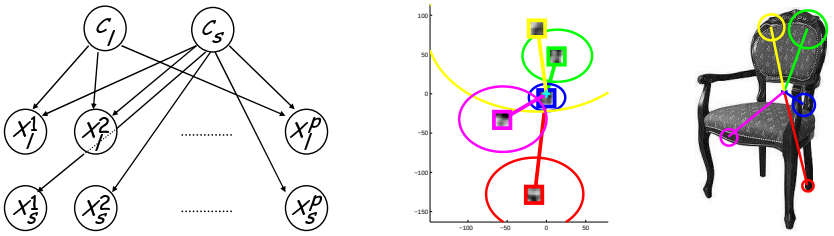


Figure 2: **Left** A Bayesian network specifying the dependencies between the hidden variables $C_l, C_s$ and the parts scale and location $X_l^k, X_s^k$ for $k = 1, .., P$. The part appearance variables $X_a^k$ are independent, and so they do not appear in this network. **Middle** The spatial relations between 5 parts from a learnt chair model. The cyan cross indicates the position of the hidden object center $c_l$. **Right** The implementations of the 5 parts in a chair image. (Better seen in color)

etition, and as a result the ML solution is vulnerable to repetitive choices of the same part. Instead, the model is optimized to minimize a discriminative loss function.

Specifically, labeling object images by $+1$ and background images by $-1$, the learning algorithm tries to minimize the exp loss of the margin, $L(f) = \sum_{i=1}^{N} \exp(-y_i f(I_i))$, which is the loss minimized by the Adaboost algorithm [12]. The optimization is done using an extended 'relational' boosting scheme, which generalizes the boosting technique to classifiers of the form (1).

In the relational boosting algorithm, the weak hypotheses (summands in Eq. (1)) are not merely functions of the image $I$, but depend also on the hidden variable $C$, which captures the unknown location and scale of the object. In order to find good part hypotheses, the weak learner is given the best current estimate of $C$, and uses it to guide the search for a discriminative part hypothesis. After the new part hypothesis is added to the model, $C$ is re-inferred and the new estimate is used in the next boosting round. Additional tweaks are added to improve class recognition results, including a gradient descent weak learner and a feedback loop between the optimization of the a weak hypothesis and its weight.

### 2.2 Subclass recognition

As stated in the introduction, we approach subclass recognition using a two-stage algorithm. In the first stage a model of the basic level class is applied to the image, and descriptors of the identified parts are concatenated into an ordered vector. In the second stage the subclass label is determined by feeding this vector into a classifier trained to identify the subclass. We next present the implementation details of these two stages.

**Class model learning** Subclass recognition in the proposed framework depends on part consistency across images, and it is more sensitive to part identification failures than the original class recognition task. Producing an informative feature vector is only possible using a rich model with many stable parts. We therefore use a large number of features ($N_f = 400$) per image, and a relatively fine grid of $C$ values, with $10 \times 10$ locations over the entire image and 3 scales (a total of $N_c = 300$ possible values for the hidden variable $C$). We also learn large models with $P = 60$ parts.[2] Note that such large values for $N_f$ and $P$ are not possible in a purely generative framework such as [5, 6] due to the prohibitive computational learning complexity of $O(N_f^P)$.

In [2], model parts are learnt using a gradient based weak learner, which tends to produce exaggerated part location models to enhance its discriminative power. In such cases parts are modeled as being unrealistically far from the object center. Here we restrict the dynamics of the location model in order to produce more realistic and stable parts. In addition, we found out experimentally that when the data contains object images with rich backgrounds, performance of subclass recognition and localization is improved when using models with increased relative location weight. Specifically, a part hypothesis in the model includes appearance, location and scale components with relative weights $\lambda_i/(\lambda_1 + \lambda_2 + \lambda_3),\ \ i = 1, 2, 3$, learnt automatically by the algorithm. We multiply

[2] In comparison, class recognition in [2] was done with $N_f = 200$, $N_c = 108$ and $P = 50$.

$\lambda_2$ of all the parts in the learnt model by a constant factor of $10$ when learning from images with rich background. Probabilistically, such increase of $\lambda_2$ amounts to smaller location covariance, and hence to stricter demands on the accuracy of the relative locations of parts.

**Subclass discrimination** Given a learnt object model and a new image, we match for each model part the corresponding image feature which implements it in the MAP solution. We then build the feature vector, which represents the new image, by concatenating all the feature descriptors implementing parts $1, ..P$. Each feature is described using a 21-dimensional descriptor including:

- The $15$ DCT coefficients describing the feature.
- The relative (x,y) location and log-scale of the feature (relative to the computed MAP value of $C$).
- A normalized mean of the feature $(m - \hat{m})/std(m)$ where $m$ is the feature's mean (over feature pixels), and $\hat{m}, std(m)$ are the empirical mean and std of $m$ over the $P$ parts in the image.
- A normalized logarithm of feature variance $(v - \hat{v})/std(v)$ with $v$ the logarithm of the feature's variance (over feature pixels) and $\hat{v}, std(v)$ the empirical mean and std of $v$ over image parts.
- The log-likelihood of the feature (according to the part's model).

