A neural network model of hierarchical category development

Chris Gorman (cgorman@cs.otago.ac.nz)

Alistair Knott (alik@cs.otago.ac.nz)
Department of Computer Science, 133 Union St East
Dunedin, 9016 New Zealand

Abstract

Object recognition and categorization is a fundamental aspect of cognition in humans and animals. Models have been implemented around the idea that categories are sets of frequently co-occurring features. Out of these models a question has been raised, namely what is the mechanism by which we learn a hierarchically organized set of categories, including types and subtypes? In this paper we introduce such a model, the Dominant Property Assembly Network (DPAN). DPAN uses an unsupervised neural network to model an agent which develops a hierarchy of object categories based on highly correlated object features. Initially, the network generates representations of high-level object types by identifying commonly co-occurring sets of features. Over time, the network will start to use an inhibition of return (IOR) operation to examine the features of a categorized object that make it unusual as an instance of its identified category. The result is a network which, early in training, represents classes of objects using coarse-grained categories and recognizes objects as members of these general classes, but eventually is able to recognize subtle differences between subtypes of objects within the broad classes, and represent objects using these more fine-grained categories.

Keywords: categorization; computational modeling; prototype theory

Introduction

Humans and animals begin developing classifications of objects starting from very soon after birth (Quinn, Slater, Brown, & Hayes, 2001) and throughout the rest of our lives. We are able recognize tens of thousands of different objects (Biederman, 1987; Brady, Konkle, & Alvarez, 2011), which is an extremely important evolutionary tool. It allows us to rapidly determine whether a token object is, for example, edible, dangerous, friendly, and so on (DiCarlo, Zoccolan, & Rust, 2012). It is embedded in so many aspects of our lives that we often don't give it a second thought. It provides us a way to compress as much information as possible with as little cognitive effort as possible (Rosch, 1999). Humans and animals learn to represent token objects as instances of object categories. The structure of categories has been explored extensively in cognitive science (see e.g. Rakison & Yermolayeva, 2010). Categories reflect several types of structures. A central idea is that categories represent **correlations** between the features of objects. We recognize that, for example, a token object is a "dog" because it has features shared with other things we call dogs. But another key idea is that categories emerge to represent types of objects that have distinctive properties. Exactly how these two apparently conflicting criteria coexist is the subject of much debate (Tyler & Moss, 2001). In the current paper we will propose a novel architecture that reconciles them.

This category system is hierarchically organized. **Basic level categories** like "chair" or "dog" afford the agent a cognitively efficient representation of the object (Mervis & Rosch, 1981). Above the basic level we have **superordinate categories**. Superordinate categories, like "furniture" or "animal," are difficult for agents to visualize and they contain several intra-category differences. For example, a car and a boat can both be considered in the "vehicles" superordinate category, but they have far fewer shared properties. In addition, the important correlations may be determined as much by culture as by the objects themselves (Liu, Golinkoff, & Sak, 2001). Below the basic level we have **subordinate categories**. This level contains categories which may not contain many functionally actionable differences when compared against their basic level category, such as "computer chair" or "pug," but rather simply provide more detailed information. Developmentally, subordinate categories are learned after basic-level categories (Rosch, Mervis, Gray, Johnson, & Boyes-Braem, 1976). Finally, beneath the subordinate level we have the level of token individuals which is able to differentiate particular, unique instances of categories from one-another, such as differentiating "a dog" from "my dog, Charles Barkley." This level includes spatiotemporal information in conjunction with other sensory input and is beyond the scope of this work. This paper focuses on the subordinate level of categorization and, to a lesser degree, the basic level.

When a person has minimal experience with a type of object[1], they tend to focus on the major features shared between members of that type before investigating the differences. For example, a child may categorize any animal with four legs as "dog" (Rakison & Yermolayeva, 2010). A young child can recognize categories, such as "dog" and "cat," from a series of commonly occurring features. Within these categories, the child is able to pick up on more frequently occurring, albeit more subtle, combinations of features, such as those shared by pugs. They also notice other sets of subtle, frequently occurring features within the same category, like those shared by spaniels. Due to the subtle nature of these regularities, they weren't noticed until after the basic-level category was solidified. If we assume that the mechanism which learns basic-level categories does so by identifying the strongest correlations among object features, then by definition we must seek a different explanation for the emergence of subordinate-level categories.

