Derived Distance: Beyond a model, towards a theory

9.520 April 23 2008

Jake Bouvrie

work with Steve Smale, Tomaso Poggio, Andrea Caponnetto and Lorenzo Rosasco

Reference:

Smale, S., T. Poggio, A. Caponnetto, and J. Bouvrie. Derived Distance: towards a mathematical theory of visual cortex, *CBCL Paper*, Massachusetts Institute of Technology, Cambridge, MA, November, 2007.

Outline

- Motivation: why should we use hierarchical feature maps or learning architectures? What can *iteration* do for us?
- Learning invariances: a brief introduction with two examples from the literature at the end.
- Derived distance: towards a theory that can *explain why the CBCL model works.*
- Derived distance: preliminary experiments.
- Derived distance: open problems and future directions.

Why Hierarchies?

Notices of the American Mathematical Society (AMS), Vol. 50, No. 5, 537-544, 2003.

The Mathematics of Learning: Dealing with Data

Tomaso Poggio and Steve Smale

How do the learning machines described in the theory compare with brains?

❑ One of the most obvious differences is the ability of people and animals to learn from very few examples.

❑ A comparison with real brains offers another, related, challenge to learning theory. The "learning algorithms" we have described in this paper correspond to one-layer architectures. Are hierarchical architectures with more layers justifiable in terms of learning theory?

❑ Why hierarchies? For instance, the lowest levels of the hierarchy may represent a dictionary of features that can be shared across multiple classification tasks.

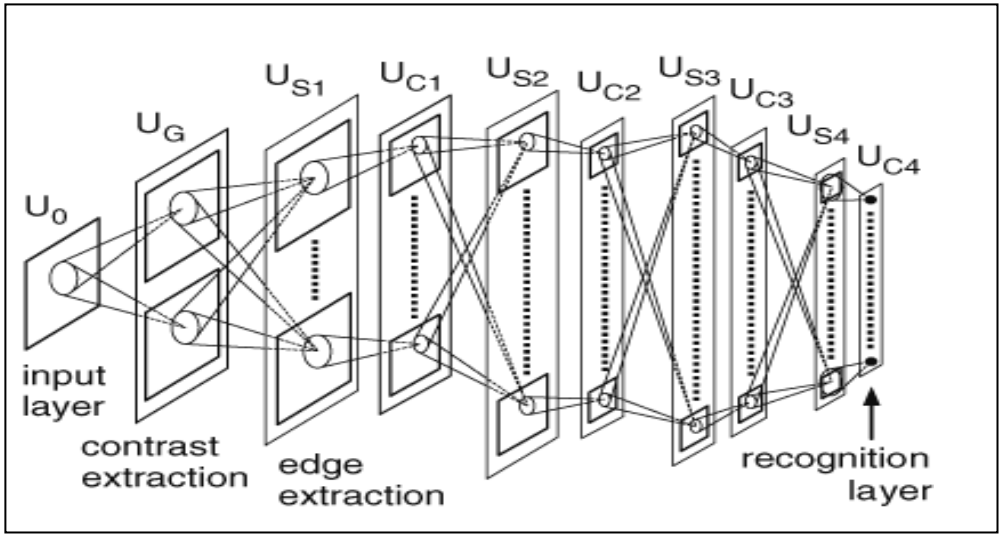
❑ There may also be the more fundamental issue of *sample complexity*. Thus our ability of learning from just a few examples, and its limitations, may be related to the hierarchical architecture of cortex. *In the limit: 1 ex.*

➢ **Hierarchies can be used to Incorporate specific kinds of invariances…(vs. virtual examples).**

Some Engineered Hierarchical Models…

…most with specific invariances built in…

Neocognitron, from Fukushima et al., 1980



CBCL Model

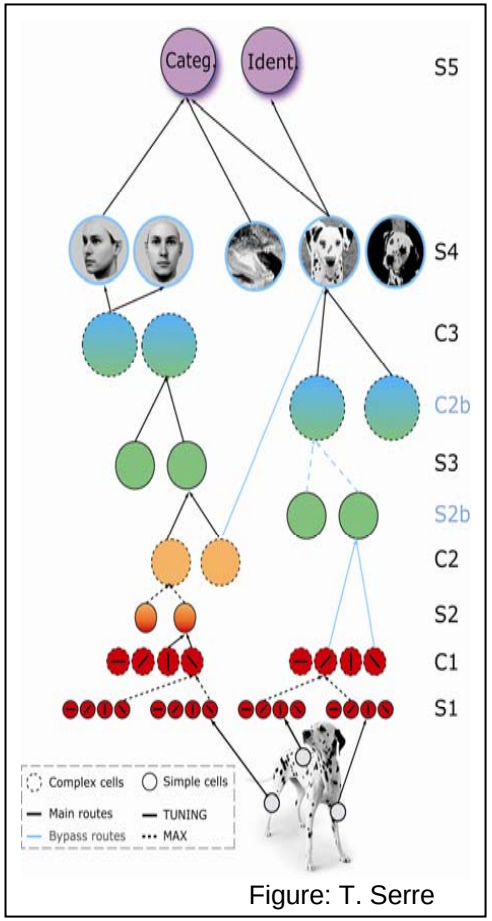
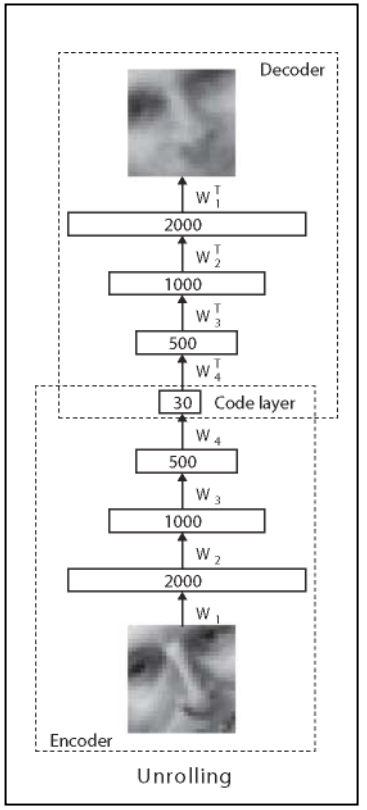


Figure: T. Serre

Hinton's Deep Autoencoder



from: G. Hinton, Science 2007.

