UNSUPERVISED LEARNING OF HIGH-LEVEL INVARIANT VISUAL REPRESENTATIONS THROUGH TEMPORAL COHERENCE

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial satisfaction of the requirements for the degree of

Master of Arts

in

Cognitive Science

by

Ahmed Emin Orhan

Boğaziçi University

2008

Unsupervised Learning of High-Level Invariant Visual Representations through Temporal Coherence

The thesis of Ahmed Emin Orhan

has been approved by:

Prof. Dr. Ethem Alpaydın (Thesis advisor):	____________________
Prof. Dr. Reşit Canbeyli:	____________________
Prof. Dr. Fikret Gürgen:	____________________
Assoc. Prof. Dr. Güven Güzeldere:	____________________
Dr. Ayşecan Boduroğlu:	____________________

June 2008

Thesis Abstract

Ahmed Emin Orhan, “Unsupervised Learning of High-Level Invariant Visual Representations through Temporal Coherence”

Temporal coherence principle is the idea of neglecting rapidly changing components of a temporal signal while keeping to the slowly varying ones, in order to extract useful invariances from the signal. We note that most of the applications of temporal coherence principle to visual stimuli aim at modeling invariances in early vision (mostly deriving invariance properties of complex cells in primary visual cortex). Temporal coherence implementing networks that can accomplish the more challenging task of modelling invariances in higher vision and perform reasonably well on real-world object data-sets requiring some such complex invariant recognition capability are scarcely found. In this work, we try to address this issue by investigating whether a specific variant of the idea of temporal coherence, i.e. slow feature analysis (SFA), can be used to build high-level visual representations that might be useful for invariant object recognition tasks. To date, we know of no network implementation of SFA that is put to challenge on a real-world data-set, rather than on some toy sets of simple, artificial stimuli. To this end, we use single SFA implementing nodes and very generic feed-forward network architectures to see whether SFA itself is capable of modeling high-level invariances in realistic object datasets. We test our models on two datasets that require some such capability for good recognition performance: firstly, on a dataset of letters undergoing translation, planar rotation and scale changes, and secondly on the COIL-20 dataset to see whether SFA can successfully learn view-point invariance. Our results suggest that SFA can yield satisfactory results on these datasets especially when used as a pre-processing step for even very simple supervised classification algorithms. The major limitations for the application of SFA to realistic object databases have been the requirement of large training sets for successful learning and the tendency to quickly overfit the training data as the SFA models become slightly more complex (especially for SFA-3 and SFA-4).

Tez Özeti

Ahmed Emin Orhan, “Zamansal Uyumluluk Yoluyla Yüksek-Düzey Değişmez Görsel Temsillerin Denetim Olmaksızın Öğrenilmesi”

Zamansal uyumluluk ilkesi, zamana bağlı bir sinyalden faydalı değişmezlikler öğrenmek amacıyla, sinyalin hızlı değişen bileşenlerini bir kenara bırakıp, yavaş değişen bileşenlerine ayırmayı ifade eder. Zamansal uyumluluk ilkesinin görsel uyaranlara uygulanması şimdiye dek daha çok erken görme aşamalarındaki degişmezlikleri, özellikle de birincil görsel korteksteki karmaşık hücrelerin değişmezlik özelliklerini, modellemeyi amaçlamıştır. Daha zor bir görev olan yüksek görme aşamalarındaki değişmezlikleri modellemeyi başarabilen, ve bu tür bir beceriyi şart koşan gerçekçi nesne veritabanlarında yüksek performansla nesne tanıyabilen, zamansal uyumluluk ilkesi tabanlı yapay ağlar literatürde nadir bulunmaktadır. Bu çalışma, zamansal uyumluluk fikrinin belli bir türü olan yavaş öznitelik analizinin değişmez nesne tanıma uygulamarında faydalı olabilecek yüksek aşama görsel temsiller olusturulmasında kullanılıp kullanılamayacağını araştırarak, önceki cümlede bahsi geçen eksikliğin giderilmesine katkıda bulunmayı amaçlamaktadır. Bugüne kadar bildiğimiz kadarıyla literatürde yavaş öznitelik analizinin basit, yapay uyaranların aksine gerçekçi nesne veritabanlarına uygulanması gerçekleştirilmemiştir. Bu amaçla, bu calışmada yavaş öznitelik analizinin kendisinin gerçekçi nesne veritabanlarındaki yüksek derece degişmezlikleri modellemeye uygun olup olmadığını araştırabilmek için herbiri ‘yavaş öznitelik analizi’ yapan ünitelerden oluşan basit ileri-beslemeli ağ mimarileri kullanılmıştır. Bu modeller iki çeşit veritabanı üzerinde test edilmiştir: birincisi, düzlemsel pozisyon, düzlemsel rotasyon ve ölçek değişiklikleri uygulanan harflerden oluşan veritabanları; ikincisi, bakış açısı değişmezliği öğreniminin test edilmesi için COIL-20 nesne veritabanı. Bu testlerden elde edilen sonuçlar, yavaş öznitelik analizinin çok basit denetimli sınıflandırma algoritmaları için bile bir ön-işleme adımı olarak kullanıldığında oldukça tatmin edici sınıflandırma performansları elde edilebileceğini göstermektedir. Yavaş öznitelik analizinin gerçekçi veritabanlarına rahatlıkla uygulanabilmesinin önündeki başlıca engellerin başarılı öğrenme için çok büyük öğrenme veri-kümeleri gerektirmesi, ve kullanılan modeller karmaşıklaştıkça (özellikle SFA-3 ve SFA-4 modelleri için) çok çabuk öğrenme veri-kümesine aşırı-uyum eğilimi göstermesi olduğu gözlenmiştir.

CONTENTS

FIGURES

CHAPTER 1

INTRODUCTION

Since Horace Barlow's seminal work suggesting that the chief aim of sensory representations in biological organisms is to reduce the redundancy inherent in a lower level representation so as to make the code more efficient (Barlow, 1961), a lot of progress has been made in sensory coding research. From an evolutionary perspective, and following Barlow, it is reasonable to assume that over the course of evolution and individual development, visual cortex has adapted itself to the statistics of the natural environment so as to make efficient use of its valuable resources and to extract 'useful' information out of the input signal (Olshausen & Field, 1997). This might presumably be achieved by devising efficient encoding-decoding schemes for extracting 'useful' information from the sensory input (Olshausen & Field, 1997). Various studies suggested different ideas as to what exactly an 'efficient' and 'useful' encoding-decoding scheme might amount to or involve: predictive coding (Deco & Schurmann, 2001), sparseness of the representations (Olshausen & Field, 1997), maximization of mutual information (Bell & Sejnowski, 1997), extraction of the statistically independent components (Hoyer & Hyvarinen, 2000), or temporal slowness (Berkes & Wiskott, 2005).

Some encouraging results have been obtained within these frameworks, especially in the visual domain. Quasi-analytic derivation of response properties of V1 cells from these first principles provided deep insight into the behavior of these cells from a computational point of view. It is now well-known for instance that sparse coding of natural images under a linear generative image model, or similarly extracting the independent components of natural images, can reproduce some important response properties of simple cells in the primary visual cortex. More recently, it has been shown that variants of the temporal coherence principle (i.e. slow feature analysis) and independent component analysis, applied to natural image sequences, yield representations whose behavior closely resemble those of the complex cells in the primary visual cortex. These include some important invariance properties in complex cell responses like spatial phase invariance (Berkes & Wiskott, 2005).

One of the shortcomings of these approaches, as we see it, has been the scarcity of sound statistical models extending these insights to higher level representations. It is not known for instance to what extent, or in what ways, the principle of maximization of the sparseness of the representations, or maximization of the temporal coherence of responses can be optimally exploited to develop high-level representations or features that could potentially account for the response properties of cells in the higher visual cortices beyond the primary visual cortex, say, in the infero-temporal cortex. This would, for instance, amount to developing sound statistical models mapping Gabors to object representations.

In this work, we investigate whether a specific variant of the idea of temporal coherence, i.e. slow feature analysis (SFA), can be used to build high-level visual representations that might be useful for invariant object recognition tasks. We use single SFA implementing nodes and very generic feed-forward network architectures to see whether SFA itself is capable of modeling high-level invariances in realistic object datasets. We test our model on two different object datasets that require some such capability for good recognition performance.

Invariant Recognition in Human Vision

A fairly large body of evidence suggests that the recognition of visual objects that are categorizable in some sense is robustly insensitive to various sorts of transformations (within certain ranges) in the sense that there is no statistically significant difference between the recognition of a stimulus and some transformed version of it with respect to accuracy and reaction time measures (Wiskott, 2003).

For instance, Biederman & Cooper (1991, 1992) found that recognition of line drawings of ordinary objects was not affected when the stimulus was shifted by as far as 4.8^o of the visual field, or when its diameter is increased in size from 3.5^o to 6.2^o or decreased from 6.2^o to 3.5^o .

Electro-physiological evidence also supports some of these findings on the invariance properties of object recognition. Tovee et al. (1994) found that a large portion of the neurons in the infero-temporal cortex and superior temporal sulcus of macaque monkeys respond shift-invariantly to presented face stimuli, in terms of their firing rates.

Of course, there are limits to the transformation invariant recognition capability of the visual system, at least in three respects: First, the type of stimuli matters. The visual system does not respond invariantly to the 'admissible' transformations of all sorts of stimuli. Dill & Fahle (1998) found that performance on the recognition of random dot patterns degrades significantly when the stimulus is shifted, suggesting that the stimuli should be categorizable (or 'meaningful' in some sense) for the visual system to respond invariantly to the translations thereof. But it is not known what the exact criteria are that define the class of invariantly recognizable stimuli. It might well be the case that, for instance, extensive training with random dot patterns makes them invariantly recognizable.