[1]For the remainder of the paper we refer to a **category** or **type** as an internal representation of commonly co-occurring features and a **token object** as a particular instance of a category.

Most modern work on computational object categorization has focused on the basic and individual levels (Riesenhuber & Poggio, 2000; Winn, Criminisi, & Minka, 2005; Galleguillos, Rabinovich, & Belongie, 2008). However, in recent years, a significant amount of work has been done in an attempt to recognize subordinate categories as well (Zhang, Gao, Xia, Dai, & Li, 2015; Farrell, Oza, Morariu, Darrell, & Davis, 2011; Yang, Bo, Wang, & Shapiro, 2012; Chai, Lempitsky, & Zisserman, 2013). Most of these models recognise subordinate categories by analysing the sub-parts of objects and discovering regularities in the identity and configuration of these sub-parts. However, the models do not pay so much attention to the question of *when* an agent begins to learn subordinate-level categories within a given basic-level category - that is, to the developmental trajectory of subcategory learning. We propose that there are particular circumstances which lead to an agent starting to learn subcategories, and that the process of learning subcategories is implemented through overtly scheduled cognitive operations that have correlates in surface language.

Our model of subcategory learning is implemented within a network called the **dominant property assembly network** (DPAN). DPAN is presented with the visual features of a series of token objects and begins learning internal representations of those objects' basic-level categories by identifying the strongest correlations amongst these features. This might lead to the emergence of categories "dog" and "cat", for example. However, in order to learn more subtle correlations identifying subordinate-level categories, we propose that the strongest correlations should be actively inhibited. We accomplish this through a cognitive operation that is a type of **inhibition of return** (IOR).

When presented with a token object, DPAN first classifies the object to activate an internal representation of its category. This internal representation is associated with a certain collection of features which identify dogs, for example. Having activated this representation, DPAN then **inhibits** the associated features, allowing it to focus on what makes this particular token object different or unusual. This idea of "inhibiting the winner" is found in several neural circuits and is often referred to as "inhibition of return." It was originally shown in spatial attention (Posner, Rafal, Choate, & Vaughan, 2007), where agents were shown to "inhibit orienting towards visual locations which have been previously attended". DPAN, however, operates on the domain of properties rather than spatial locations. The IOR operation can be understood as an operation that identifies a *property* of the currently attended object. The process of identifying properties is one that is readily reported in language, in **predicative sentences**. For instance, in the sentence "The dog is brown", the dog is predicated as having the property brown. It is interesting that object categories can feature both referentially and as predicates: for instance, in the sentence "The dog is a pug", "The dog" is a referential expression, but "a pug" is simply a property that is predicated of the dog (Partee, 1987). The difference between referential nominals and predicating nominals is still a matter of debate for linguists; our model of IOR in category learning will make a suggestion about this difference. Once training is completed, the network will have developed representations of object classes as well as subtypes of those classes. In our example, it would contain category representations of "dog" alongside "pug," "beagle," "spaniel," and "corgi."

The other key novelty in DPAN is the use of **per-unit learning rates**. DPAN uses localist units to represent highly correlated features. The method by which the network learns these correlations is explained in detail in a later section. Once a localist unit emerges which strongly and sufficiently represents a token object's category, the unit stops learning to prevent further input from altering its encoded category. To achieve this end, the network employs per-unit learning rates. These learning rates are associated with a particular localist unit, monotonically decreasing, and are based on a measure of the rate of change of the connections for their respective unit.

The use of a per-unit local learning rate is well established in the field of neural networks (Thimm, Moerland, & Fiesler, 1996; Bengio, Simard, & Frasconi, 1994; Becker & Le Cun, 1988; Schiffmann & Geffers, 1993). The notion of a learning rate which changes based on its objective performance is also well established (Senior, Heigold, Ranzato, & Yang, 2013; Renals, Morgan, Cohen, & Franco, 1992). Since the learning rate is a monotonically decreasing function, it doesn't cause a feedback loop. For example, if the learning rate updated as a function of the gradient of change, then it would simply raise itself up or lower itself down because the learning rate directly affects the weights' gradient of change. By allowing the weights to update using a constant value, we are able to more accurately measure the amount of learning each unit is accomplishing. Then, when the weight change gradient for that unit is low, we can confidently say that it is because the unit has completed its learning process and no longer needs to be updated.