Convolutional Neural Networks (LeCun)

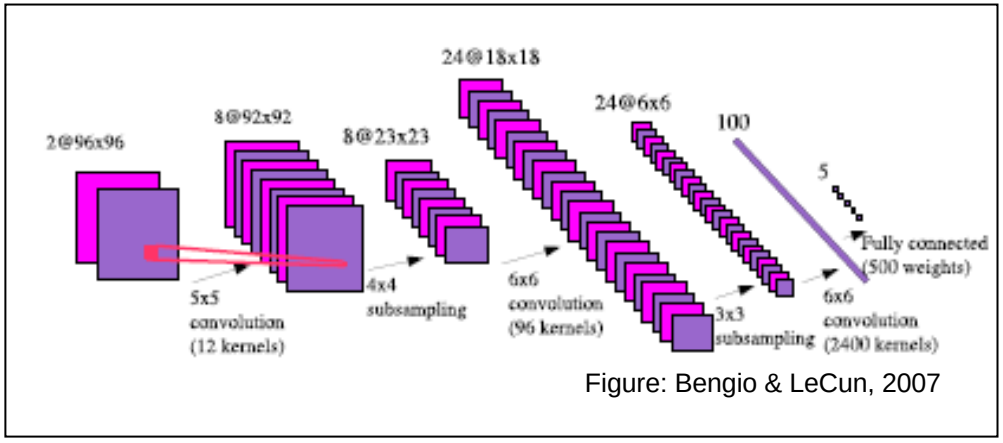
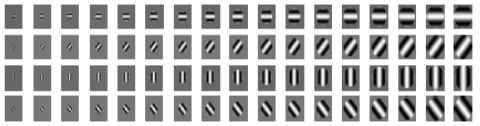


Figure: Bengio & LeCun, 2007



Learning Invariant Representations

- Hierarchies can be used to incorporate particular pre-defined invariances in a straightforward manner, by e.g. pooling, and transformations.
- Combinations of features, combinations of combinations agglomerate into a complex object or scene.
- But what if we don't know how to characterize variation in the data, or even know what kinds of variation we need to capture?
- Nonparametric representations with *random templates:* look for patterns that we've seen before, whatever they might be.
- Learning features and invariances automatically from sequential data, in an unsupervised setting. (Maurer, Wiskott, Caponnetto) – more on this later.

Learning Invariant Representations: Derived Distance

- Steve Smale has proposed a simple yet powerful framework for constructing invariant representations with a hierarchy of associations.
- Derived distance can be seen as a simplification of the CBCL model that lends itself well to analysis.
- Some outstanding questions to be answered:

  -Does it reduce sample complexity? Poverty of the stimulus implies some additional apparatus.

  - Does it provide a separation of classes that is more useful for classification than just the image pixels?

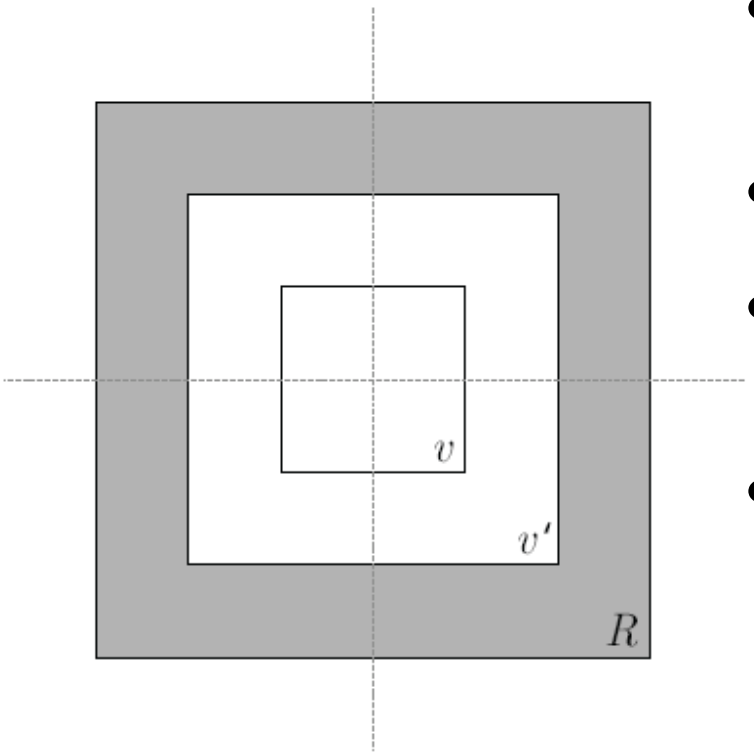
  - If so, what are the optimal parameters?
    - How many layers are needed?
    - Does the distance converge to something interesting in the limit of templates or layers?

Derived Distance: Sketch

- Iterated analysis with arbitrary transforms and nonlinearities in between layers.
- Template dictionaries at each layer encode objects in terms of similarities.
- The set of templates give an empirical approximation to the true distribution of image patches.
- First layer performs operations similar to template matching over the set of allowed transformations.
- At higher layers, we work with representations based on previous layers' templates.
- Final output is the distance (or similarity) between two objects (images, strings,…)
- Summary: Images are represented as a hierarchy of similarities to templates of increasing complexity, modulo scaling and translations.

Derived Distance (2-layer case): Definitions

Smale, S., T. Poggio, A. Caponnetto, and J. Bouvrie. Derived Distance: towards a mathematical theory of visual cortex, *CBCL Paper*, Massachusetts Institute of Technology, Cambridge, MA, November, 2007.



- Consider an image defined on R, $f : R \to [0, 1]$. f belongs to $Im(R)$.
- Domains in $\mathbb{R}^2 : v \subset v' \subset R$
- $Im(v), Im(v')$ are subsets of restrictions of the image $f \in Im(R)$ to v and v'.
- H is a set of transformations $h : v \to v'$ of the form $h = h_\beta h_\alpha$ with $h_\alpha(x) = \alpha x$ and $h_\beta(x) = x + \beta$. Similar definition for $h' \in H'$ with $h' : v' \to R$.