Second, even for invariantly recognizable stimuli and for admissible transformations (i.e. transformations under which the visual system responds invariantly), invariance holds within a limited range. So, for instance, in the Biederman & Cooper (1992) experiment mentioned above, shifts of stimuli beyond 4.8^o begin to cause degradations in the recognition performance. Similarly, for human subjects, inverted frontal face images are notoriously more difficult to process and recognize than upright frontal face images, whereas presumably, recognizing frontal face images slightly rotated from the upright position would be nearly as easy as recognizing upright faces.

Thirdly, not all transformations are admissible, i.e. the visual system is capable of responding invariantly only to some transformations of visual objects. Again, the exact criteria defining this class of admissible transformations are not clear.

A good theory of invariant object recognition should be able to explain these limitations in a principled way.

CHAPTER 2

PREVIOUS WORKS

Various neural (network) models for invariant object recognition have been proposed in the literature and these models can be broadly divided into two categories (Wiskott & Sejnowski, 2002): a first group of models more or less build invariances into the architecture of the model. The Neocognitron (Fukushima et al., 1983) and weight-sharing networks are of this type. Here, much of the burden is on the architecture and thus it becomes possible to use very general learning algorithms (e.g. back-propagation) that do not depend on the structure of the input (i.e. the statistics of the natural visual environment in our case) in any way.

The second group of models take just the reverse route. They assume a generic architecture (e.g. layers of neurons with all-to-all or dense connections) and take advantage of the special structure of the input to learn invariances. It is clear that in this case the burden is on the learning algorithm which has to devise special strategies to be able to exploit the structure of the input. The basic idea here is to neglect rapidly changing components of the stimuli while keeping to the slowly varying ones, which is quite reasonable and intuitive if one wants to learn invariances (Einhauser et al., 2005). It was Foldiak's (1991) influential paper which first used this idea to demonstrate how translation (phase)-invariant representations of simple oriented bars can be learned from temporally structured visual experience alone, where the relevant temporal structure is the slower dynamics of local orientation compared to the fast dynamics of local position (or phase).

Now, we believe that, for a couple of reasons, the first approach mentioned above (the architecture driven generation of invariances) is not the right way to go about if we want to have a good model of invariant object recognition in biological systems. First, these models usually assume biologically implausible architectures, like in weight-sharing networks. Second, and perhaps more important, in building these models, the desired invariances should be specified in advance by the designer (Wiskott & Sejnowski, 2002). This restricts the applicability of these models, because we do not know the full range of invariances. In any case, it is better to be able to extract whatever invariances it is theoretically possible to extract, given the statistics of natural visual stimuli, without restricting ourselves to a few pre-defined ones.

On the other hand, again for a number of reasons, the second approach (that of input driven generation of invariances) seems to us to be a promising one. Firstly, the idea of temporal coherence is a general, powerful and intuitive one, potentially capable of extracting any possible invariance and assuming little dependence on a specific form of architecture. Secondly, note that some of the limits on invariant recognition that are touched upon in the first chapter can be quite naturally explained within this framework. It all depends on the temporal statistics of the natural visual environment. Take, for instance, Foldiak's explanation of the generation of phase-invariance. This emerges because it just happens that in our environment local orientation changes more slowly than local position (or phase), so truncating rapidly changing components (here, position or phase) in accordance with temporal coherence principle causes invariance to that feature. Had this temporal relation between local orientation and local position been somehow the other way around, there wouldn't have been any orientation-selective, phase-invariant representations, there would rather have been phase-sensitive, orientation-invariant ones. So, this goes some way in explaining why we have certain invariances rather than others.

Similarly, stimulus-specificity of invariant recognition might be explained by the fact that different types of stimuli display different temporal statistics. For instance, the fact that random dot patterns are not translation-invariantly recognizable might be explained by their different temporal statistics from that of the common visual objects, i.e. it might be that local position is not the fastest changing feature for random dot patterns, so it doesn't get truncated by temporal coherence requirements etc.

There are a couple of different specific proposals within the temporal coherence framework. Einhauser et al. (2005) propose a feed-forward hierarchical model and update the weights at the highest layer by gradient ascent so as to maximize a certain temporal coherence criterion called 'stability'. The stability objective maximized in Einhauser et al. (2005) is the following:

$$\Psi^{\text{stable}} = \sum_i \psi_i^{stable} = \sum_i -\frac{\left\langle (\frac{d}{dt} A_i(t))^2 \right\rangle}{var_t(A_i(t))}, \tag{1}$$

where $A_i(t)$ represents the activity of the unit i (called an 'object cell', OC) at the top layer at time t. They also put an additional decorrelation term into the objective, to make sure that different object cells acquire different object selectivities. The objective in (1) is thus to minimize the average squared derivative of the object cells, hence to obtain more 'stable' OC responses so that as the object within the visual field gets transformed, the OC response stays more or less the same, thus acquiring invariance to that transformation.

Einhauser et al. (2005) shows that OCs trained with the objective in (1) can be used to achieve a high level of invariance to changes in view-point and a good recognition performance in the COIL-100 object database even using a very simple unsupervised classification schema (basically, using k-means clustering as a classification algorithm).

Stone (1996) and Stone & Bray (1995) use another objective for learning invariances from temporally structured signals. The objective that they use for this purpose is the following:

$$F = \frac{1}{2} log \frac{V}{U} = \frac{1}{2} log \left(\frac{\sum_{t=1}^{T} (y^t - \bar{y}^t)^2}{\sum_{t=1}^{T} (y^t - \tilde{y}^t)^2} \right), \tag{2}$$

where y^t represents the activity of one output unit at time t, $\bar{y}^t$ represents a long-term average of outputs y (an exponentially weighted sum of outputs prior to time t, with a long decay time) and $\tilde{y}^t$ represents a short-term average of outputs y (similarly, an exponentially weighted sum of outputs prior to time t, this time with a shorter decay time). Maximizing this objective amounts to maximizing the long-term variability (or long-term predictability) of the signal y, while simultaneously minimizing its short-term variability (or short-term predictability).

Using simple stimuli, Stone (1996) shows that using the objective in (2), one can extract some useful low level visual information such as stereo disparity or surface depth from a rapidly changing visual stream. Again, the key feature of visual stimuli enabling the extraction of such information is the relatively slow change of these features over time, compared with the much more rapid change of sensor (i.e. retinal) inputs. A learning algorithm maximizing an objective function such as the one in (2) will be able to extract those features by maximizing the short-term variability (or predictability) of the outputs.

Bray & Martinez (2002) develops a kernel-based version of Stone's (1996) algorithm and shows that disparity and translation-invariant complex cell-like features can be learned by applying their algorithm to temporally structured visual sequences. In Martinez & Bray (2003), they also show that the same algorithm can be successfully used for non-linear blind source separation. This kernel-based version of the algorithm has the additional advantage that it can search for the slowly changing components in much more complex and higher-dimensional spaces.

Another proposal along the lines of temporal coherence principle is ‘Slow Feature Analysis (SFA)’ which is particularly interesting because it can learn to extract multiple invariances (scale-invariance, shift-invariance, rotation-invariance, contrast-invariance, luminance-invariance etc.) simultaneously, which seems to be a more realistic scenario under conditions of natural visual stimulation than the sequential learning of invariances by separate, ‘favorable’ sets of visual stimuli that involve only certain transformations. SFA was first introduced in Wiskott and Sejnowski (2002) which also presents a purely feed-forward hierarchical network of SFA-modules as a simple model of how multiple invariances might be learned simultaneously, using one-dimensional, simple, artificial stimuli. They note, though, that performance degrades significantly as the network tries to learn more and more invariances.

It is interesting to note that most of these temporal coherence networks aim at modeling invariances in early vision (mostly deriving invariance properties of complex cells in primary visual cortex). Temporal coherence implementing networks that can accomplish the more challenging task of modelling invariances in higher vision and perform reasonably well on real-world object data-sets requiring some such complex invariant recognition capability are scarcely found. Notable exceptions are the Einhauser et al. (2005) network mentioned above which performs quite well on the COIL-100 object database and Wallis & Rolls (1997) network which implements Foldiak’s trace rule in successive layers iteratively and learns to recognize faces.

In particular, we know of no network implementation of SFA that is put to challenge on a real-world data-set, rather than on some toy sets of simple, artificial stimuli. In this work, we try to address this issue by suggesting, among other things, a straightforward 2-D extension to the 1-D hierarchical feed-forward network presented in Wiskott & Sejnowski (2002).

CHAPTER 3

SLOW FEATURE ANALYSIS (SFA)

Slow feature analysis was introduced in Wiskott & Sejnowski (2002). As in other implementations of the temporal coherence principle, the motivation behind slow feature analysis is the simple observation that meaningful variables of the environment vary on a much slower time scale than the raw sensor inputs. Thus, one might expect to extract useful information about the environment by simply finding a mapping of the rapidly changing sensor signal such that the output signal of the mapping, in contrast to the raw sensor signal, changes as slowly as possible. Thus, similarly to the optimization carried out in Einhauser et al. (2005), SFA basically finds the directions along which the average derivative of a time-varying signal is minimal, and to be able to extract 'interesting' features, it does this not directly in the input space, but in a non-linearly expanded high dimensional space.