The remainder of this paper is organized as follows. First we describe DPAN in detail, followed by an introduction to our experiments and walk through of their results. In the final section we present our conclusions.

Architecture

DPAN is effectively organized into three separate layers (Figure 1). The first layer is the **rich property complex** (RPC) which contains the raw object properties provided by the sensorimotor system. The next layer is the **dominant property assembly** (DPA). The DPA essentially provides a workspace for computations to be done on the RPC without permanent modification. When the network is presented with a token object, it first copies the information directly from the RPC into the DPA. Above the DPA layer lies the **conditional principal component analysis** (CPCA) units, introduced by O'Reilly and Munakata (2000), which constitute the localist property

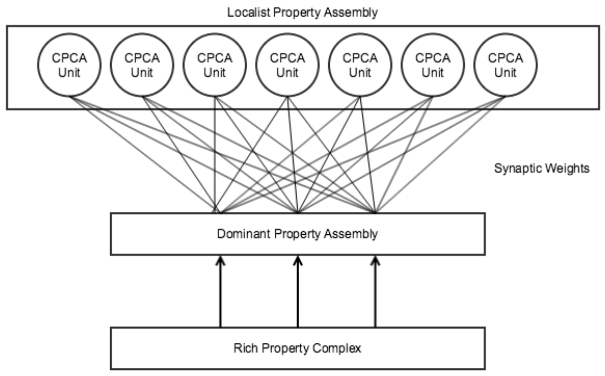


Figure 1: An overview of the DPAN architecture.

assembly (LPA) layer. The weights connect each feature in the DPA layer with each unit in the LPA layer. CPCA provides the core learning mechanism for the network and is explained in detail in the next section. The CPCA units in the LPA layer represent the basic level and subordinate categories.

CPCA

As we stated previously, the core learning of DPAN is accomplished via CPCA. CPCA is an unsupervised artificial neural network based on Hebbian learning which generates an internal model of strongly correlated features. The algorithm is explained in detail in (O'Reilly & Munakata, 2000), but we present a brief overview here as our method differs slightly from the original implementation. CPCA takes binary input and generates binary outputs using a competitive winner-take-all approach. It uses one layer of neurons fully connected to the input vector. The neurons produce output by calculating the weighted sum of of their inputs, selecting the one with the highest output, setting that value to 1 and the rest to 0. The weights are updated using Equation 1 where y_j is the activity of the unit, x_i is the input feature, w_{ij} is the weight between them, and α_j is a learning rate, between 0 and 1, of the unit. O'Reilly et al. provide a derivation proving that $w_{ij} = P(y_j = 1|x_i = 1)$. In a practical sense, the weight between a CPCA unit and an input vector increases when the unit is active and the input is 1, or decreases when the neuron is active and the input is 0. In DPAN, the weights are connected to the dominant property assembly rather than to the rich property complex. This allows us to manipulate the CPCA's input vector, during inhibition of return for example, while maintaining a reference to the original input data.

$$\Delta w_{ij} = \alpha_j y_j (x_i - w_{ij}) \tag{1}$$

The Training Algorithm

We start by defining a few terms. A **training item** is a combination of feature values originally presented to the RPC. A **training episode** encompasses all of the processing that is done on a single training item. If IOR is invoked, the episode may involve several **iterations**, i.e. an update of the network's weights. An **epoch** is a collection of training episodes such that each training item has been presented to the network once. The maximum number of epochs for each execution is represented by λ and the current epoch is represented by t. Training typically consists of multiple epochs where the list of training items has been shuffled each time. It's also worth mentioning that there should be at least enough CPCA units to represent the number of basic level and subordinate categories in the training data, but fewer than the number of features in the input vectors. The network is initialized with psuedorandom weights between 0.4 and 0.6. This allows each CPCA unit to have a decent chance of learning each of the objects without any major bias initially. At this point, we also initialize the learning rate for each unit. At the start of each training iteration, the RPC is copied into the DPA and the network calculates the activity of the CPCA units using Equation 2. The network then chooses the unit with the highest output $\max \vec{y}$, selects it as the winner, sets its output to 1 and the output of all other units to 0. Once the winner is selected, the weights for that unit $\vec{w}_j$ are updated using equation 1. The total change in weight is now measured to determine whether or not to begin IOR and to disable further learning on this unit. If the gradient of change is steeper than the threshold τ, the iteration is complete and the process begins again.