-Note that here, the function h translates the entire *image* over the "receptive field" v. Usually we think of sliding a filter over the image…

Derived Distance (2-layer case): Definitions

A key property:

Axiom: $f \circ h : v \to [0,1]$ is in $Im(v)$ if $f \in Im(v')$ and $h \in H$,
that is *the restriction of an image is an image* and similarly for H'. Thus

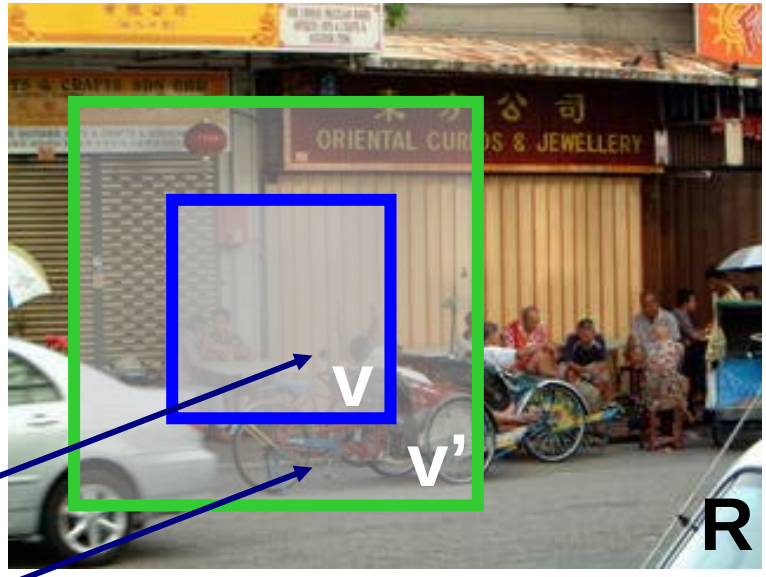
$$f \circ h : v \to [0,1] \in Im(v) \text{ if } f \in Im(v') \text{ and } h \in H,$$
$$f \circ h' : v' \to [0,1] \in Im(v') \text{ if } f \in Im(R) \text{ and } h' \in H'.$$

- A patch of an image is isolated by restricting the image to a transformed domain via composition. $f(h(x))$ is an image itself from $v \to [0,1]$. Depending on how h is chosen (via the parameters α, β) we get a transformed piece of f.

example (translations only):

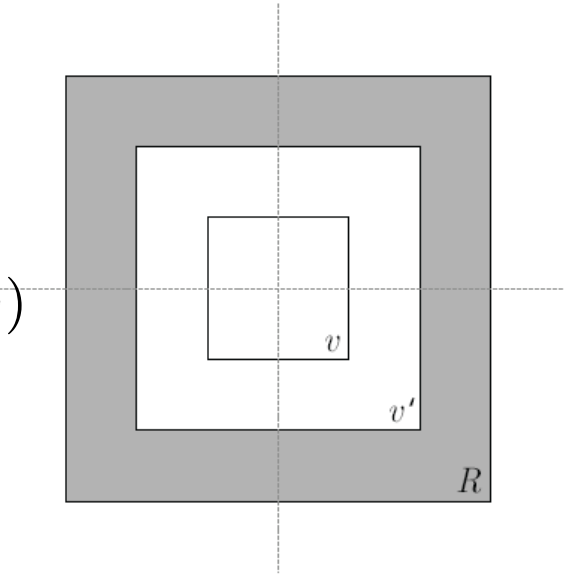
$$f \circ h_{\beta_0} = f \circ h'_{\beta_0} \circ h_{\beta_1}$$

$$f \circ h'_{\beta_0}$$



Derived Distance: Templates

- We will be encoding image patches in terms of "similarities" to some collection of "templates".
- Assume there are finite sets of "templates" $T \subset Im(v)$ and $T' \subset Im(v')$, with probability measures ρ_T and $\rho_{T'}$.
- In practice we could construct template sets by sampling patches randomly from a set of images.
- We'll need one more definition: The set "$\mathbf{R}_+^T$" of functions $T \to \mathbf{R}_+$ accepting a template and producing a positive real number. (This space is also given the structure of an L_p-normed, measurable space with measure ρ_T)



Derived Distance: Neural Similarity

(1) The process starts with some initial distance on $Im(v)$ provided by

$$d'_0(f,g) = d(f,g) = ||f-g||_p, \tag{1}$$

where $||\cdot||_p$ is an appropriate L_p norm $\left(||f||_p = \left(\int_v |f(x)|^p d\mu(x)\right)^{1/p}\right)$, for the space of functions $Im(v)$.

(2) Then we define a first stage *Neural Similarity* as

$$N^1_t(f) = \min_{h \in H} d'_0(f \circ h, t), f \in Im(v')$$

Thus $N^1 : Im(v') \rightarrow \mathbf{R}^T_+$ can be defined[1] by $N^1(f)(t) = N^1_t(f)$.

Derived Distance: Neural Similarity (2)

(2) Then we define a first stage *Neural Similarity* as

pooling step

$$N_t^1(f) = \min_{h \in H} d_0'(f \circ h, t), f \in Im(v')$$

Thus $N^1 : Im(v') \to \mathbf{R}_+^T$ can be defined[1] by $N^1(f)(t) = N_t^1(f)$.

$\longrightarrow N_t^1(f)$ is the best match of the template of size v in the image f. Here f is a sub-image of size v', not the entire image.

$\longrightarrow \mathbf{N}^1(f)$ is an *encoding* of an image f in terms of similarities to templates $t \in T$. The templates are smaller than the image, and are compared to all sub-regions of size v in the image.