SFA: Problem Statement and the Algorithm

More formally, the learning problem in Wiskott & Sejnowski (2002) consists of finding an input-output function $\mathbf{g} : \mathbb{R}^N \to \mathbb{R}^M$ such that for a given input signal $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]^T$ and for each $m \in \{1, \ldots, M\}$:

$$\Delta_m \equiv \left\langle \dot{y}_m^2 \right\rangle = \left\langle \dot{g}_m(\mathbf{x})^2 \right\rangle \textit{ is minimized} \tag{3}$$

subject to the constraints:

$$\langle y_m \rangle = 0 \tag{4}$$

$$\left\langle y_m^2 \right\rangle = 1 \tag{5}$$

$$\forall\, m' < m : \quad \langle y_{m'} y_m \rangle = 0 \tag{6}$$

Here, $\langle\rangle$ always denotes a time average. Constraints (4) and (5) are introduced for avoiding the trivial solutions $y_m(t) =$ constant, while constraint (6) ensures that different functions are learned in each dimension. In general, this is an ill-defined optimization problem, unless we restrict the function space from which $\mathbf{g}$ comes. In Wiskott & Sejnowski (2002), this restriction is done by requiring each g_m to be a weighted sum of K pre-defined non-linear basis functions, $h_k(\mathbf{x})$. Usually K is much bigger than the input and output dimensions, N and M. More formally, this can be expressed as:

$$g_m(\mathbf{x}) = \sum_{k=1}^{K} w_{mk} h_k(\mathbf{x}). \tag{7}$$

Possible choices for the set of non-linear basis functions $h_k(\mathbf{x})$ to be used in this expansion might, for instance, be the set of all monomials of the input of degree one, or of one and two, yielding different variants of the SFA algorithm (to be introduced shortly) called SFA-1 and SFA-2 respectively. Now, the expansion in (7) reduces the optimization problem defined in (3)-(6) to one of finding the weights, $\mathbf{w}_m = [w_{m1}, \ldots, w_{mK}]^T$, that satisfy (3)-(7), which is a linear problem in the expansion space, $\mathbf{h}$.

The algorithm for solving this optimization problem consists of the following steps:

1. Normalization: First, normalize the input signal so that in each dimension, n, the input has zero mean and unit variance:

$$\langle x_n \rangle = 0 \tag{8}$$

and

$$\left\langle x_n^2 \right\rangle = 1 \tag{9}$$

2. Non-linear expansion: Expand the signal into a high-dimensional space, e.g. the space of all first- and second-order monomials of the input, for which case the algorithm is called SFA-2, or quadratic SFA:

$$\tilde{\mathbf{z}}(t) \equiv \mathbf{h}(\mathbf{x}(t)) = [x_1(t), \ldots, x_N(t), x_1(t)x_1(t), x_1(t)x_2(t), \ldots, x_N(t)x_N(t)]^T \tag{10}$$

Other choices for the non-linear expansion space are possible. Besides the quadratic expansion, we will be using third-order (SFA-3) and fourth-order (SFA-4) expansions in the simulations below. Similar to the quadratic expansion, third-order expansion involves expansion into the space of all first, second, and third-order monomials of the input and fourth-order expansion involves expansion into the space of all first, second, third and fourth-order monomials of the input. Note that if the original input signal is N-dimensional, the expanded signal becomes N + N(N+1)/2 dimensional in quadratic SFA (i.e., SFA-2), N + N(N+1)/2 + N(N+1)(N+2)/6 dimensional in SFA-3, and N + N(N+1)/2 + N(N+1)(N+2)/6 + N(N+1)(N+2)(N+3)/24 dimensional in SFA-4.

3. Sphering: Sphere the data in the expanded space (e.g. by using PCA), i.e.

find a linear transformation, $\mathbf{S}$, in the expanded space such that the transformed input, $\mathbf{z}(t) \equiv \mathbf{S}(\tilde{\mathbf{z}}(t) - \langle \tilde{\mathbf{z}} \rangle)$, has the following properties:

$$\langle \mathbf{z} \rangle = \mathbf{0} \tag{11}$$

and

$$\left\langle \mathbf{z}\,\mathbf{z}^T \right\rangle = \mathbf{I} \tag{12}$$

4. PCA: Apply PCA to the matrix $\left\langle \dot{\mathbf{z}}\,\dot{\mathbf{z}}^T \right\rangle$ to solve the following eigenvalue problem:

$$\left\langle \dot{\mathbf{z}}\,\dot{\mathbf{z}}^T \right\rangle \mathbf{w}_m = \lambda_m \mathbf{w}_m \tag{13}$$

with $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_M$. Note that this induces an ordering on the output components (henceforward called slow components), y_m, according to their average rate of change $\langle \dot{y}_m^2 \rangle$, with y_1 being the most slowly changing variable, and y_M the least.

It is also important to note that in some cases, successive applications of the algorithm might be required for properly extracting the slowly varying components of the input signal, as will be the case in some of the simulations below.

Applications and Extensions of SFA

Berkes & Wiskott (2005) shows that applying quadratic SFA to natural image sequences that involve rotation, translation and scale changes yields features that, both qualitatively and quantitatively, display many of the important characteristics of the complex cells in primary visual cortex (V1), such as phase-invariance, direction-selectivity, non-orthogonal inhibition, side- and end-inhibition.

Similarly, Franzius et al. (2007) uses a hierarchical network of SFA implementing nodes together with a sparse coding (or similarly independent component analysis) node at the top layer and shows that the network, when trained on quasi-natural image sequences intended to model what a simulated animal sees while freely moving on a small grid, can learn to generate responses highly resembling the responses of various types of cells in the hippocampal formation such as place cells, head-direction cells and spatial view cells.

These results together suggest that extraction of slow components might be one of the principles or objectives underlying (or guiding) cortical computation at large.

In another application of SFA that is most closely related to what we are doing in this work, Berkes (2006) applies SFA to a recognition task: recognition of handwritten digits from the MNIST database. Since it is essentially the derivative signal, and not the raw input signal, that is relevant to the computation of the slow components, Berkes (2006) first forms a huge dataset (~1,000,000 training instances in total) consisting of the differences of randomly selected image pairs belonging to the same class. After applying SFA to this new dataset, he uses a simple classification strategy to assign classes to instances from the test set. He shows that combined with even such a simple classification strategy, an unsupervised SFA-3 (with 35 input dimensions) yields an error rate of 1.5% on the test set, which is comparable to the performances of more sophisticated supervised learning algorithms, such as an SVM with fifth degree polynomials. The requirement of large datasets for successful training of SFA is one of the major limitations of the algorithm, as we will see in the simulations below.

Sprekeler et al. (2007) provides a concrete link between the abstract SFA algorithm and its biologically plausible implementation within the neural-ware. They show that an experimentally verified and well-known synaptic plasticity rule called 'spike time dependent plasticity' (STDP) might be interpreted as a possible implementation of the slowness principle. This paper also provides a unifying framework for different variations of the temporal coherence principle by showing that a simple modified Hebbian learning rule such as:

$$\dot{w}_i = \gamma \left[f^{in} \circ a_i^{in} \right] \left[f^{out} \circ a^{out} \right] \tag{14}$$

can be used to implement different variants of the temporal coherence principle. Here, a_i^{in} denotes the i-th input, f^{in} is the input filter; a^{out} denotes the output activity, f^{out} is the output filter and $\circ$ denotes the convolution operation. They show that with different choices of f^{in} and f^{out}, one can implement Foldiak's trace rule, Stone's (1996) algorithm, or SFA, using one and the same filtered Hebbian rule in (14).

More theoretical results concern the relationships between SFA and independent component analysis (ICA).

Blaschke et al. (2006) shows that linear slow feature analysis (SFA-1) is equivalent to second-order ICA with one time-delay and Blaschke et al. (2007) shows that temporal slowness might be used as a useful complementary constraint to statistical independence in non-linear blind source separation problems and they also develop an algorithm that combines both of these objectives to extract *both* slowly changing *and* independent components from a non-linearly mixed input signal. Note that in the form presented in the previous section, SFA algorithm extracts components that are slowly changing *and only* de-correlated (not necessarily independent).

CHAPTER 4

MODELS AND DATASETS

Two datasets have been used in the simulations below. The first dataset consists of the first 15 letters of the English alphabet (in Times New Roman font). Letters in the dataset have been transformed in three ways: random translation, random translation plus planar rotation and random translation plus scale changes. These different cases are treated in different subsections in the next chapter. In each case, 6000 instances from each class were generated by applying the transformation (or transformations) to the last generated instance of each class. Two different generalization performances were assessed: generalization to unseen instances and generalization to unseen classes. In the first case, some views of each training letter were retained for testing, and in the second case, all instances of all letters not used during the training phase were included in the test set. Classification performance on the training sets was also assessed. Generalization to unseen instances (and classification performance on the training set as well) was assessed by training multivariate linear classifiers on the outputs of the models, but only the outputs to those instances used for training the model were included in the training set of the linear classifier.

For testing generalization to unseen classes, a different procedure based on k-means clustering was used. This method is essentially borrowed from Einhauser et al. (2005) and will be described in the next chapter in more detail.

The second dataset used in the simulations below is the COIL-20 object database (Nene et al., 1996). We used this database to investigate whether SFA can learn view-point invariance in a complex object database. The large number of images required for training SFA necessitated some pre-processing. We have chosen to use the pre-processing employed in Einhauser et al. (2005), again described in the next chapter. Performance evaluation of the models were done in the same way as in the letters dataset.

The models compared here include two-layered hierarchical networks of SFA nodes, single SFA nodes, PCA nodes, and for the COIL-20 database, pre-processed inputs themselves. Three types of SFA nodes have been tested: SFA-2, SFA-3 and SFA-4, all described in Chapter 3. For the network SFA models, the first layer consisted of 3x3 SFA nodes with overlapping receptive fields of size 8x8 each. The second layer consisted of a single SFA node that received inputs from all of the 9 layer-1 nodes. At each node, dimensionality reduction (if necessary) was made with SFA-1. Thus, in the simulations we used the linear SFA model as a dimensionality reduction method. The term 'input dimensionality' refers to the dimensionality of the space thus reduced by SFA-1. The input dimensionalities of all nodes in a network were kept equal in order to reduce the free parameters of the model. The number of slow components retained at each node was held fixed and equal to 7 for all the simulations below, unless explicitly stated otherwise. Experiments with 'deeper' networks demonstrated that they had poor generalization performance presumably because of the explosion in the number of parameters to be estimated.

For the simulations with COIL-20 database, we did not use the network models, since the low dimensionality of the input after pre-processing made this unnecessary.