$$\vec{y} = \mathbf{W}^T \vec{x} \tag{2}$$

To calculate the gradient of change of the weight vector $\vec{w}_j$ at time t, DPAN allocates space for a temporary weight vector, $\vec{w}_j(t-1)$, updates the weights as normal, and then subtracts the current weight from the previous one, taking the absolute values (equation 3). $\vec{d}$ is now summed up to produce the scalar value of the total change the weight vector $\vec{w}_j$ underwent (equation 4). This sum is appended to a vector $\vec{\Omega}_j$ such that the contents of the vector are the total change in weight for each iteration. $\vec{\Omega}_j{}'$ is calculated such that it contains the gradient of $\vec{\Omega}_j$. Finally, the last two elements of $\vec{\Omega}_j{}'$ are subtracted from one another and if their difference is less than τ, we consider the gradient of change to be minimal.

$$\vec{d} = |\vec{w}_j(t) - \vec{w}_j(t-1)| \tag{3}$$

$$\sum_{k=1}^{|\vec{w}_j|} w_{ik} \tag{4}$$

If the network determines that there was minimal change, IOR begins. The DPA is now updated such that $\vec{x} = \mathbf{W}\vec{y}$, essentially copying the object prototype from the LPA into the DPA. The RPC is now subtracted from the DPA in the inhibition step, $\vec{x} := \vec{z} - \vec{x}$, and stored back into the DPA. At this stage, the DPA to now contains the difference between the prototype object and the actual token object. For example, if the token object was "pug" and the winning unit was "dog," the DPA would now contain the properties that pugs have which differentiate them from other dogs.

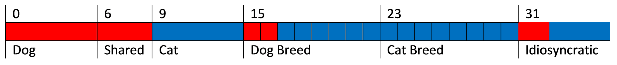
Since the IOR operation can run indefinitely, we must define stopping conditions. If, after the inhibition operation, the DPA doesn't contain anything "interesting," IOR ceases and no learning is done. That is, if there is minimal difference between the token object and the winning unit's property assembly, there is nothing for the network to learn, so it stops. For example, if the winning unit near-perfectly represented the token object there would be very little difference between the RPC and the prototype object. If there is something interesting left in the DPA, the network learns in the same way as before: the activity of each unit is calculated, a winner is selected, and that unit's weights are updated. The only differences this time are that the LPA layer performs a self-inhibition operation which prevents previous winners from this iteration to win again and that before the weights are updated, the contents of the DPA are replaced with the contents of the RPC; the network chooses a winner based on the unique properties, but then trains on the entire property complex. After the weights are updated, the gradient of change for the new winning unit is calculated. If the change was large, the inhibition loop finishes and the training iteration for this input is complete. Otherwise, the loop repeats, allowing more interesting features to bubble up.

The initial result of training (Stage 1) is that the coarse-grained object categories are learned. At this stage, when an object is presented to the network, the winning LPA unit represents a supertype. Now the network starts to systematically inhibit the units representing coarse-grained types after they are activated, allowing other units in the network to develop representations of finer-grained subtypes. When these are first learned (Stage 2), the network activates *first* a unit representing a supertype, and *then* (after IOR) a unit representing a subtype. After even more training, the subtype units learn better representations of the training objects than the supertype units, and the network activates a subtype representation *as its first response* (Stage 3). In linguistic terms, the response at Stage 1 could be rendered "that is a dog", at Stage 2, "That dog is a pug", and at Stage 3, "That is a pug".