Derived Distance: Iterating the Process

(4) We now repeat the process by defining the second stage *Neural Similarity* as

$$N^2_{t'}(f) = \min_{h' \in H'} d'_1(f \circ h', t'), f \in Im(R), t' \in T'. \quad (4)$$

The new derived distance is now on $Im(R)$

$$d'_2(f, g) = ||N^2(f) - N^2(g)||_p. \quad (5)$$

1. $f \circ h'$ selects a patch of size v' from the full image.
2. Then we compute $N^1(f \circ h')$ and $N^1(t')$.

Derived Distance: Example (Top-Down/Recursive)

$$d'_2(f,g) = ||N^2(f) - N^2(g)||_p.$$

$$N^2_{t'}(f) = \min_{h' \in H'} d'_1(f \circ h', t'), f \in Im(R)$$

$$d'_1(f,g) = ||N^1(f) - N^1(g)||_p$$

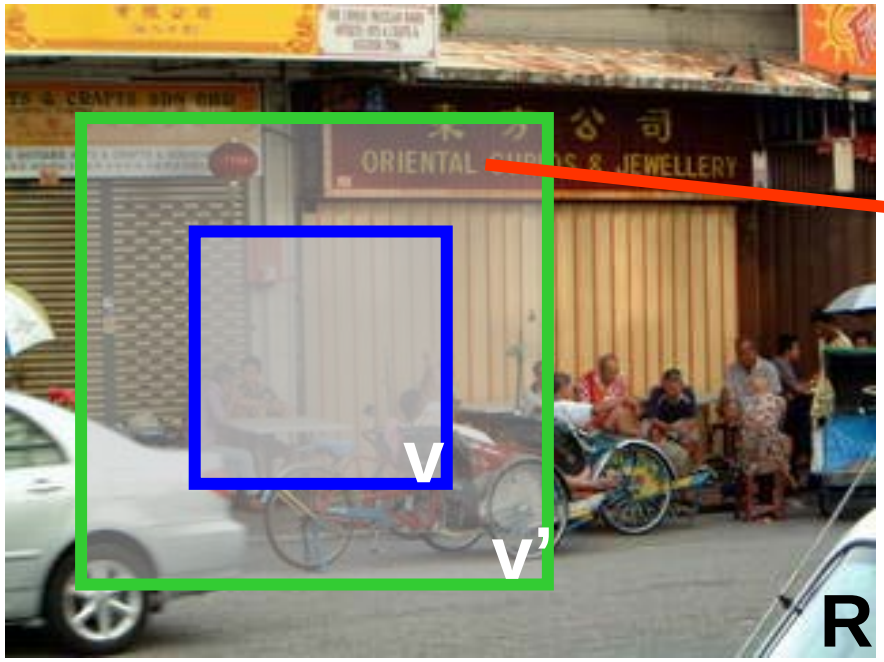
$$N^1_t(f) = \min_{h \in H} d'_0(f \circ h, t), f \in Im(v')$$

$$d'_0(f,g) = d(f,g) = ||f - g||_p$$

Alternating "pooling" and "filtering" steps.

1. Separately find the top-level encodings for f and g, as encodings of encodings.
2. $f \circ h'$ selects a patch of size v' from the full image.
3. Then compute $N^1(f \circ h')$ and $N^1(t')$.
4. To compute $N^1(f \circ h')$, the encoding of the patch $f \circ h'$, we need to consider patches of size v within each patch of size v', $f \circ h' \circ h$, and compare those patches to templates of size v'.

Derived Distance: Example (2)



Every patch p' of size v' is represented as a vector of minimum distances…

$$N^1(f \circ h') = \begin{bmatrix} \min_p d_0(t_1, p) \\ \min_p d_0(t_2, p) \\ \vdots \\ \min_p d_0(t_T, p) \end{bmatrix}$$

templates of size v

patch within p' of size v

Step (2)

An image is first encoded by the hierarchy:

(1) For every patch of size v' in the image (R), we find its encoding by saving the minimum distances between sub-regions of size v in v' and the templates t of size v in the first layer's dictionary.

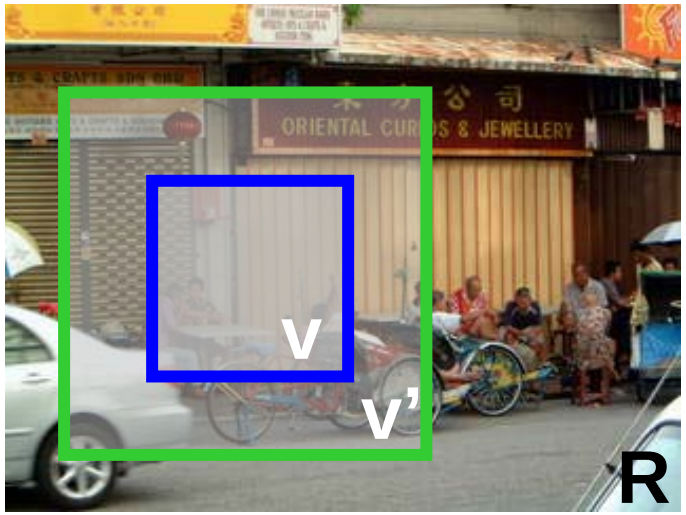
(2) We find similar encodings for the *templates* of size v' in the second layer's dictionary.

(3) We then find the image's final encoding by saving the smallest distances between each template t' of size v' and the set of image patches v', where the distances are taken between the respective *encodings* of templates and patches.