All simulations were done using MATLAB (The Mathworks, Natick, NJ) software (R13). Basic MATLAB routines for performing slow feature analysis were obtained from Sfa_tk: Slow feature analysis toolkit for MATLAB provided in Berkes (2003).

CHAPTER 5

SIMULATION RESULTS

Experiments with Letters

We first use simple letter stimuli for training single SFA nodes and networks of SFA nodes. 32x32 images of the first 15 letters (Times New Roman font) of the English alphabet were created using CorelDRAW computer-graphics software (Corel Corp., Ottawa, ON). Each letter was centered in the image and the image consisted of the smallest square containing the letter.

Translation Invariance with Letters

Random Translation - Using Single SFA Nodes versus Networks of SFA Nodes

To obtain stimuli for training SFA for translation invariance, we prepared a 64x64 background image. Letters were placed on a random initial position on the background image and then randomly translated across the image such that at each step the letter (i.e. the central pixel of the image letter) moved randomly to one of the eight neighboring pixels on the background image, if that pixel indeed exists, otherwise the letter moved to one of the existing neighbors. The background pixel intensities were all set to random values (all pixels uniform between [0,1]) using a simple threshold technique. The resulting image was down-sampled to 16x16. 6000 such images were generated for each of the 15 letters (classes). Note that this number is larger than the number of possible positions that the letter might occupy on the background image (which is 1089). However, since it is the difference between consecutive images, i.e. linear approximation to the derivative of the training signal, that is relevant for the training of a SFA node (and not single images themselves), this does not necessarily imply a redundancy in the training set.

We first trained single SFA nodes, and then two-layered hierarchical networks of SFA nodes on this training set. Three types of SFA nodes have been tested: SFA-2, where the expansion space is the space of all monomials of the input up to order 2 (i.e. a quadratic kernel), SFA-3, where the expansion space is the space of all monomials of the input up to order 3, and SFA-4, where the expansion space is the space of all monomials of the input up to order 4. We have also tested the dependence of the classification performance of the algorithm on the number of input dimensions used (i.e. the dimensionality of the reduced space before the application of SFA-2, SFA-3 or SFA-4) and on training set size.

In this case, the first 10 letters were used for training and the remaining 5 letters for validation (test set). So for this case, we did not test the generalization performance of the model to unseen instances of the training set. The classification performance on the training set was assessed by training simple linear classifiers on the output of each model. On the other hand, classification performance on the test set was assessed as suggested by Einhauser et al. (2005). K-means clustering was applied to the output of the model multiple times, with different initial settings. The number of clusters was not learned, but simply given to the clustering algorithm as the actual number of classes in the test set, which is 5 in this case. Classes are then assigned to the learned clusters according to the procedure described in Einhauser et al. (2005): first, a kxk hit-matrix was constructed, whose *(i,j)* entry showed how many items cluster i shared with class j; the maximum entry of this matrix was found and the corresponding class was assigned to the corresponding cluster; after eliminating the row and the column corresponding to that class and the cluster, this procedure was repeated until all classes were assigned to a cluster and vice versa. Classification performance was then assessed according to this classification schema. K-means algorithm was run 5 times and averages over these runs were taken to determine the classification performance on test set. In the figures below, unless otherwise stated, error bars show (one) standard deviation of the classification performance over different runs of the k-means clustering.

As an illustration, we first show the kinds of responses learned by one of our models. Figure 1 shows responses of the first 3 slow components to 8 classes of the training set in a two-layered hierarchical network of SFA-2 nodes. This network consisted of two layers of SFA-2 implementing nodes. As mentioned in Chapter 4, the first layer consisted of 3x3 SFA-2 nodes with overlapping receptive fields of size 8x8 each. The second layer consisted of a single SFA-2 that received inputs from all of the 9 layer-1 nodes. Thus, at each layer-1 node the input received was 64-dimensional. This 64-dimensional input was first reduced to 34 dimensions using linear SFA, i.e. SFA-1, and then SFA-2 was applied. Each SFA-2 node extracted 7 slow components, and fed those to the layer-2 node. Thus, the layer-2 node received a 9 x 7 = 63-dimensional input. This 63-dimensional input was, in turn, reduced to 34 dimensions again using SFA-1, and finally 7 slow components were extracted by the layer-2 node. This network architecture was held fixed in all the simulations below, unless otherwise stated. We only tested the dependence of the classification performance on the reduced input dimensionality, which is 34 in this example.

This network was able to linearly separate all training classes from each other, yielding zero classification error with a linear classifier that used just the first 4

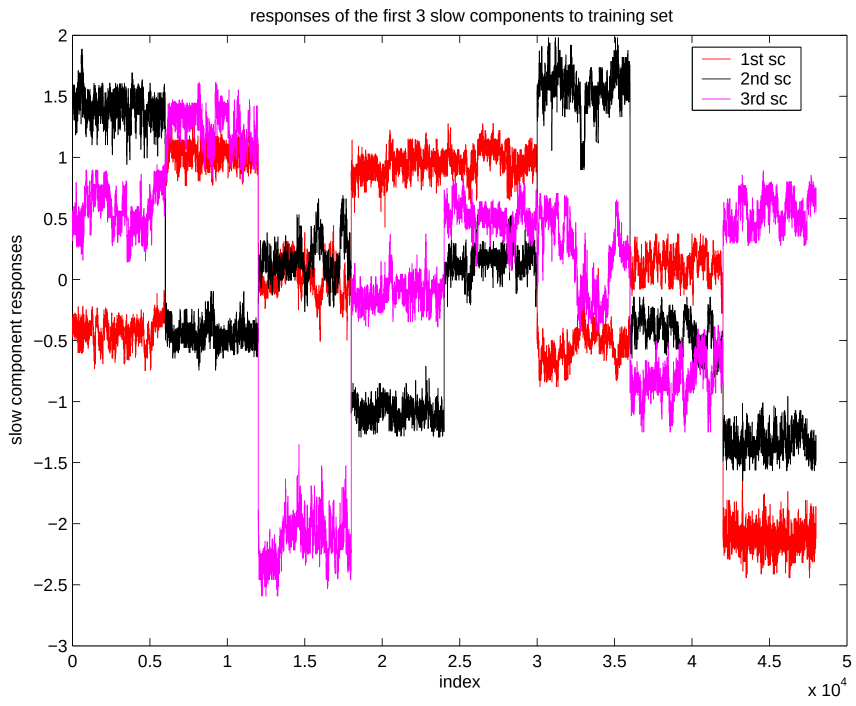


Figure 1: Responses of the first 3 slow components to 8 classes of the training set in a two-layered hierarchical network of SFA-2 nodes.

components extracted at the top layer-2 node. This is illustrated in Figure 2, where we plot the projection of the training set on the first 3 slow components extracted at the top layer-2 node. As can be seen from the figure, all classes except A and F become pair-wise linearly separable. Adding one more component made them linearly separable as well.

Generalization to Unseen Classes - Using Hierarchical Networks of SFA Nodes

We first present the results for hierarchical networks of SFA nodes. Figure 5 shows k-means classification performance of networks with different types of SFA nodes and reduced input dimensionalities under different training set sizes: (a) 3000 instances from each of the 10 classes, (b) 4000 instances, (c) 5000 instances and

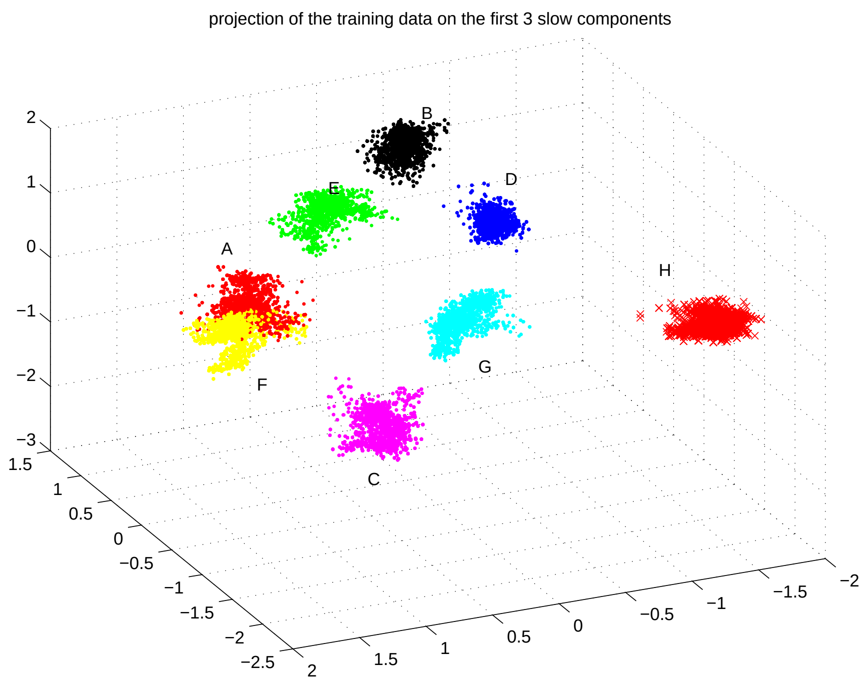


Figure 2: Projection of the training set on the first 3 slow components extracted at the top layer-2 node.