In the end, each image is represented by a vector of length $21 \times P$. The training set is then normalized to have unit variance in all the dimensions, and the standard deviations are stored in order to allow for identical scaling of the test data. Vector representations are prepared in this manner for a training sample including objects from the sub-ordinate class, objects from other sub-ordinate classes of the same basic category, and background images. Finally, a linear SVM [3] is trained to discriminate the target subordinate class images from all other images.

## 3 Experimental results

**Methods:** In our experiments, we regard subclass recognition as a binary classification problem in a retrieval scenario. Specifically, The learning algorithm is given a sample of background images, and a sample of unsegmented class images. Images are labeled by the subclass they represent, or as background if they do not contain any object from the inclusive class. The algorithm is trained to identify a specific subclass. In the test phase, the algorithm is given another sample from the same distribution of images, and is asked to identify images from the specific subclass.

Several methodological problems arise in this scenario. First, subclasses are often not mutually exclusive [13], and in many cases there are borderline instances which are inherently ambiguous. This may lead to an ill-defined classification problem. We avoid this problem in the current study by filtering the data sets, leaving only instances with clear-cut subclass affiliation. The second problem concerns performance measurements. The common measure used in related work is the equal error rate of the ROC curve (denoted here EER), i.e., the error obtained when the rate of false positives and the rate of false negatives are equal. However, as discussed in [1], this measure is not well suited for a detection scenario, where the number of positive examples is much smaller than the number of negative examples. A better measure appears to be the equal error rate of the recall-precision curve (denoted here RPC). Subclass recognition has the same characteristics, and we therefore prefer the RPC measure; for completeness, and since the measures do not give qualitatively different results, the EER score is also provided.

**The algorithms compared:** We compare the performance of the following three algorithms:

- *Basic Level Primacy (BLP)* - The two-stage method for subclass recognition described above, in which a model of the basic level category is used to form the vector representation.
- *Subordinate level primacy (SLP)* - A two-stage method for subclass recognition, in which a model of the sub-ordinate level category is used to form the vector representation.
- *One stage method* - The classification is based on the likelihood obtained by a model of the sub-ordinate class.

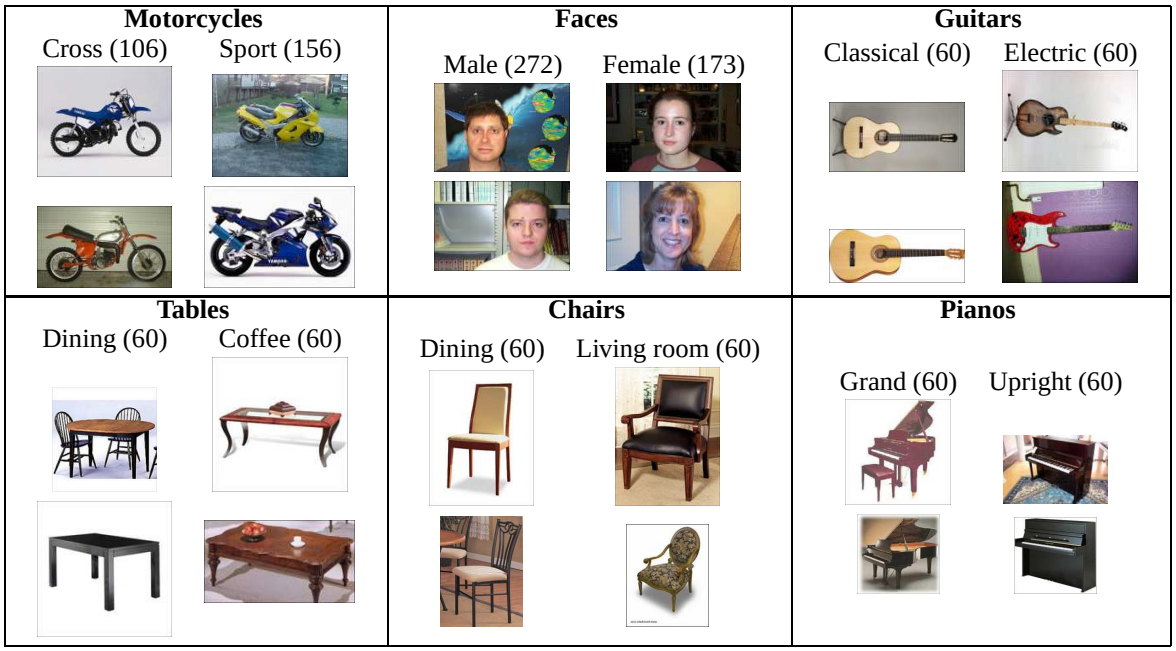


Figure 3: Object images from the subclasses learnt in our experiments. We used 12 subclasses of 6 basic classes. The number of images in each subclass is indicated in the parenthesis next to the subclass name. Individual faces were also considered as subclasses, and the males and females subclasses above include a single example from 4 such individuals.

The three algorithms use the same training sample in all the experiments. The class models in all the methods were implemented using the algorithm described in Section 2.1, with exactly the same parameters (reported in section 2.2). This algorithm is competitive with current state-of-the-art methods in object class recognition [2].