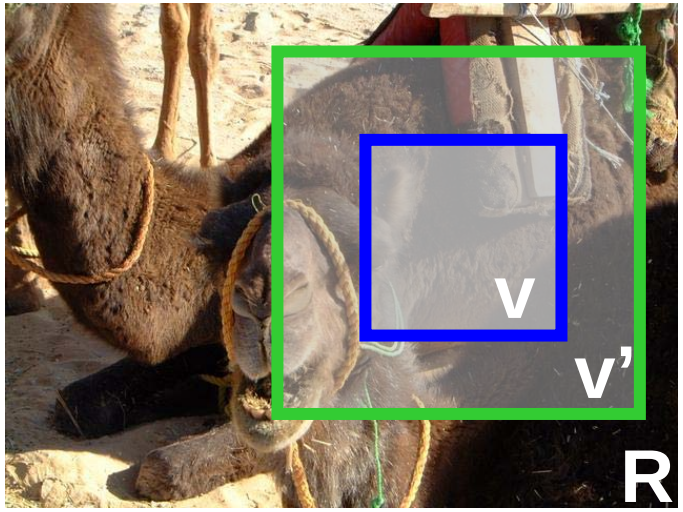
Derived Distance: Example (3)

The image is represented as a vector of minimum distances between template encodings and encodings of patches of size v'

f



g



Step (4)

$$N^2(f) = \begin{bmatrix} \min_{p'} d_1(t_1', p') \\ \min_{p'} d_1(t_2', p') \\ \vdots \\ \min_{p'} d_1(t_T', p') \end{bmatrix}$$

templates of size v' patches in f,g of size v'

$$N^2(g) = \begin{bmatrix} \min_{p'} d_1(t_1', p') \\ \min_{p'} d_1(t_2', p') \\ \vdots \\ \min_{p'} d_1(t_T', p') \end{bmatrix}$$

$$d_2'(\cdot,\cdot)$$

The derived distance between two images is the distance between their encodings.

Derived Distance: Connection to the CBCL Model

CBCL Model

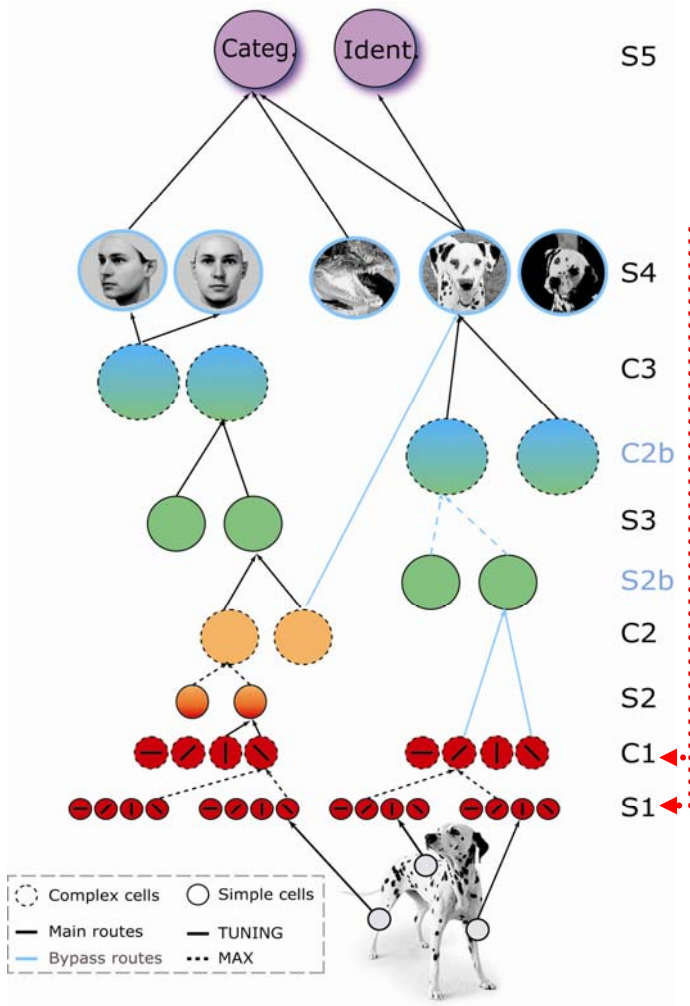


Figure: T. Serre

- First we have $N_{h,t}^{1,S}(f) = d_0(f \circ h, t)$ where $N_{h,t}^{1,S}(f)$ corresponds to the response of an S1 cell with template t with receptive field $h \circ v$.
- The Neural Similarity is now $N_t^{1,C}(f) = \min_{h \in H} d_0(f \circ h, t)$ where $N_t^{1,C} : Im(v') \to \mathbf{R}_+^T$. $N_t^{1,C}(f)$ corresponds to the response of a C1 cell with template t and with receptive field – the region over which the min is taken – corresponding to v'.

(There is still a slight difference between the derived distance formulation and the model involving the pooling…)

Derived Distance with Normalized Kernels

(*recent suggestion from Steve Smale*)

- We can reformulate the model in terms of normalized kernels (similarities) with max as the pooling operation.
- Define

$$\widehat{K}(x,y) = \frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}}.$$

  Proposition: $\widehat{K}(x,x) \equiv 1$ and $\hat{K}$ is p.d. if K is.

- The *derived kernel* at layer n is defined recursively as

$$K_n(f,g) = \operatorname*{Av}_{t \in T_{n-1}} \left\{ \max_{h \in H_{n-1}} \widehat{K}_{n-1}(f \circ h, t) \max_{h \in H_{n-1}} \widehat{K}_{n-1}(g \circ h, t) \right\}.$$

  Proposition: K_n is p.d. if K_{n-1} is.

- The first (base) kernel K_0 can be any p.d. kernel, but thereafter the kernels are normalized dot products (acting on different domains however).

Derived Distance: Remarks

-The probability measure ρ on Im(R) should be given by the real world images for the given vision problem under study. Then the templates $t_i \in T$ can be most conveniently be taken as random draws from ρ_v on Im(v), where ρ_v is the probability measure induced from ρ by restriction.

-The distance can be considered as a hierarchy of associations, with templates of increasing complexity.

- A classifier can be trained on the final similarity vectors, or on combinations of similarity vectors from different layers as is done with the CBCL model.

-The derived distance might also be used to construct a kernel function via $K(f,g)=\exp(-d^2_n(f,g))$, or used in a nearest-neighbor classifier directly.

-The same model can be immediately applied to other domains, such as text and nucleotide sequences.

Image Classification Experiments

-Task: Classify 14x14 pixel handwritten 3's and 8's.
-Compare performance with translations from 0 to 6 pixels in 8 random directions.
- Compare to Euclidean distance between the pixels (straw-man sanity check).
-Simple nearest-neighbor classifier using d_1 or d_2.
- Randomly sampled patches as templates.