(d) 6000 instances from each class. In each subfigure, x-axis is the reduced input dimensionality, used in all the nodes throughout the network, and y-axis is the k-means correct classification rate on the test set. Note that with 5 classes in the test set, the chance level for the k-means correct classification rate would be 0.2. Again, in each subfigure magenta indicates a network with SFA-2 nodes, black indicates a network with SFA-3 nodes, and red, a network with SFA-4 nodes. The large memory costs associated with explicitly working in a high-dimensional expanded space, especially for the case of networks with SFA-3 and SFA-4 nodes precluded us from carrying out an extensive search of the parameter space for these cases. For instance, an input dimensionality of 9 at a SFA-4 node would mean working in a 714-dimensional space after the expansion. Similarly, an input dimensionality of 14 at a SFA-3 node would mean working in a 679-dimensional space after the expansion. On the other hand, even an input dimensionality of 34 at a SFA-2 node means working in a 629-dimensional space after the non-linear expansion, which is still a smaller space than the two. Working in a higher dimensional space means learning of more parameters, which in turn necessitates using larger training sets to avoid overfitting and poor test set performances. Indeed this is reflected in Figure 5. With smaller training sets, e.g. in (a) and (b), one can notice the sharp decrease in classification performance in networks of SFA-3 nodes (black lines), as input dimensionality is increased from 9 to 14 inputs, whereas classification performance on training set actually increases. Training errors of linear classifiers are shown for hierarchical models and single SFA nodes, in Figures 3 and 4 respectively.

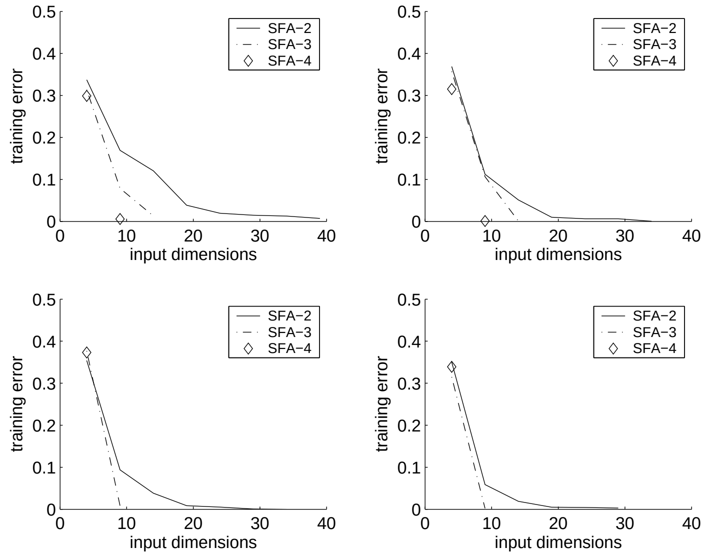


Figure 3: Average training errors of linear classifiers trained on the outputs of hierarchical networks of SFA nodes, with different training set sizes: 3000 (top left), 4000 (top right), 5000 (bottom left), 6000 (bottom right) instances per class.

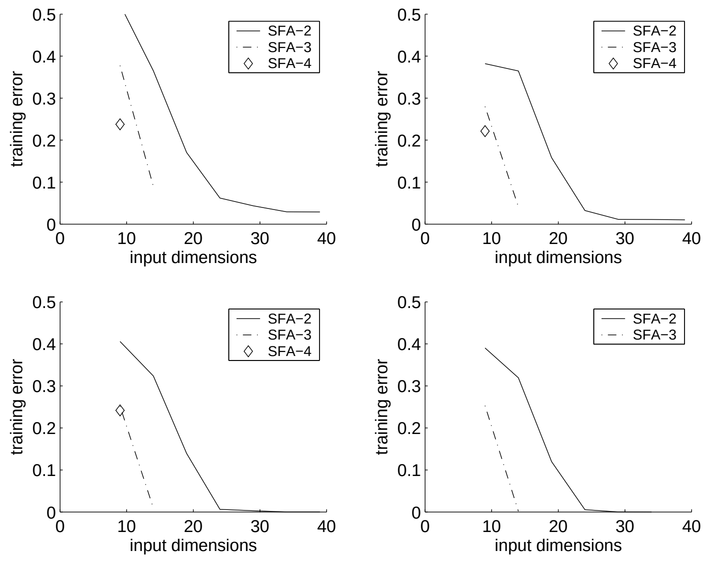
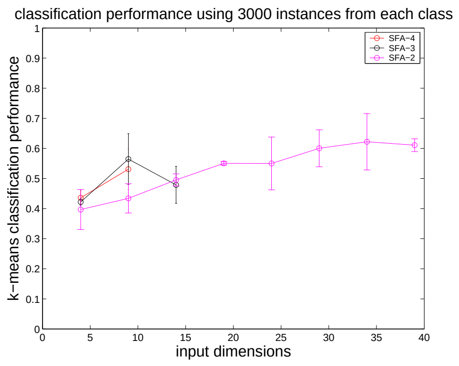
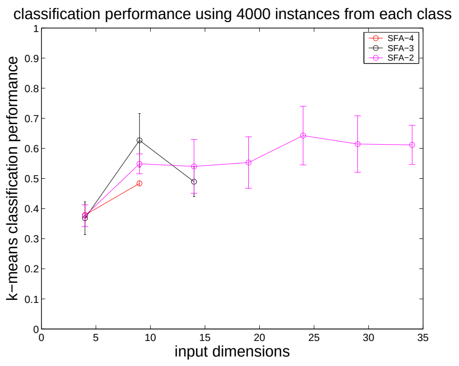


Figure 4: Average training errors of linear classifiers trained on the outputs of single SFA nodes, with different training set sizes: 3000 (top left), 4000 (top right), 5000 (bottom left), 6000 (bottom right) instances per class.

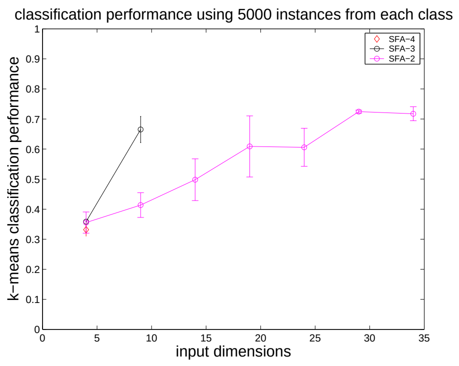
Overall, the best classification performance on the test set was achieved with a SFA-2 network using 29-dimensional inputs at each node, which corresponds to a 319-dimensional expanded space in which SFA finds the directions of minimum average rate of change. K-means correct classification rate achieved in this case was a mean rate of 0.7244, with a standard deviation of 0.005.



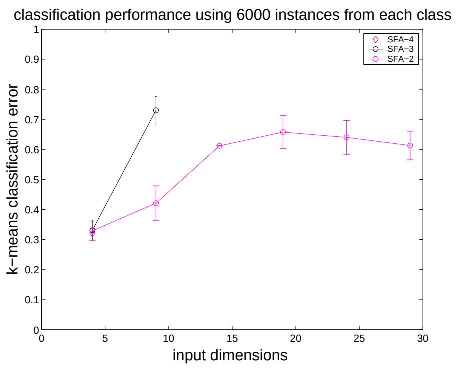
(a)



(b)



(c)



(d)

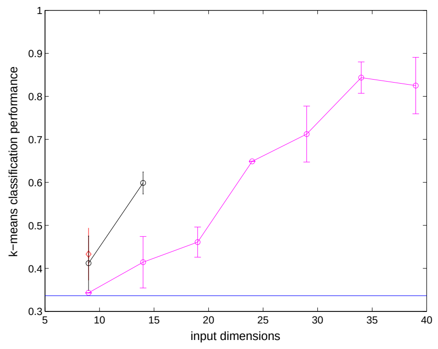
Figure 5: K-means correct classification performance of networks with different types of SFA nodes and reduced input dimensionalities under different training set sizes: (a) 3000 instances, (b) 4000 instances, (c) 5000 instances and (d) 6000 instances from each of the 10 classes.

Generalization to Unseen Classes - Using Single SFA Nodes

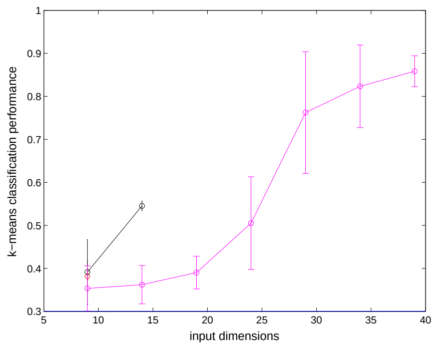
We now present the results for single SFA nodes. Here, the 256-dimensional raw input was first reduced to a much lower-dimensional input, using a linear SFA step, SFA-1, and then fed to a SFA-2, SFA-3 or a SFA-4 node. Similar to Figure 5, Figure 6 shows k-means classification performances of different types of single SFA nodes, with different input dimensionalities, under different training set sizes: again (a) 3000, (b) 4000, (c) 5000 and (d) 6000 instances from each of the 10 classes.

Here for comparison, we also include the k-means correct classification performance of PCA nodes with 7 components, i.e. PCA nodes which extract the leading 7 orthogonal directions of maximum variance of the 256-dimensional training set (blue lines in (a), (b), (c) and magenta line in (d)). Again, in each subfigure, x-axis is the reduced input dimensionality, and y-axis is the k-means correct classification rate on the test set. In (d), we were only able to calculate values for SFA-2 and SFA-3 nodes (red and black lines, respectively), so (d) does not include classification performance values for SFA-4 nodes. For larger training sets, i.e. (c) and (d), we observe a sharper increase in classification performance, as the input dimensionality is increased, whereas performance increases more gradually with smaller training sets, i.e. (a) and (b); note especially the sharp increase in performance as the input dimensionality is increased from 19 to 24 in (c) and (d).

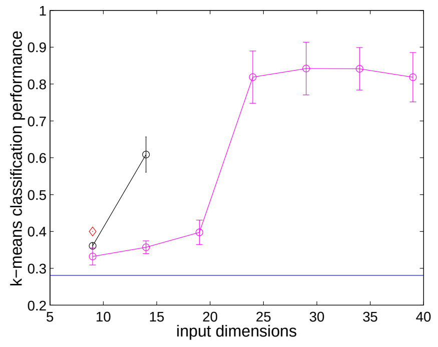
Overall, the best classification performance on the test set was achieved with a SFA-2 node of input dimensionality 29. K-means correct classification rate achieved in this case was a mean rate of 0.8711, with a standard deviation of 0.0004.