The third and the second method learn a different model for each subordinate category, and use images from the other sub-ordinate classes as part of the background class during model learning. The difference is that in the third method, classification is done based on the model score (as in [2]), and in the second the model is only used to build a representation, while classification is done with an SVM (as in [7]). The first and second method both employ the distinction between a representation and classification, but the first uses a model of the basic category, and so tries to take advantage of the structural similarity between different subordinate classes of the same basic category.

**Datasets** We have considered 12 subordinate classes from 6 basic categories. The images were obtained from several sources. Specifically, we have re-labeled subsets of the Caltech Motorcycle and Faces database[3], to obtain the subordinates of sport and cross motorcycles, and male and female faces. For these data sets we have increased the weight of the location model, as mentioned in section 2.2. We took the subordinate classes of grand piano and electric guitar from the Caltech 101 dataset [4] and supplemented them with classes of upright piano and classical guitar collected using google images. Finally, we used subsets of the chairs and furniture background used in [2][5] to define classes of dining and living room chairs, dining and coffee tables. Example images from the data sets can be seen in Fig. 3. In all the experiments, the Caltech office background data was used as the background class. In each experiment half of the data was used for training and the other half for test.

In addition, we have experimented with individual faces from the Caltech faces data set. In this experiment each individual is treated as a sub-ordinate class of the Faces basic class. We filtered the

[3] Available at http://www.robots.ox.ac.uk/ vgg/data.html.
[4] Available at http://www.vision.caltech.edu/feifeili/Datasets.htm
[5] Available at http://www.cs.huji.ac.il/ aharonbh/#Data.

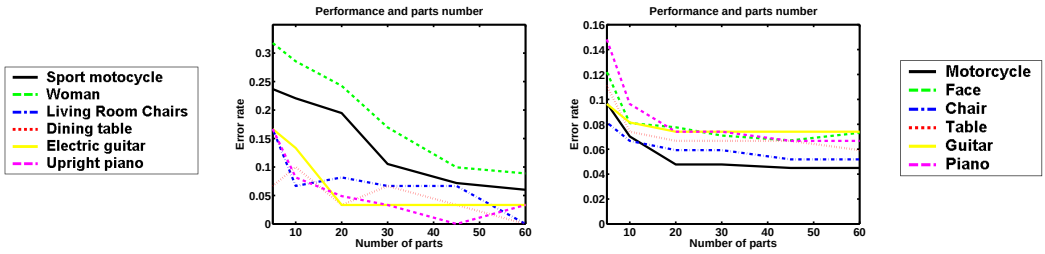


Figure 4: **Left:** RPC error rates as a function of the number of model parts $P$ in the two-stage BLP method, for $5 \leq P \leq 60$. The curves are presented for 6 representative subclasses, one from each basic level category presented in Fig. 3 **Right:** classification error of the first stage classifier as a function of $P$. This graph reports errors for the 6 basic level models used in the experiments reported on the left graph. In general, while adding only a minor improvement to inclusive class recognition, adding parts beyond 30 significantly improves subclass recognition performance.

faces data to include only people which have at least 20 images. There were 19 such individuals, and we report the results of these experiments using the mean error.

**Classification results** Table 1 summarizes the classification results. We can see that both two-stage methods perform better than the one-stage method. This shows the advantage of the distinction between representation and classification, which allows the two-stage methods to use the more powerful SVM classifier. When comparing the two two-stage methods, BLP is a clear winner in 7 of the 13 experiments, while SLP has a clear advantage only in a single case. The representation based on the basic level model is hence usually preferable for the fine discriminations required. Overall, the BLP method is clearly superior to the other two methods in most of the experiments, achieving results comparable or superior to the others in 11 of the 13 problems. It is interesting to note that the SLP and BLP show comparable performance when given the individual face subclasses. Notice however, that in this case BLP is far more economical, learning and storing a single face model instead of the 19 individual models used by SLP.

| **Subclass** | **One stage method** | | **Subordinate level primacy** | | **Basic level primacy** | |
|---|---|---|---|---|---|---|
| Cross motor. | 14.5 | (12.7) | 9.9 | (3.5) | **5.5** | **( 1.7)** |
| Sport motor. | 10.5 | (5.7) | 6.6 | (5.0) | **4.6** | **(2.6)** |
| Males | **20.6** | **(12.4)** | 24.7 | (19.4) | 21.9 | (16.7) |
| Females | 10.6 | (7.1) | 10.6 | (7.9) | **8.2** | **(5.9)** |
| Dining chair | 6.7 | (3.6 ) | **0** | **(0)** | **0** | **(0)** |
| Living room chair | 6.7 | (6.7) | **0** | **(0)** | **0** | **(0)** |
| Coffee table | 13.3 | (6.2) | 8.4 | (6.7) | **3.3** | **(3.6)** |
| Dining table | 6.7 | (3.6) | 4.9 | (3.6) | **0** | **(0)** |
| Classic guitar | 4.9 | (3.1) | **3.3** | **(0.5)** | 6.7 | (3.1) |
| Electric guitar | 6.7 | (3.6) | **3.3** | (3.6) | **3.3** | **(2.6)** |
| Grand piano | 10.0 | **(3.6)** | 10.0 | **(3.6)** | **6.7** | (4.0) |
| Upright piano | **3.3** | (3.6) | 10.0 | (6.7) | **3.3** | **(0.5)** |
| Individuals | 27.5* | (24.8)* | **17.9*** | (7.3)* | 19.2* | **(6.5)*** |

Table 1: Error rates (in percents), when separating subclass images from non-subclass and background images. The main numbers indicate equal error rate of the recall precision curve (RPC). Equal error rate of the ROC (EER) are reported in parentheses. The best result in each row is shown in bold. For the individuals subclasses, the mean over 19 people is reported (marked by $*$). Overall, the BLP method shows a clear advantage.