(500) 4x4 templates

Figure 2: *Four of the 500 (4x4) templates in T.*

(50) 8x8 templates

Figure 3: *Four of the 50 (8x8) templates in T′.*

Image Classification Experiments (2)

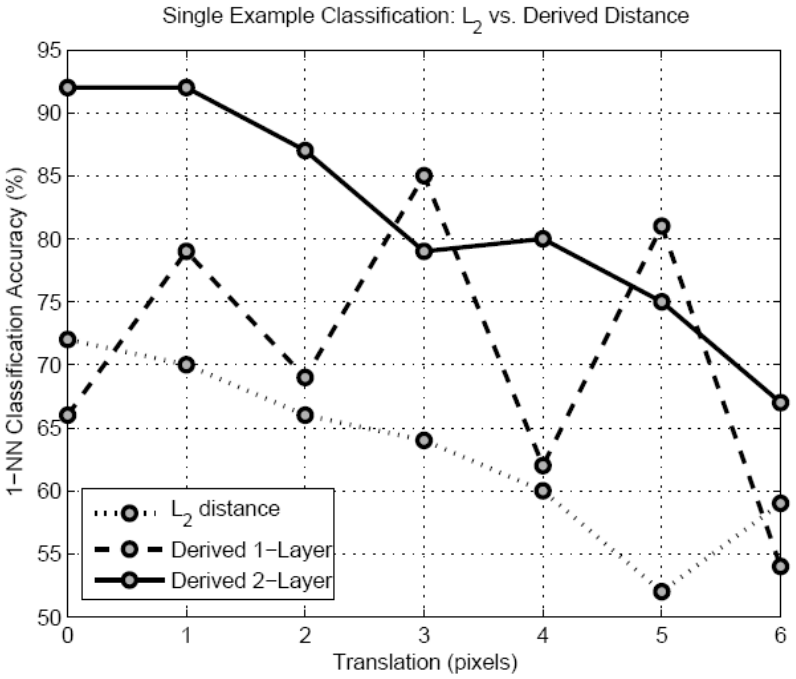
Here, the norms on images and neural similarities take the form:

$$\|f\|_p = \left(\int_v |f|^p d\mu(x) \right)^{1/p} \implies \left(\frac{1}{M} \sum_{i=1}^{M} |f_i|^p \right)^{1/p}$$

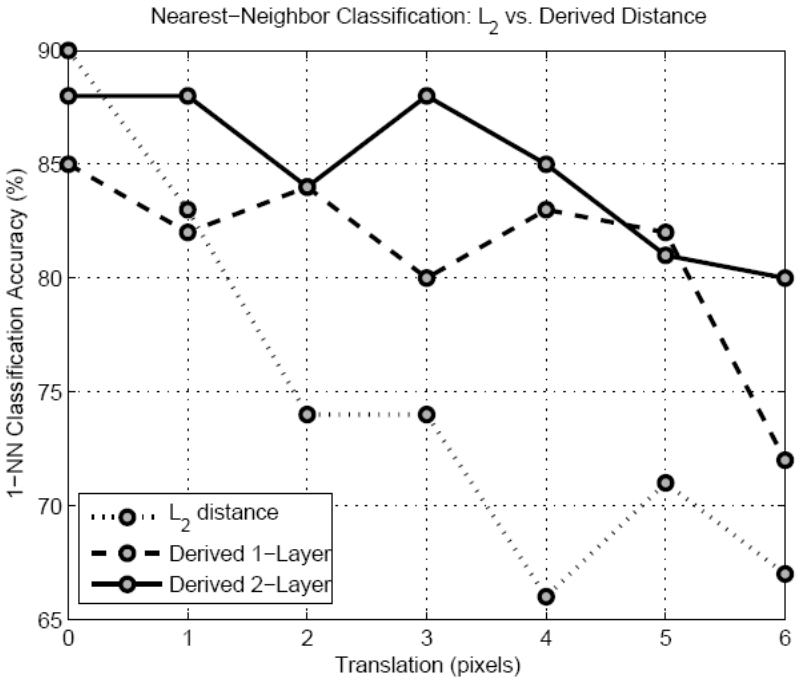
$$\|N(f)\|_p = \left(\int_T |N_t|^p d\rho(t) \right)^{1/p} \implies \left(\frac{1}{|T|} \sum_{i=1}^{|T|} |N_{t_i}|^p \right)^{1/p}$$

where p=2, M is the number of pixels in an image, and we assume a uniform measure throughout. These are just *normalized* Euclidean p-norms.

Image Classification Experiments (3)



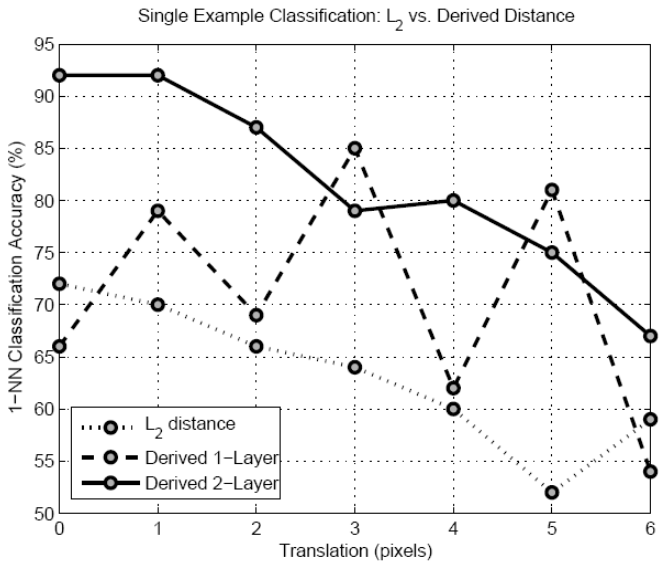
1-NN classification accuracy over 100 test examples with 2 labeled examples, one per class, when using the L_2 distance vs. d_1 and d_2 derived distances. The experiment is repeated with images subjected to translations of 0-6 pixels (x-axis) to test robustness of the distance under translations.