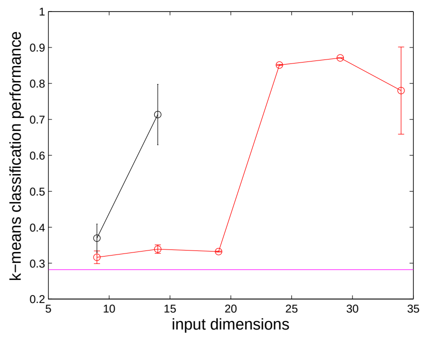
(a)



(b)



(c)



(d)

Figure 6: K-means classification performances of different types of single SFA nodes, with different input dimensionalities, under different training set sizes: again (a) 3000, (b) 4000, (c) 5000 and (d) 6000 instances from each of the 10 classes.

Comparison of Network Models and Single SFA Nodes

Figure 7 compares the k-means classification performances of network models with SFA-2 nodes and that of single SFA-2 nodes. We observe that for small input dimensions, say, up to 20 inputs, it would be more advantageous to use a network model, since for such small input dimensions, network models outperform single SFA nodes with the same dimensionality. On the other hand, for larger input dimensions, single SFA nodes outperform network models in terms of classification on the test set. For network models, as the input dimension becomes larger, the increase in classification performance becomes more gradual, or even vanishes with possibly a slight decrease (red dashed line in Figure 7). This stands to reason, since with the same number of input dimensions in the nodes, much more parameters need to be learned in a network model compared to a single SFA node. In fact, with the architecture that we are using here, the number of parameters to be estimated in a network with, say, SFA-2 nodes is 10 times larger than the number of parameters to be estimated in a single SFA-2 node with the same number of input dimensions. So, the kind of overfitting observed with the network models in Figure 7 is to be expected. One advantage of network models would be to make some infeasible computations more feasible, by simply dividing up the computation into smaller and more feasibly computable problems. Another advantage, especially when using non-linear nodes in the network as in our case, would be to increase the complexity of the model. For instance, in our case, using two layers of SFA-2 nodes would make the model essentially equivalent to a SFA-4 node. But, as we see here, that comes with the cost of increasing the necessary training set size needed to estimate the parameters of the model.

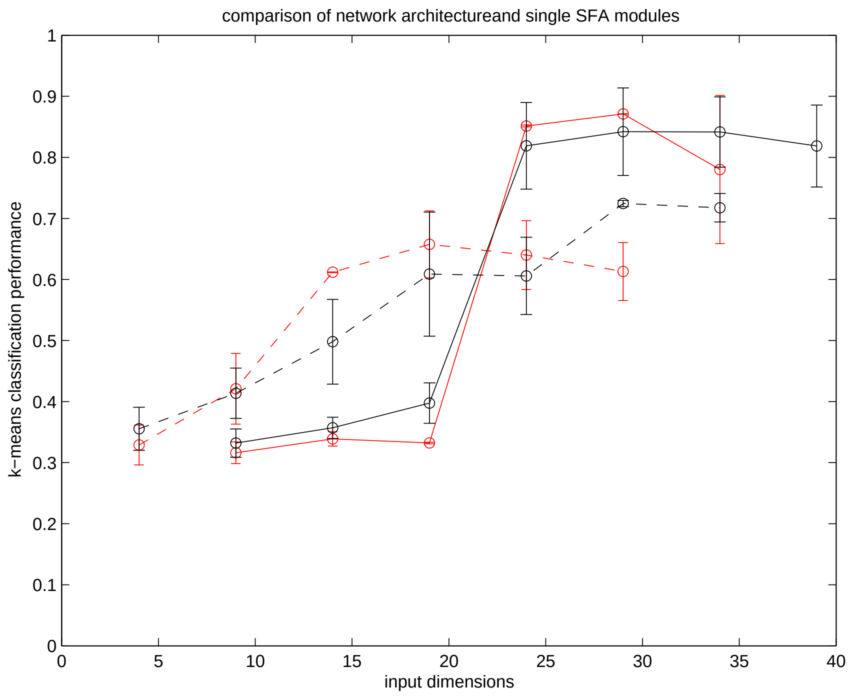


Figure 7: Comparison of the k-means classification performances of network models and single SFA nodes. Dashed lines correspond to network models, continuous lines to single SFA nodes. Red represents the cases where the training set size is 6000 instances per class and black represents the cases where the training set size is 5000 instances per class. All data are from SFA-2 type nodes and networks with SFA-2 nodes.

Dependence on the Training Set Size

Figure 8 shows the dependence of classification performance on training set size for the case of single SFA-2 nodes. Again, for larger training sets, we observe a sharper increase in classification performance, as the input dimensionality is increased, whereas performance increases more gradually with smaller training sets.

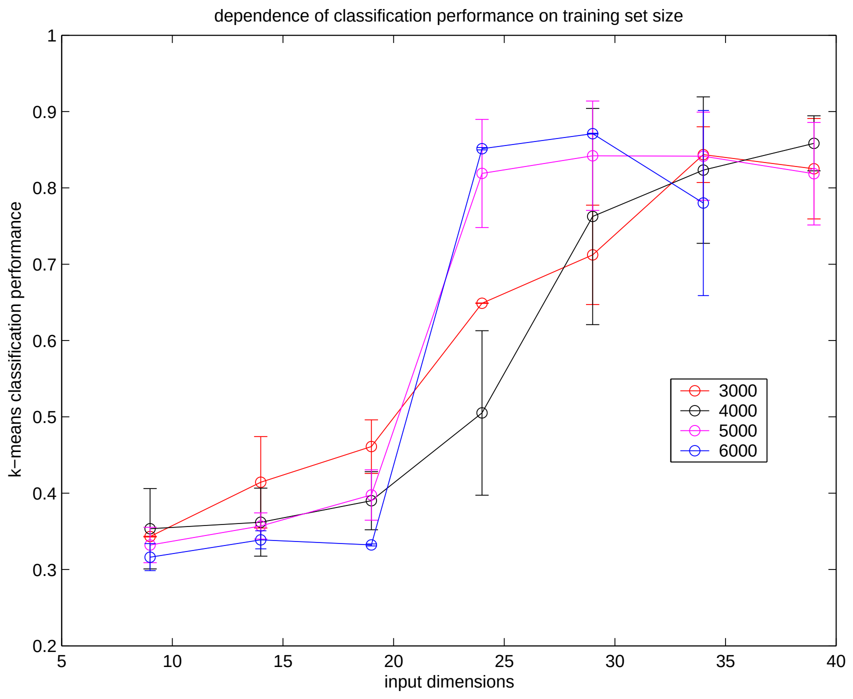


Figure 8: Dependence of classification performance on training set size. All data are from single SFA-2 nodes with different input dimensions

Rotation and Translation Invariance with Letters

We faced a difficulty with the generation of datasets containing variation in only rotation or in only scale size. SFA nodes require large amounts of data for training, but there is simply no room for generating 6000 sufficiently distinct images of size 32x32 (down-sampling to 16x16 makes it even worse) with all having different angles of rotation. Small angles of rotation do not make any difference to the image. We first reminded ourselves that it was the derivative of the training signal that is relevant for the training of a SFA node (and not single images themselves), and tried generating randomly rotating letters with steps of 1^0 between the angles of rotation of consecutive images, either increasing or decreasing (determined randomly at each step). But this wouldn't do, since, we were using a linear approximation for calculating the derivative signal and no matter what the signal might be like 20 steps ago, with a linear approximation, the only thing that matters for the calculation of the derivative is what the signal was like just one step ago, or will be like one step ahead. So most of the training set turn out to be redundant for the computation of slow components. To all intents and purposes, such a training set is effectively equivalent to another one which just traverses the available rotations from one end to the other and then back again (say, from 30^0 to -29^0 and back to 30^0 in steps of 1^0). This is only true for a linear approximation to the derivative signal. If we were using higher-order approximations, these two datasets would not be effectively equivalent. We have tried using higher-order approximations, but this proved to be infeasible for large datasets.

The same problem applies to the generation of a dataset containing variation in only scale size.

To overcome this problem, we decided to generate a dataset containing simultaneous variation in *both* position (random translation) *and* planar rotation. The same strategy was applied to generate a dataset containing simultaneous variation in *both* position *and* scale size.

Thus, for the following set of experiments, we prepared 15 letter stimuli undergoing planar rotation as well as random translation as in the previous subsection. From one frame to the next the angle of rotation varied in steps of 1^0 between 30^0 and -29^0. The change in rotation angle was ordered, i.e. it always went from -29^0 to 30^0 and back in steps of 1^0. This also allowed us to assess generalization performance to unseen instances. The rotation was implemented with the imrotate function in Matlab Image Processing Toolbox using the bicubic interpolation method. The rotated image was then cropped to its original size (32x32), using the 'crop' option, and embedded into the 64x64 background image, across which it also randomly moved. The final image was obtained by down-sampling to 16x16 as in the previous section.

The first 10 letters were used for training and the remaining 5 letters were reserved for testing the generalization performance of the models to previously unseen classes. We also assessed the generalization performance of the models to unseen instances by training them only with certain views of the letters. As in the previous section, generalization to unseen classes was assessed by the average correct classification rate of multiple applications of k-means clustering on the outputs of SFA models. Generalization to unseen instances, on the other hand, was assessed by the average test error rates of linear classifiers trained on the outputs of SFA models.

As an illustration, we first show in Figure 9 the projection of the training set on the first 3 of the 7 slow components extracted by a hierarchical network of SFA-3 nodes with 14 input dimensions. Note how letters with ‘similar’ features are clustered together, e.g. the letters C, D, G form a cloud, A, B, F form a separate cloud and I and J are a separate cloud. The training set in this illustration consisted of 3000 instances from each class, and half of the views of each letter were not shown to the model during training, so that the model saw only 30 views (corresponding to the views with odd angles of rotation, i.e. -29^0, -27^0, ..., 29^0) of each letter presented 100 times at random positions. Average training error of a (multivariate) linear classifier trained on the 7-dimensional output of this model was 0.22.