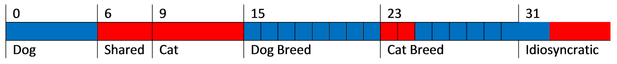
Experimental Setup

Input Data

DPAN was trained on a set of binary input vectors, each representing a token object containing 36 features. Each element of the vector encodes a distinct, abstract property of the token object. When a bit is set to 1 it indicates the presence of a property and when it is set to 0 it indicates the absence of that property. Each input vector represents one of four cat or four dog breeds. Figure 2 describes the layout of the input bits and provide examples of token individuals. All four breeds of dog shared the same set of "generic dog" properties and all four breeds of cat shared the same set of "generic cat" properties. There is an overlap of three bits between these two sets representing properties shared between all dogs and cats. The next 16 bits represent breed-specific properties, e.g. a wrinkled face and short snout for a pug. Each breed is identified



(a) A token pug



(b) A token tabby

Figure 2: Two vectors representing token individuals. Red indicates a value of 1 and blue a value of 0.

by which two of those bits are active, so there are eight bits for the dog breeds (pug, beagle, spaniel, and corgi) and eight bits for the cat breeds (tabby, maine coon, siamese, and persian). Finally, there is a set of five idiosyncratic property bits which are each uniformly randomly set to 0 or 1 for each token individual. These bits represent weak, uncorrelated properties, a unique color or a marking of some kind for example.

Training Runs

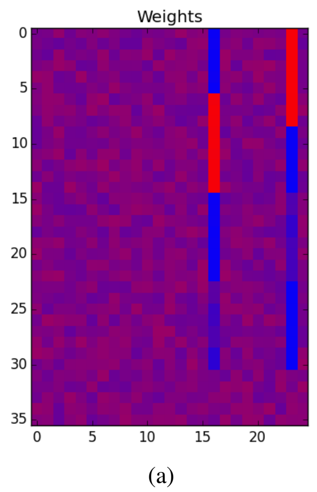
DPAN was trained on a set of 5000 input vectors. Each of the eight breeds were equally distributed in the training set. The network was trained using the following input parameters:

- $|y| := 25$
- $\alpha_j := 0.02 \forall j \in y$
- $\tau := 0.000001$
- $\lambda := 500$

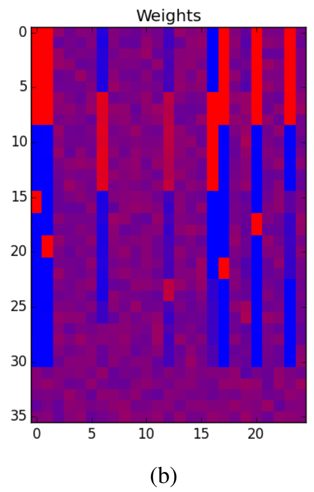
These parameters were selected based on empirical evidence showing they provide the best results for this particular dataset. The network produces output at arbitrary epoch and input intervals. The output of the network is a heat-map of the weight matrix, again where red indicates a value of 1 and blue a value of 0. The column vectors of the weight matrix represent the CPCA units and the row vectors represent the input features.

Results

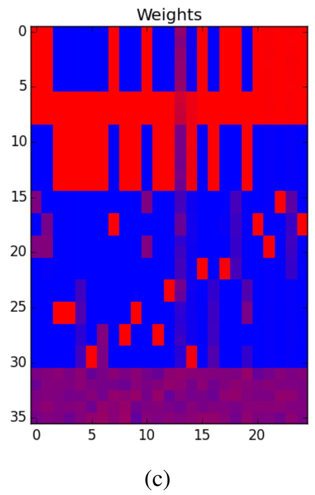
We present the results of the execution by examining three key stages of training. Early in training, DPAN learns localist representations of the basic-level categories for dog and cat, as shown in Figure 3a where units 23 and 16 represent dog and cat respectively. The basic-level units represent the highly correlated properties found in the dog and cat breeds while also maintaining the weak correlations of each subordinate-level category. For example, the network's localist dog unit encapsulates the common features associated with each dog it has been exposed to, but also maintains weak connections to the subtle correlations for each dog breed it has seen. At this stage of training, DPAN's each dog or cat it sees will activate its corresponding basic-level localist unit and reinforce these connections.



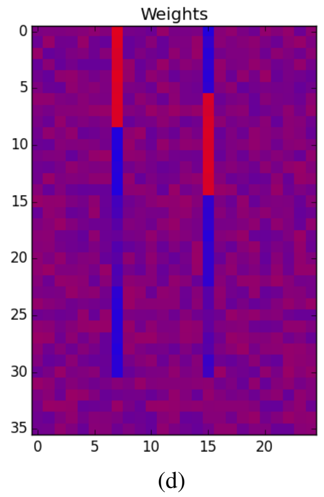
(a)



(b)



(c)



(d)

Figure 3: DPAN results: (a) aDPAN weight matrix during the first stage of training when basc-level categories are formed. (b) bDPAN weight matrix during the second stage of training. Subordinate-level units are beginning to emerge as a result of the IOR operation. (c) cDPAN weight matrix after the final stage of training. Each subordinate-level and basic-level category is well represented by at least one unit in the LPA layer. (d) dModified DPAN without IOR operation after 100 epochs.