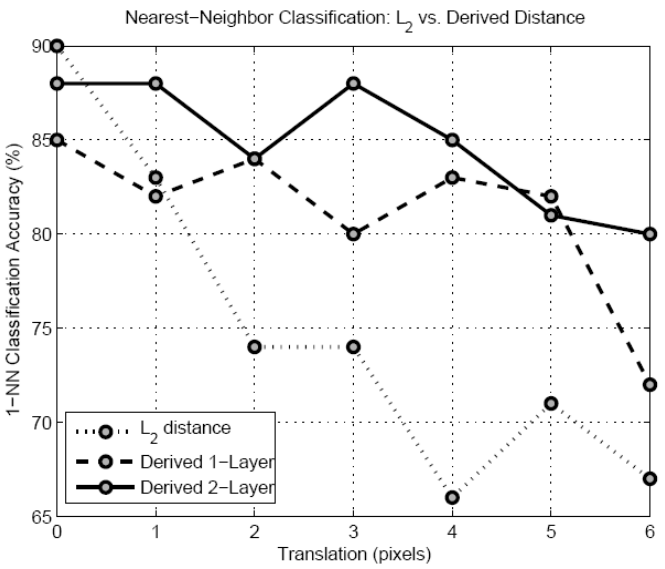


Same, but with 10 labeled examples per class.

…both labeled and unlabeled examples are translated randomly…so L_2 also includes some notion of translation (via virtual examples).

Image Classification Experiments (4)





Observations:

-As might be expected, the derived distances are better able to accommodate image translations than *L2* on the whole, and classification accuracy decays more gracefully in the derived distance cases as we increase the translation size.

-The 2-layer d_2 derived distance is seen to generally outperform the 1-layer d_1 derived distance. Choosing the optimal number of layers is an open problem.

- Even with one canonical example, the derived distance is robust.

Derived Distance: Three-layer String Example

- Consider computing a distance between (DNA) strings with alphabet {**A,T,G,C**}.

- Computing the derived distance involves considering all substrings of a given length or set of lengths.

-Transformations include only linear (but possibly circular) shifts along the length of the string. Other transformations such as transposition, point mutation, insertion, and deletion would be natural choices.

- We define the base distance between two strings of equal length to be the fraction of entries that do not match:

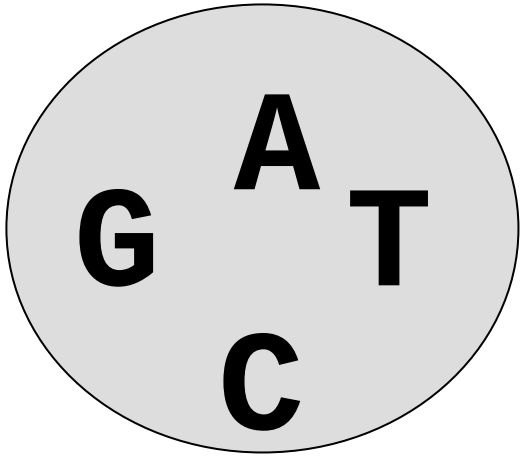
$$d_0(f,g) = \frac{1}{\ell}\sum_{i=1}^{\ell} I_{[f_i \neq g_i]}.$$

Derived Distance: Three-layer String Example (2)

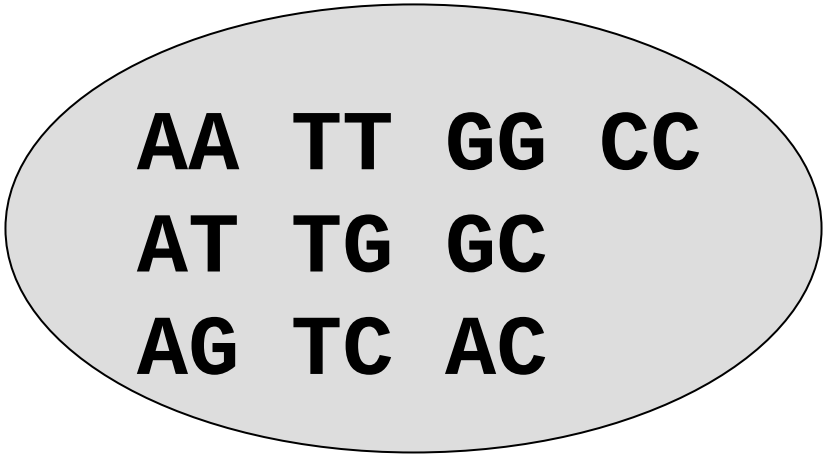
ATCACCGATCGAATAG

v''
v'
v

The derived distance in the case of strings analyzes substrings of decreasing size.
Only circular shifts are allowed in the set of transformations.



Layer 1 Templates: The string alphabet.



Layer 2 Templates:
All pairs represented **uniquely** at the first level.

Layer 3:
All triplets…

Derived Distance: Three-layer String Example (3)

Suppose the string f to be encoded reads `ATTGC`. Then for each template t', $N^2_{t'}(f)$ is the minimum over $\{d_1(t', \texttt{AT}), d_1(t', \texttt{TT}), d_1(t', \texttt{TG}), d_1(t', \texttt{GC})\}$. The lower layer distances d_1 we need to construct $N^2(f)$ are computed analogously:

$$d_1(f, g) = d_0(N^1(f), N^1(g))$$

with

$$N^1_t(f) = \min_{h \in H} d_0(f \circ h, t)$$

where H is now the set of restriction functions isolating each single character in the string f.

Derived Distance: Three-layer String Example (4)

- The neural similarity vector of a 2-char substring is simply a binary vector marking the presence or absence of each character in the alphabet in the sub-string.
- Because the substrings under consideration consist of only two characters and the bottom layer only analyzes (sub)substrings of single characters, we can see that the order of the characters in the two-digit strings is irrelevant. The encoding for `AC` would be the same as that for `CA`.
- With exhaustive template sets, the two-layer derived distance in this case can be summarized as encoding which (order-independent) pairs of characters are present in a string.

Derived Distance: Three-layer String Example (6)

-Task: distinguish random strings from corrupted versions of a single master string.