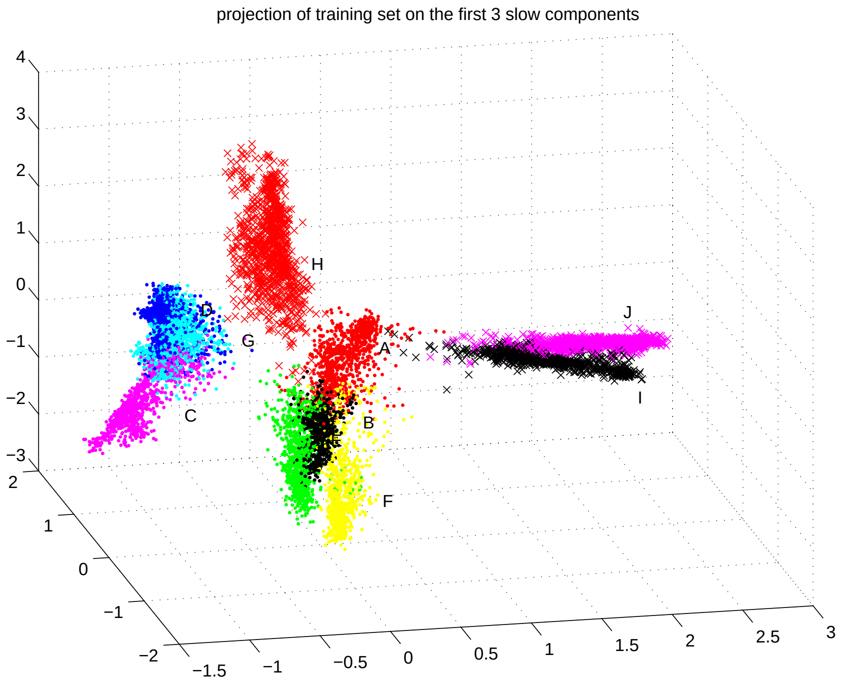


Figure 9: Projection of the training set on the first 3 of the 7 slow components extracted by a hierarchical network of SFA-3 nodes with 14 input dimensions.

Generalization to Unseen Instances

For testing generalization to unseen instances, the models were presented only with certain views (corresponding to only certain angles of rotation) of the letters. In the first experiment, every second view of each of the 10 training letters was retained for training (corresponding to the views with odd angles of rotation, i.e. -29^0, -27^0, ..., 29^0). So the models saw only 30 views of each letter presented 100 times at random positions. In a second condition, every third view of each object was retained for training, so that the models saw only 20 views of each letter presented 100 times at random positions. Finally, in the third condition, every sixth object was retained for training, so that the models saw only 10 views of each letter presented 100 times at random positions. In each condition, all the images not used for training were included in the test set.

For each condition, linear classifiers were trained on the outputs of the models. For each of the models compared below, the outputs were always 7-dimensional, that is, for SFA models, we retained only the first 7 slow components and for the PCA model, the leading 7 principal components were retained. Average classification errors of the linear classifiers were computed.

Figure 10 compares the classification performances of 7 types of models: a hierarchical network of SFA-2 nodes (hSFA-2), a hierarchical network of SFA-3 nodes (hSFA-3), a hierarchical network of SFA-4 nodes (hSFA-4), a single SFA-2 node (sSFA-2), a single SFA-3 node (sSFA-3), a single SFA-4 node (sSFA-4), and finally a PCA node with 7 components (PCA-7). We also considered the effect of the number of input dimensions on classification performance; Figure 10 shows, for each type (except for the PCA model), the performance of the best model among the models of that type. For instance, among the single SFA-2 type models, the model with 32 input dimensions had the best average (over three conditions mentioned above) classification performance on the test set. So, the average classification errors of the sSFA-2 model shown in Figure 10 is that of a sSFA-2 model with 32 dimensions. Average training errors are shown in blue, average classification errors on test sets is shown in red.

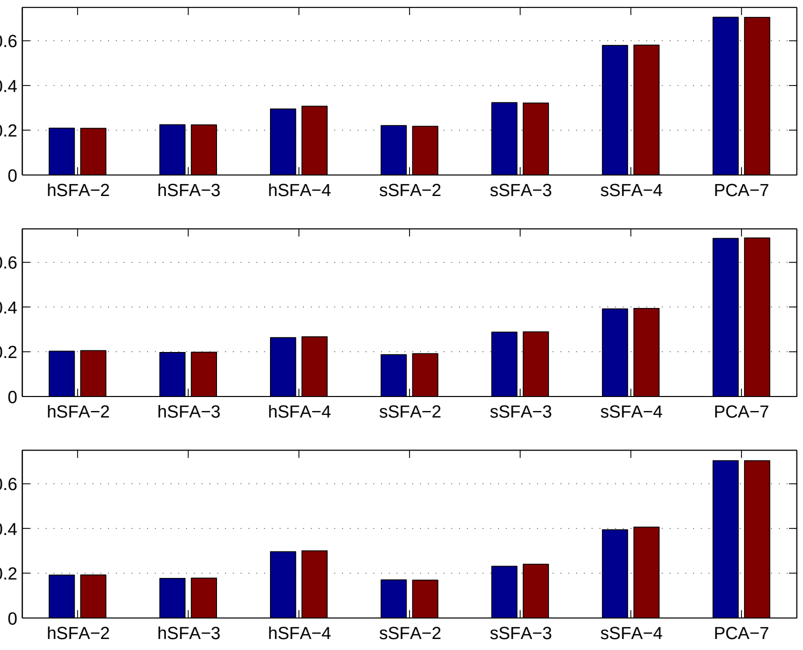


Figure 10: Classification performances of 7 types of models. Average training errors are shown in blue, average classification errors on test sets are shown in red. Top row corresponds to the condition where the training set consisted of every second view of each letter, middle row to the condition where the training set consisted of every third view of each letter and the bottom row to the condition where the training set consisted of every sixth view of each letter.

Overall, the best performances were achieved with single and network SFA-2 models (for the sSFA-2 model, 0.177 and 0.1781 for training and test errors averaged over three different training set sizes). SFA-3 (except perhaps hierarchical SFA-3 models) and SFA-4 models did not do as well as the SFA-2 models. However, for SFA-3 and SFA-4 type models, using a hierarchical network model can be observed to improve the classification performance compared to single SFA-3 and SFA-4 nodes. Average training and test errors are comparable in each case and comparing the three rows in Figure 10 we can see that training set size does not have a significant effect on generalization to unseen instances. Also note that average training errors are significantly higher than those achieved on the random translation dataset with similar models and training set sizes. For instance, with a training set size of 3000 instances per class and using a hierarchical network of SFA-3 nodes with 14 input dimensions, average training error on the random translation dataset was 0.013, whereas on the rotation and translation dataset the average training error was 0.2245. Most likely, this can be explained by the fact that rotation plus random translation dataset contains more variation, hence, is probably more difficult to learn than learning just translation invariance.

Generalization to Unseen Classes

Generalization to unseen classes was assessed by the k-means classification procedure described in the previous section. The test set in this case consisted of all views of 5 letters not used during training, that is, 60 views of each letter presented 100 times at random positions, making a total of 6000 instances per letter. Figure 11 shows the k-means correct classification rates of hierarchical networks of SFA-2, SFA-3 and SFA-4 nodes. The models compared in the graph were trained on 30 views (every second) of each of the 10 training letters presented 100 times at random positions. Again only the best performing models of each type are shown in the figure. A hierarchical network of SFA-3 nodes with 14 input dimensions performed the best, with mean k-means correct classification rate of 0.5702 and standard deviation of 0.0594. This is somewhat similar to the mean k-means correct classification rate achieved with the best performing network SFA-3 model trained on the random translation dataset with the same size, which was 0.5646 with a standard deviation of 0.0845. On the other hand, SFA-2 and SFA-4 models showed significantly better generalization performances on the random translation dataset than on the rotation plus translation dataset. Compared to the mean k-means correct classification rates of 0.6220 and 0.4358 on the random translation dataset, the best performing SFA-2 and SFA-4 network models, trained on the same-sized training sets, had k-means correct classification rates of 0.4231 and 0.3711, respectively, on the random translation plus rotation dataset.

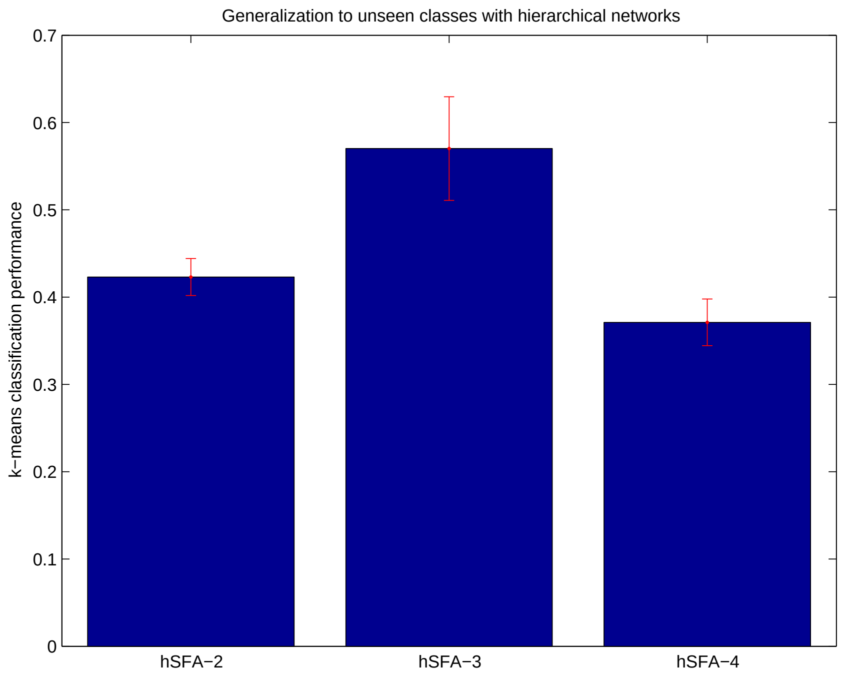


Figure 11: K-means correct classification rates of hierarchical networks of SFA-2, SFA-3 and SFA-4 nodes. Only the best performing models of each type are shown.