**Performance as a function of number of parts** Fig. 4 presents errors as a function of $P$, the number of class model parts. The graph on the left plots RPC errors of the two stage BLP method on 6 representative data sets. The graph on the right describes the errors of the first stage class models in the task of discriminating the basic level classes background images. While the performance of inclusive class recognition stabilizes after $\sim 30$ parts, the error rates in subclass recognition continue to drop significantly for most subclasses well beyond 30 parts. It seems that while later boosting rounds have minor contribution to class recognition in the first stage of the algorithm, the added parts enrich the class representation and allow better subclass recognition in the second stage.

## 4 Summary and Discussion

We have addressed in this paper the challenging problem of distinguishing between subordinate classes of the same basic level category. We showed that two augmentations contribute to performance when solving such problems: First, using a two-stage method where representation and classification are solved separately. Second, using a larger sample from the more general basic level category to build a richer representation. We described a specific two stage method, and experimentally showed its advantage over two alternative variants.

The idea of separating representation from classification in such a way was already discussed in [7]. However, our method is different both in motivation and in some important technical details. Technically speaking, we use an efficient algorithm to learn the generative model, and are therefore able to use a rich representation with dozens of parts (in [7] the representation typically includes 3 parts). Our experiments show that the large number of model parts i a critical for the success of the two stage method.

The more important difference is that we use the hierarchy of natural objects, and learn the representation model for a more general class of objects - the basic level class (BLP). We show experimentally that this is preferable to using a model of the target subordinate (SLP). This distinction and its experimental support is our main contribution. Compared with the more traditional SLP method, the BLP method suggested here enjoys two significant advantages. First and most importantly, its accuracy is usually superior, as demonstrated by our experiments. Second, the computational efficiency of learning is much lower, as multiple SVM training sessions are typically much shorter than multiple applications of relational model learning. In our experiments, learning a generative relational model per class (or subclass) required 12-24 hours, while SVM training was typically done in a few seconds. This advantage is more pronounced as the number of subclasses of the same class increases. As scalability becomes an issue, this advantage becomes more important.

## References


[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *ECCV*, 2002.

[2] A. Bar-Hillel, T. Hertz, and D. Weinshall. Effi cient learning of relational object class models. In *ICCV*, 2005.

[3] G.C. Cawley. MATLAB Support Vector Machine Toolbox [http://theoval.sys.uea.ac.uk/˜gcc/svm/toolbox].

[4] P. Feltzenswalb and D. Hutenlocher. Pictorial structures for object recognition. *IJCV*, 61:55–79, 2005.

[5] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale invariant learning. In *CVPR*, 2003.

[6] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for effi cient learning and exhaustive recognition. In *CVPR*, 2005.

[7] AD. Holub, M. Welling, and P. Perona. Combining generative models and fi sher kernels for object class recognition. In *International Conference on Computer Vision (ICCV)*, 2005.

[8] T. Kadir and M. Brady. Scale, saliency and image description. *IJCV*, 45(2):83–105, November 2001.

[9] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV workshop on statistical learning in computer vision*, 2004.

[10] Fritz M., Leibe B., Caputo B., and Schiele B. Integrating representative and discriminative models for object category detection. In *ICCV*, pages 1363–1370, 2005.

[11] E. 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.

[12] R.E. Schapire and Y. Singer. Improved boosting using confi dence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[13] B. Tversky and K. Hemenway. Objects, parts, and categories. *Journal of Experimental Psychology: General*, 113(2):169–197, 1984.

# Hierarchical Neural Networks Feature Extraction for LVCSR system

*Fabio Valente*[1]*, Jithendra Vepa*[1]*, Christian Plahl*[2]*, Christian Gollan*[2]*, Hynek Hermansky*[1]*,Ralf Schlüter*[2]

[1]IDIAP Research Institute, CH-1920 Martigny, Switzerland
Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland
[2]Lehrstuhl für Informatik 6, Computer Science Department
RWTH Aachen University, Aachen, Germany
{*fabio.valente,jithendra.vepa,hynek.hermansky*}*@idiap.ch*
{*plahl,gollan,schlueter*}*@cs.rwth-aachen.de*

## Abstract

This paper investigates the use of a hierarchy of Neural Networks for performing data driven feature extraction. Two different hierarchical structures based on long and short temporal context are considered. Features are tested on two different LVCSR systems for Meetings data (RT05 evaluation data) and for Arabic Broadcast News (BNAT05 evaluation data). The hierarchical NNs outperforms the single NN features consistently on different type of data and tasks and provides significant improvements w.r.t. respective baselines systems. Best results are obtained when different time resolutions are used at different level of the hierarchy.