- "Corrupted" sequences are randomly (circularly) shifted versions of the master sequence.

-Compare to two baseline distances: Number of differing entries ("L0") and one given by the Smith-Waterman alignment score (DP-based alignment algorithm).

1-NN Classification:

	2 Examples	4 Examples	8 Examples
2 Layer DD	78.43	83.32	86.79
3 Layer DD	80.28	85.69	90.19
4 Layer DD	77.53	86.69	85.96
S-W	76.61	81.52	87.08
L_0	49.72	50.24	49.93

Table 1: *Average percent classification accuracies for different derived distance configurations and labeled set sizes. The first three rows give derived distance based accuracies, while the last two rows give the corresponding Smith-Waterman accuracy, and the L_0 baseline accuracy. The accuracies are averaged over random labeled/unlabeled datasets, with 500 trials for the baseline and 2- and 3-layer derived distance cases, and 20 trials for the 4-layer derived distance cases.*

- Derived distance is competitive with an algorithm specifically designed to do alignment.

Derived Distance: Complexity

At first glance, computing the derived distance between two objects might appear to be expensive, but…

L :	number of layers in the hierarchy, with layer 1 the first/bottom layer and layer L the last/top layer with the largest analysis domain.
T_i :	number of templates in the set at layer i.
A_i :	number of operations needed to compute the pooling function (e.g. average, min, max) once at layer i, over regions of size equal to the next layer's templates. The last region, applicable to A_L, is defined to be the entire image.
D_i :	number of operations needed to compute the distance/similarity between a pair of image patches (layer $i = 1$) or a pair of feature vectors (layers $i > 1$).
m_i, n_i :	number of vertical/horizontal pixels in the analysis region at layer i, with $m_1 < \cdots < m_{L+1} = M$, and $n_1 < \cdots < n_{L+1} = N$. Note that the last layer must have an analysis region strictly less than the size of the retina R, in either dimension.

$$\tau = \sum_{i=1}^{L} (M - m_i)(N - n_i)T_i D_i + (M - m_{i+1})(N - n_{i+1})A_i T_i$$

Assuming $m_i = r \cdot m_{i+1}, n_i = r \cdot n_{i+1}, 0 < r < 1,$

$$\tau = NM \sum_{i=1}^{L} (1 - r^{L-i+1})^2 T_i D_i + (1 - r^{L-i})^2 A_i T_i$$

Derived Distance: Complexity

The derived distance can actually be computed in ***linear time*** using a bottom-up algorithm…

- *Linear* in the number of layers L: $\tau = O(L)$ as $L \to \infty$.
- *Polynomial* in a canonical template set size T: $\tau = O(T^2)$ as $T \to \infty$. In practice, it will most likely be the case that $D_i \propto T_{i-1}$, since D_i summarizes the cost incurred when computing the distance between two vectors of length T_{i-1}. If we assume a canonical template set size T so that $T_1 = \cdots = T_L = T$, then we can eliminate the D_i and say that the complexity is $O(T^2)$ as $T \to \infty$.
- *Polynomial* in any single dimension of the input: $\tau = O(P^2)$ as $P \to \infty$, where $P = M$ or $P = N$. A polynomial dependence on the input arises from the fact that $A_i \propto (m_{i+1} - m_i)(n_{i+1} - n_i)$. Each pooling operation will involve looking at as many "elements" (which can be scalars or vectors) of the output from the previous layer as will fit into a region equal in size to the current layer's templates. The previous layer computes an output element for each region of size equal to its templates. Thus A_i is proportional to the number of regions of size equal to templates from that layer which fit into the next layer's region size.
- *Constant* in the ratio of the size of any single dimension of the analysis region at layer $(i+1)$ to the corresponding dimension of the analysis region at layer i: For fixed L, $\tau = O(1)$ as $r \to 0^+$.

A Step Further: Learning Invariant Representations: Slowness

very briefly…

- Wiskott, Maurer, Caponnetto all use some form of the slowness principle to design a concrete objective function. Following (Maurer, ALT, 2006):
- "Significant signals should have a large variance." (this is PCA) $\Rightarrow$ maximize $E\left[\|PX\|^2\right]$. P is an orthogonal projection, and X is a (random) signal.
- "Sensory signals vary more quickly than their significance." $\Rightarrow$ minimize $E\left[\|P\dot{X}\|^2\right]$. In video, pixels change quickly compared to object identities.
- Combining these two ideas, including a tradeoff parameter α, and switching to an emipirical basis, Maurer's objective is

$$\max_P L(P) = \frac{1}{m}\sum_{i=1}^{m}\left(\alpha\|PX_i\|^2 - (1-\alpha)\|P\dot{X}_i\|^2\right)$$

A Step Further: Learning Invariant Representations: RCA

- Connections to distance learning and Relevant Components Analysis (Bar-Hillel et al., JMLR 2005):

  - **(1)** Distances between all pairs of points should be infinite.

  - **(2)** Distances between points within a predefined group should be small.

$$\max_{B \succ 0} log|B| \quad s.t. \quad \frac{1}{N} \sum_{j=1}^{n} \sum_{i=1}^{n_j} ||x_{ji} - m_j||_B^2 = 1$$

$\|\cdot\|_B$ is the Mahalanobis distance with weight matrix B, m_j are group means and x_{ji} are data points.

The solution is a projection matrix $B \propto C^{-1}$ where

$$\hat{C} = \frac{1}{N} \sum_{j=1}^{n} \sum_{i=1}^{n_j} (x_{ji} - m_j)(x_{ji} - m_j)^t$$

This can be thought of as scaling-down variability within the data by assigning lower weight to directions with high variability (due to within-class changes).

Summary

-We looked at learning invariant representations using derived distance and, briefly, using the slowness principle for the case of sequences.

-Derived distance encodes an object in terms of a hierarchy of associations, which can be thought of as a repeatedly applying feature maps to the input data.

-Early experiments show promising results….but many interesting questions remain:

- -What are the optimal parameters for a given task?
- -Can hierarchies reduce computational and/or sample complexity?

-Derived distance can serve as a tractable framework within which one might try to answer these and other questions.