The next stage of training occurs after learning in the 'dog' and 'cat' units stabilizes. At this point, whenever a cat or dog is presented, DPAN executes an IOR operation and chooses another unit to represent the 'unusual' features of the cat or dog that has just been classified. The results of this stage of training are illustrated in Figure 3b. At this point, there are CPCA units that represent subtypes of dogs and subtypes of cats. For example, unit 23 still represents the 'dog' category while unit 17 represents the 'corgi' subcategory.

Finally, after each subordinate-level unit has been exposed to enough token objects, DPAN reaches its last stage of training. During this stage, when the network is exposed to a token object it will simply activate the corresponding subordinate-level localist unit. Note as well that the basic-level units remain intact. If a new subordinate-level category is presented to the network at this point, i.e. one which has no localist representation, the basic-level unit will still activate.

In order to illustrate the importance of the IOR operation, an experiment was carried out wherein IOR was disabled during execution. The results of this experiment are presented in Figure 3d. As anticipated, after 100 epochs the network is still only able to learn the basic-level categories. Without the inhibition of return operation, the network is unable to learn the subtle differences that define the subordinate-level categories.

A Possible Account of the Difference Between Referential and Predicative Nominals

While describing the execution of DPAN, we separated it into three distinct stages. As discussed previously, these three stages of training model a human acquiring expertise in a given category. In the first stage, the network corresponds to referential uses of basic-level categories: "dog" and "cat." Stage 2 corresponds to predicative sentences, featuring subordinate-level categories as predicates: "The dog is a pug." We also posit that the use of the word "is" represents the inhibition operation that allows DPAN to learn these fine grained categories. That is, when describing the token object, a person first activates their internal representation of the object's basic level category ("The dog..."), then inhibits that ("..is a...") to focus on the subtle correlations of the subordinate-level category ("...pug."). Stage 3 corresponds to referential uses of subordinate categories. The agent no longer actively inhibits the basic-level category and instead initially activates its internal representation of the token object's subordinate-level category ("The pug.").

Conclusions and Future Work

Members of basic-level categories contain many highly-correlated features. After enough experience we can inhibit these more obvious connections, allowing us to hone in on more subtle correlations and to create finer-grained subordinate-level categories. The dominant property assembly network is able to learn basic-level and subordinate-level categories in the same manner. DPAN learns the strong correlations that exist in the data to create a representations of basic-level categories. Once these have been encoded, DPAN then uses its inhibition of return operation to learn the dataset's weaker correlations, generating new representations of subordinate-level categories. After enough experience, the subordinate-level categories win out over their basic-level counterparts, mimicking human categorization.

The next step for DPAN is to integrate it into a computational vision system, allowing it to train on real-world image data. DPAN would also benefit from an additional supervised network to assign names to its category representations. DPAN could potentially be used in a large-scale image recognition system to create categories and subcategories of any number of objects.

References

Becker, S., & Le Cun, Y. (1988). Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school* (pp. 29–37).

Bengio, Y., Simard, P., & Frasconi, P. (1994, January). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–66. doi: 10.1109/72.279181

Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological Review*, *94*(2), 115.

Brady, T., Konkle, T., & Alvarez, G. (2011). A review of visual memory capacity: Beyond individual items and toward structured representations. *Journal of Vision*, *11*(5), 1–34.