**Index Terms**: Neural Network, feature extraction, LVCSR

## 1. Introduction

Data driven feature extraction aims at producing features for ASR using a statistical front-end. An efficient method of generating discriminative features is based on the use of Neural Networks (NNs) trained to classify phonetic targets [1]. Input to the NN is generally a section of the time-frequency plane. Different kinds of input have been investigated in the past: short temporal context input (9 frames PLP, see [1]), long temporal context input (HATS see [2]) and multiple time resolution input (MRASTA see [3]).

Data driven features provide complementary information to classical spectral features (PLP, MFCC) obtained from a temporal window of 30ms and considerable improvements in LVCSR tasks ([4]). Neural Network structure has been an active research topic as well: three-layers ([1]) and four-layers NN ([2]) has been studied and several hierarchical structures have been considered [5], [6].

In this paper we study two kinds of hierarchical neural network structures: the first structure based only on short time-frequency input and the second structure that combines long and short time-frequency input. We show that in the hierarchy, the second NN performs error correction on the most probable errors of the first one. Features are tested in two LVCSR systems trained on Meetings data and Arabic Broadcast News transcription, showing consistent improvement over single NN features and classical spectral features (PLP, MFCC).

The paper is organized as follows: section 2 describes the general idea of NNs based feature extraction, section 3 reports some preliminary experiments with a single NN, sections 4 and 5 describe experiments with hierarchal neural networks with short and long temporal context, sections 6.1 and 6.2 report results of hierarchical NNs feature extraction in two different LVCSR tasks: transcription of meetings data (RT05 evaluation data) and transcription of Arabic Broadcast News (BNAT05 evaluation data).

## 2. NN based feature extraction

In a multi-class problem, NN can be trained so that the output approximates class posterior probabilities (see [8]). Generally, a three-layer NN structure is used but other topologies have been investigated as well. Output from third layer are then normalized using a softmax function so that the sum of outputs is one.

In speech recognition, targets are represented by phonetic units, thus NN estimates posterior distribution of a given phoneme. A speech segment can be turned into a *posteriogram* i.e. a representation of the posterior probability of phonemes for each time frame (e.g. figure 4). Ideally a well trained NN will activate an output unit when a given spectro-temporal pattern is presented as input. For instance, in [13] input consists of 9 consecutive PLP frames while in [3] it consists of one second long segment obtained from the time-frequency plane. In order to use NN features in the classical HMM system, they are first gaussianized using log transform and then transformed using a KLT transform: this technique is referred as TANDEM [1]. In LVCSR systems they are typically appended to spectral features [4].

## 3. Experiments with single net

In this section we briefly describe experiments with a single NN on meetings data. Database consists of about 100 hours of meetings recorded at different sites. The independent headset microphone channel is used. Phoneme set consists of 46 targets including silence. Training data are phonetically labeled using forced alignment through the AMI LVCSR system (see section 6.1 and [10]).

We trained a Neural Net on those data with a temporal context of 9 frames. Almost 25% of total frames are labeled as silence thus it is interesting to report frame error rate without silence. In Table 1, Frame Error Rate (FER) with silence and without silence are reported together with FER obtained considering the two and three-best output list.

| | w silence | w/o sil | 2 best w/o sil | 3 best w/o sil |
|---|---|---|---|---|
| FER | 34.6 % | 43.0% | 32.3 % | 26.3% |

Table 1: FER computed with silence, without silence and using the two highest output and the three highest output.

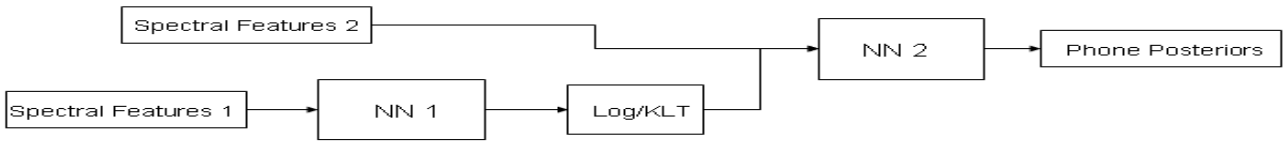


Figure 1: Hierarchical Neural Net processing

We notice that around 40% of the errors are generated by confusion in between two or three phonemes. For instance figure 2 plots confusion patterns for phonemes /*g*/. Several instances of phoneme /*g*/ are classified as phoneme /*k*/. Very similar error patterns are seen for all other phonemes in which the largest part of confusion is found in between two or three best competitors.

In literature, different approaches have been proposed for correcting confusion pattern in NN based phoneme classifiers. For instance, in [7] the use of error correction code is investigated. We show here that the use of a hierarchy of NNs provides an effective tool for correcting those kinds of errors.

## 4. Hierarchical Neural Network

We consider a hierarchical processing based on a cascade of NNs in which the second net has as input, posterior features (i.e. posteriors after Log/KLT transform) from the first net together with spectral features. Intuitively, the first net will activate a given output when a certain spectro-temporal pattern is seen as input. As described before, this will lead to a certain amount of errors, most of them in between two of three competing phonemes. The second net ideally will correct those kind of errors, using the spectro-temporal patterns that disambiguates in between different phonemes.

Figure 1 plots the general scheme for hierarchical NNs used for experiments. An initial set of spectral features is used for training a first net. Output is processed using a Log/KLT transform. New features are used for training a second neural net together with a second set of spectral features.

We investigate two different schemes; in the first one, already investigated in [6], we build the hierarchy using only short-context time-frequency input i.e. a block of 9 frames PLP features augmented with dynamic features. We refer to this as Hierarchical TANDEM. In the second scheme, we consider both long temporal context features (one second) and short-context features. We refer to this as Hierarchical MRASTA.

| | w silence | w/o sil | 2 best w/o sil | 3 best w/o sil |
|---|---|---|---|---|
| NN | 34.6 % | 43.0% | 32.3 % | 26.3% |
| HNN 1 | 29.2 % | 35.9 % | 27.5 % | 22.7 % |
| HNN 2 | 27.0 % | 33.0 % | 25.5 % | 21.1 % |

Table 2: FER computed with silence, without silence and using the two highest output and the three best outputs for single NN, and hierarchy of two and three NNs with 9-frame PLP input.

In Table 2, we reports FER for the single NN, and for hierarchical TANDEM with two and three NNs. Input is 9 frames PLP features. 5.5% absolute FER reduction is obtained by the second NN and further 2% is obtained by the third. It is also interesting to notice that the difference between the FER of the best output and of three highest outputs is progressively reduced. Figure 3 plots confusion patterns for phonemes /*g*/ across different level of the hierarchy: the largest gain in performance is obtained against most confusing phonemes from the previous NN. This behavior is observed for *all* the phonemes i.e. there is a reduction in frame error rate for every phoneme in the set.

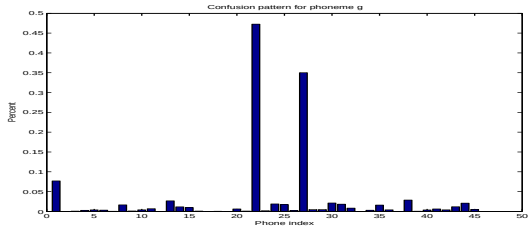


Figure 2: Confusion pattern for phoneme /*g*/; several instances of /*g*/ are classified as /*k*/

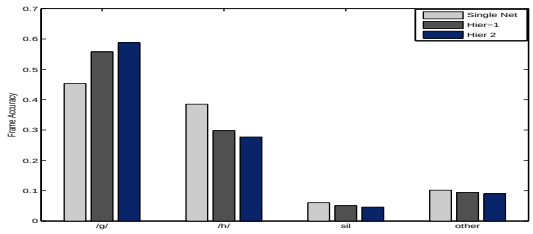


Figure 3: Confusion pattern for phonemes /*g*/ for the three different level of hierarchy.

An interesting effect of such a hierarchy (already described in [6]) is that at each layer the acoustic context is progressively increased: if the first NN has a temporal context of 9 frames, the second NN will use a temporal context of 9+8 frames. In next section, we investigate the use of a hierarchy of NN using directly different temporal context inputs at different layers.

## 5. Hierarchical NN with different temporal context input

MRASTA features are obtained giving as input to a NN a spectro-temporal cut of one second processed according to a zero mean multi-resolution filter ([3]). Thus MRASTA features are very robust to channel distortions.

Hierarchy of Neural Network can be used to incorporate different time resolution input at different level. In this section we investigate this framework. Considering schema of figure 1 the first set of spectral features has a temporal context

of one second while the second set of spectral features uses only 9-frames context. We refer to this method as Hierarchical MRASTA. Table 3 shows FER for single MRASTA and Hierarchical MRASTA.

| | w silence | w/o sil | 2 best w/o sil | 3 best w/o sil |
|---|---|---|---|---|
| NN | 36.3 % | 45.9% | 34.9 % | 28.4 % |
| HNN 1 | 28.9 % | 35.6 % | 27.7 % | 23 % |

Table 3: FER computed with silence, without silence and using the two highest output and the three best outputs for MRASTA and hierarchical MRASTA.

Results in terms of FER are close to those obtained in Table 2. Hierarchy with three levels doesn't further improve FER. The same behavior previously described can be noticed i.e. confusion between phonemes is drastically reduced. It is possible to visualize the difference between the two methods using a *posteriogram*. Figure 4 plot posteriograms for MRASTA (left) and hierarchical MRASTA (right). Posteriogram on the right is much smoother than the one on the left, achieving at the same time a lower FER.

# 6. LVCSR experiments

### 6.1. Meeting system

For investigation purposes, we run experiments on RT05 [9] data without concatenation with other features and compare with PLP features. The training data for this system comprises of individual headset microphone (IHM) data of four meeting corpora; the NIST (13 hours), ISL (10 hours), ICSI (73 hours) and a preliminary part of the AMI corpus (16 hours). Acoustic models are phonetically state tied triphone models trained using standard HTK maximum likelihood training procedures. The recognition experiments are conducted on the NIST RT05s [9] evaluation data. We use the reference speech segments provided by NIST for decoding. The pronunciation dictionary is same as the one used in AMI NIST RT05s system [10]. Juicer large vocabulary decoder [11] is used for recognition with a pruned trigram language model.