Chai, Y., Lempitsky, V., & Zisserman, A. (2013, December). Symbiotic Segmentation and Part Localization for Fine-Grained Categorization. In *Iccv 2013* (pp. 321–328). IEEE. doi: 10.1109/ICCV.2013.47

DiCarlo, J. J., Zoccolan, D., & Rust, N. C. (2012, February). How does the brain solve visual object recognition? *Neuron*, *73*(3), 415–34. doi: 10.1016/j.neuron.2012.01.010

Farrell, R., Oza, O., Morariu, V. I., Darrell, T., & Davis, L. S. (2011, November). Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *Iccv 2011* (pp. 161–168). IEEE. doi: 10.1109/ICCV.2011.6126238

Galleguillos, C., Rabinovich, A., & Belongie, S. (2008, June). Object categorization using co-occurrence, location and appearance. In *Cvpr 2008* (pp. 1–8). IEEE. doi: 10.1109/CVPR.2008.4587799

Liu, J., Golinkoff, R. M., & Sak, K. (2001, January). One cow does not an animal make: Young children can extend novel words at the superordinate level. *Child Development*, *72*(6), 1674–94.

Mervis, C. B., & Rosch, E. (1981, January). Categorization of Natural Objects. *Annual Review of Psychology*, *32*(1), 89–115. doi: 10.1146/annurev.ps.32.020181.000513

O'Reilly, R. C., & Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. MIT Press.

Partee, B. (1987). Noun phrase interpretation and type-shifting principles. *Studies in discourse representation theory and the theory of generalized quantifiers*, *8*, 115–143.

Posner, M. I., Rafal, R. D., Choate, L. S., & Vaughan, J. (2007, August). Inhibition of return: Neural basis and function. *Cognitive Neuropsychology*, *2*(3), 211–228. doi: 10.1080/02643298508252866

Quinn, P. C., Slater, A. M., Brown, E., & Hayes, R. A. (2001, June). Developmental change in form categorization in early infancy. *British Journal of Developmental Psychology*, *19*(2), 207–218. doi: 10.1348/026151001166038

Rakison, D. H., & Yermolayeva, Y. (2010, November). Infant categorization. *Wiley Interdisciplinary Reviews: Cognitive Science*, *1*(6), 894–905. doi: 10.1002/wcs.81

Renals, S., Morgan, N., Cohen, M., & Franco, H. (1992). Connectionist probability estimation in the DECIPHER speech recognition system. In *Icassp-92* (Vol. 1, pp. 601–604 vol.1). IEEE. doi: 10.1109/ICASSP.1992.225837

Riesenhuber, M., & Poggio, T. (2000, November). Models of object recognition. *Nature neuroscience*, *3 Suppl*, 1199–204. doi: 10.1038/81479

Rosch, E. (1999). Principles of categorization. *Concepts: Core Readings*, 189–206.

Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M., & Boyes-Braem, P. (1976, July). Basic objects in natural categories. *Cognitive Psychology*, *8*(3), 382–439. doi: 10.1016/0010-0285(76)90013-X

Schiffmann, W. H., & Geffers, H. W. (1993, January). Adaptive control of dynamic systems by back propagation networks. *Neural Networks*, *6*(4), 517–524. doi: 10.1016/S0893-6080(05)80055-3

Senior, A., Heigold, G., Ranzato, M., & Yang, K. (2013). An empirical study of learning rates in deep neural networks for speech recognition. In *Icassp 2013* (pp. 6724–6728).

Thimm, G., Moerland, P., & Fiesler, E. (1996, February). The Interchangeability of Learning Rate and Gain in Backpropagation Neural Networks. *Neural Computation*, *8*(2), 451–460. doi: 10.1162/neco.1996.8.2.451

Tyler, L., & Moss, H. (2001, June). Towards a distributed account of conceptual knowledge. *Trends in Cognitive Sciences*, *5*(6), 244–252. doi: 10.1016/S1364-6613(00)01651-X

Winn, J., Criminisi, A., & Minka, T. (2005). Object categorization by learned universal visual dictionary. In *Iccv'05 volume 1* (Vol. 2, pp. 1800–1807 Vol. 2). IEEE. doi: 10.1109/ICCV.2005.171

Yang, S., Bo, L., Wang, J., & Shapiro, L. G. (2012). Unsupervised Template Learning for Fine-Grained Object Recognition. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 3122–3130). Curran Associates, Inc.

Zhang, L., Gao, Y., Xia, Y., Dai, Q., & Li, X. (2015, January). A fine-grained image categorization system by cellet-encoded spatial pyramid modeling. *IEEE Transactions on Industrial Electronics*, *62*(1), 564–571. doi: 10.1109/TIE.2014.2327558