Table 4 reports results obtained using PLP (baseline system), TANDEM, MRASTA, hierarchical TANDEM and hierarchical MRASTA.

Out of the proposed feature set, hierarchical MRASTA is providing the best performance and reduces WER of 3% absolute compared to PLP features. As general remark the use of the hierarchy always improves w.r.t. the single net. Let us consider results in more details.

TANDEM features slightly outperforms PLP features on AMI and ICSI data. On the other hand, there is a consistent drop in performance in VT data which are particularly noisy . Let us consider now the effect of the hierarchy. When a second NN is used an average improvement of 2.2% absolute is obtained; this improvement is verified on all type of data. Thus second NN is improving recognition at both phoneme and word level when used for generating features. On the other hand, when a third NN is added, overall performance deteriorates of 0.6% probably because of over fitting to the data.

MRASTA features are designed to remove mean value in the modulation spectrum trough the use of a multi-resolution filter thus more robust to noise and distortions. Furthermore they use an acoustic context of one second. Overall performance of MRASTA is better than TANDEM features. Contrarily to TANDEM, on VT data, they hold performance comparable to PLP.

Most interesting results we obtained is based on using hierarchical MRASTA with different temporal context as described in section 5: an average improvement of 6% absolute is verified w.r.t single net MRASTA. Improvements are seen on all data sets in the RT05; 2% improvement is also observed on VT data where results for TANDEM are very poor. Furthermore this set of features outperforms by 3% classical PLP front-end.

### 6.2. Arabic BN system

In this section we investigate the use of hierarchical features in concatenation with spectral features in a LVCSR system for transcription of Arabic Broadcast News. As described in [14], the acoustic front end uses MFCC features with cepstral mean normalization. The MFCC features are augmented with a *voicedness feature* [12] and includes Vocal Tract Length Normalisation (VTLN). Features from nine consecutive frames are concatenated and a linear discriminative analysis (LDA) is used to reduce the feature dimensions. The Neural Network features were concatenated with the LDA-transformed MFCC baseline system. Acoustic models were triphone based Viterbi trained Gaussian mixture models (GMMs) with a global pooled covariance matrix. The triphones are top down clustered using CART, rendering 4501 generalized triphone states with cross-word context.

The training corpus consists 120 hours of speech, derived from the FBIS (30h) and the Arabic TDT4 (60h) corpus. 30h of additional data is taken from the first two quarter releases of the first year (Y1Q1, Y1Q2) of the GALE project. All data consists of Arabic Broadcast News. Most available training material for the Arabic speech recognition don't inlcude diacritics. Ignoring theses diacritics increase the error rate ([16]). For this purpose the Buckwalter Arabic Morphological Analyser [1] is used to vowelise the transcriped data. Because not all words are mapped to a diacritic form we used a data-driven approach known as Grapheme-to-Phoneme conversion [15]. In the training process for the Grapheme-to-Phoneme conversion a mapping beetween the orthographic form of a word and its phonetic transcription is build. These models are used to create the transcription of unknown words.

The language model used for the experiments is derived from the Gigaword Arabic, the Arabic TDT4 and from the FBIS corpus. Additional data is taken from the Y1Q1 and Y1Q2 corpora of the GALE project. The language model is a bigram with a vocabulary of 256k words.

The BNAT05 evaluation corpus has been used for evaluation purpose Table 5 summarizes the improvement produced by using the concatenated Neural Network features. As shown in Table 5, the single net features improves the recognition rate by 0.3% absolute for TANDEM and by 0.6% absolute for MRASTA. Using the hierarchical features there is an additional improvement by 0.6% absolute towards the single MRASTA and TANDEM approach. Overall, the best results are archieved using the hierarchical MRASTA feature approach and results are consistent with what is observed on meeting data.

# 7. Conclusion

In this paper we investigate the use of a hierarchy of Neural Networks for performing data driven feature extraction. Two different framework are proposed: one with the same temporal context at both level of the hierarchy (Hierarchical TANDEM) and one with changing context (Hierarchical MRASTA).

[1] http://www.qamus.org/morphology.htm

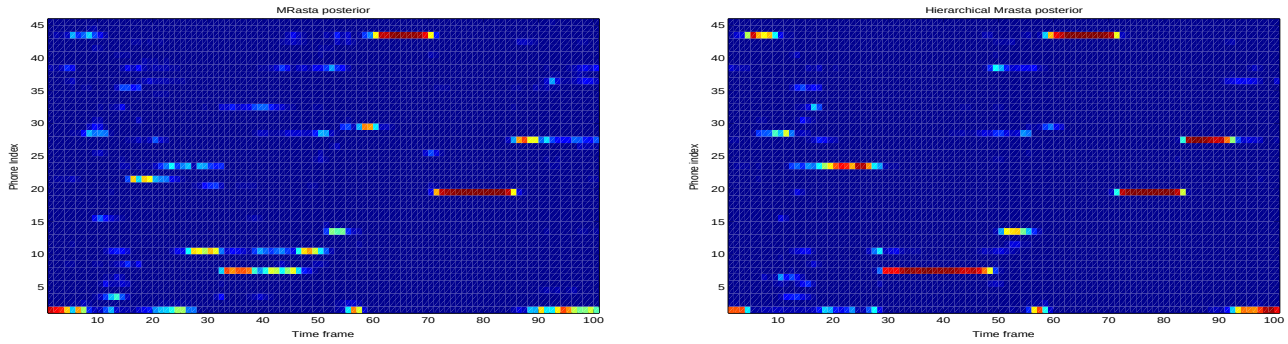


Figure 4: Posteriograms for MRASTA (left) and Hierarchical MRASTA (right)

| Features | TOT | AMI | CMU | ICSI | NIST | VT |
|---|---|---|---|---|---|---|
| PLP+D+A | 42.4 | 42.8 | 40.5 | 31.9 | 51.1 | 46.8 |
| TANDEM | 46.6 | 41.4 | 43.7 | 31.3 | 54.5 | 64.9 |
| Hier TANDEM 1 | 44.4 | 39.6 | 42.3 | 28.9 | 51.5 | 62.0 |
| Hier TANDEM 2 | 45.0 | 40.5 | 44.4 | 29.4 | 51.1 | 61.9 |
| MRASTA | 45.9 | 48.0 | 41.9 | 37.1 | 54.4 | 48.8 |
| Hier MRASTA 1 | 39.4 | 38.1 | 36.9 | 28.2 | 48.0 | 46.9 |

Table 4: WER for Meeting data.

| Features | WER |
|---|---|
| MFCC | 23.6 |
| MFCC + TANDEM | 23.3 |
| MFCC + MRASTA | 23.0 |
| MFCC + Hier TANDEM | 22.7 |
| MFCC + Hier MRASTA | 22.4 |

Table 5: WER for Arabic BN task.

Consistent reduction in Frame Error Rate are observed for both framework. The second net has the property of correcting the most confusable patterns of the first one. NN based features are investigated in two LVCSR systems for transcription of meetings and Arabic BN. Hierarchical MRASTA method is showing the best performance in both systems providing consistent improvements w.r.t. respective baseline systems. Changing time resolution across different level of the hierarchy seems to be very effective and must be further addressed in future works.

## 8. Acknowledgments

This material is based upon work supported by the EU under the grant DIRAC IST 027787 and by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023 . Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). We thank SRI International for the help to build up the Arabic recognition system, especially Dimitra Vergyri and Andreas Stolke for the closed collaboration. We thank Thomas Hain and the AMI ASR team for their help with the meeting system.

## 9. References

[1] Hermansky H., Ellis D.P.W. and Sharma S. , “Tandem connectionist feature extraction for conventional HMM systems”, Proceedings of ICASSP 2000.

[2] Chen B., Zhu Q., and Morgan N., “ Learning Long-Term Temporal Features in LVCSR Using Neural Networks”, Proceedings of ICSLP 2004.

[3] Hermansky H. and Fousek P., ”Multi-resolution RASTA filtering for TANDEM-based ASR.”, Proceedings of Interspeech 2005.

[4] Zhu Q., Chen B., Morgan N., and Stolcke A., “On using MLP features in LVCSR”, Proceedings of ICSLP 2004.

[5] Sivadas S. and Hermansky H., ”Hierarchical Tandem Feature Extraction”, Proceedings of ICASSP-2002.

[6] Schwarz P., Matejka P., Cernock J.,”Hierarchical structures of neural networks for phoneme recognition”, Proceedings of ICASSP 2006.

[7] Hagen H. and Bourlard H., “Error Correcting Posterior Combination for Robust Multi-Band Speech Recognition”, in Proceedings of EUROSPEECH, 2001

[8] Bourlard, H. and Wellekens, C.J. (1989), “Speech Pattern Discrimination and Multilayer Perceptrons” Computer, Speech and Language (Academic Press), vol. 3, pp. 1-19.

[9] http://www.nist.gov/speech/tests/rt/rt2005/spring/

[10] Hain, T. et al, “The 2005 AMI System for the Transcription of Speech in Meetings” NIST RT05 Workshop, 2005, Edinburgh, UK.

[11] Moore, D et al. “Juicer: A weighted finite state transducer speech coder” Proc. MLMI 2006 Washington DC.

[12] Zolnay A., Schlter R., Ney H. “Robust Speech Recognition using a Voiced-Unvoiced Feature”, Proc. of ICSLP, Denver, CO, Vol. 2, pp. 1065–1068, Sept. 2002.

[13] Bourlard H. and Morgan N.,”Connectionist Speech Recognition - A Hybrid Approach”,Kluwer Academic Publishers, 1994.

[14] Lööf J. et al. “The 2006 RWTH Parliamentary Speeches Transcription System”, In Proceedings of ICSLP 2006.

[15] Bisani M., Ney H. “Multigram-based grapheme-to-phoneme conversion for LVCSR”, Proc. Eurospeech, 2003.

[16] Vergyri D., Kirchhoff K. “Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition”, COLING Workshop on Arabic-script Based Languages , 2004.