Scale and Translation Invariance with Letters

As explained at the beginning of the previous subsection, for the experiments in this subsection letters were generated that underwent scale changes simultaneously with random translation. Scale changes were modeled by simply up-sampling the 32x32 pixel original images to larger sizes. Sizes up to 61x61 pixels were used, making a total of 30 different scale sizes. The change in scale was ordered, i.e. it always went from 32x32 pixels to 61x61 pixels and back in steps of 1 pixel per edge from one frame to the next. Translation, as always, was modeled as random movement across a 64x64 background image. At a certain step, all the pixels of the image containing the letter randomly moved one pixel on the background image along one of the available directions. The final images were obtained by down-sampling the 64x64 image to 16x16 pixels.

Again as in the previous subsection, the first 10 letters were used for training and the remaining 5 letters were reserved for testing the generalization performance of the models to previously unseen classes. Generalization to unseen instances was assessed by training the models only with certain views of the letters. Generalization to unseen classes was assessed by the average correct classification rate of multiple applications of k-means clustering on the outputs of SFA models. Generalization to unseen instances was assessed by the average test error rates of linear classifiers trained on the outputs of SFA models.

As an example, we first show in Figure 12 the projection of the training set on the first 3 of the 7 slow components extracted by a hierarchical network of SFA-2 nodes with 32 input dimensions. Note how the letters E and F on the one hand, and C and G on the other are projected onto neighboring places on the 3-dimensional space and thus form distinct clusters. The training set in this illustration consisted of 3000 instances from each class, and half of the views of each letter were not shown to the model during training, so that the model saw only 15 views (corresponding to the views with odd scale sizes, i.e. 33x33, 35x35, ..., 61x61 pixels) of each letter presented 200 times at random positions. Average training error of a (multivariate) linear classifier trained on the 7-dimensional output of this model was 0.0004.

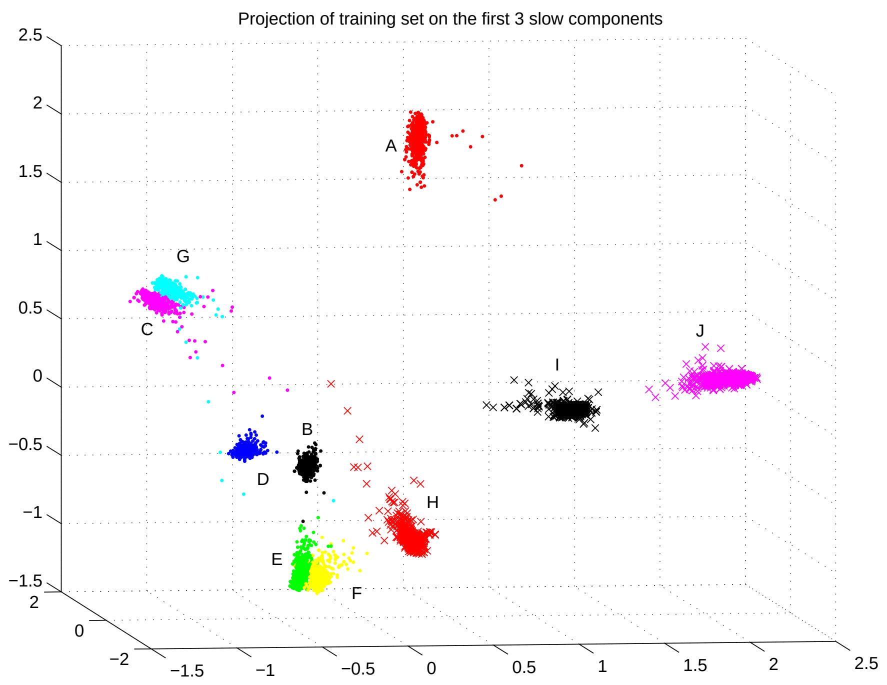


Figure 12: Projection of the training set on the first 3 of the 7 slow components extracted by a hierarchical network of SFA-2 nodes with 32 input dimensions.

Generalization to Unseen Instances

For testing generalization to unseen instances, the models were presented only with certain views (corresponding to only certain scale sizes) of the letters. In the first experiment, every second view of each of the 10 training letters was retained for training (corresponding to the views with odd scale sizes, i.e. 33x33, 35x35, ..., 61x61 pixels). So the models saw only 15 views of each letter presented 200 times at random positions. In a second condition, every third view of each object was retained for training, so that the models saw only 10 views of each letter presented 200 times at random positions. Finally, in the third condition, every sixth object was retained for training, so that the models saw only 5 views of each letter presented 200 times at random positions. In each condition, all the images not used for training were included in the test set.

For each condition, linear classifiers were trained on the outputs of the models. For each of the models compared below, the outputs were always 7-dimensional, that is, for SFA models, we retained only the first 7 slow components and for the PCA model, the leading 7 principal components were retained. Average classification errors of the linear classifiers were computed.

Figure 13 compares the classification performances of 6 types of models: a hierarchical network of SFA-2 nodes (hSFA-2), a hierarchical network of SFA-3 nodes (hSFA-3), a hierarchical network of SFA-4 nodes (hSFA-4), a single SFA-2 node (sSFA-2), a single SFA-3 node (sSFA-3) and a single SFA-4 node (sSFA-4). Results for the PCA model with 7 components were not included in the figure, since classification errors for this case were much higher than the SFA models considered here (on the order of 0.4 for both training and test errors and in all conditions). Again, we also considered the effect of the number of input dimensions on classification performance. Figure 13 shows, for each type, the performance of the best model among the models of that type. For instance, among the single SFA-3 type models, the model with 14 input dimensions had the best average (over three conditions mentioned above) classification performance on the test set. So, the average classification errors of the sSFA-3 model shown in Figure 13 is that of a sSFA-3 model with 14 dimensions. Average training errors are shown in blue and average classification errors on test sets is shown in red.

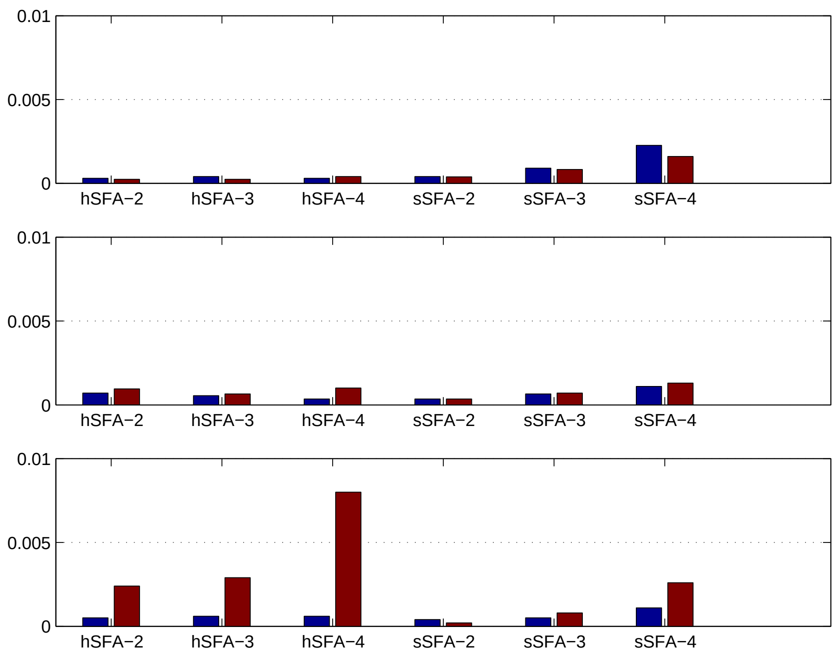


Figure 13: Classification performances of 6 types of models. Average training errors are shown in blue, average classification errors on test sets are shown in red. Top row corresponds to the condition where the training set consisted of every second view of each letter, middle row to the condition where the training set consisted of every third view of each letter and the bottom row to the condition where the training set consisted of every sixth view of each letter.

The first thing to be noted about these results is the significantly low classification errors achieved by all models. Comparing Figure 10 with Figure 13, we see that in the random translation plus rotation dataset, the error rates are all on the order of 0.2, e.g. the best training error rate achieved was 0.177. However, in the random translation plus scale change dataset, we can see that the error rates are much lower than 0.005 (on the order of 0.001). This might perhaps be explained by hypothesizing that scale invariance is much easier to learn than rotation invariance, which has also been demonstrated in Wiskott & Sejnowski (2002) for the case of one-dimensional, synthetic datasets. However, what is perhaps more surprising is that the error rates achieved in most of the cases here are even lower than the ones achieved in the random translation dataset with comparable models and training set sizes. To give an example, with a training set size of 3000 instances per class and using a hierarchical network of SFA-3 nodes with 14 input dimensions, average training error on the random translation dataset was 0.013, whereas on the translation and scale change dataset the average training error was 0.00055. But surely, one might think, learning translation invariance must be easier than learning translation and scale invariance.

This apparent tension might be resolved by noting one important feature of the translation plus scale change dataset. In this dataset, as the scale size grows larger and larger than the original image size, which is, 32x32 pixels, the number of positions that the letter could occupy on the background image decreases. So, for instance, there are 33x33 different ways to position an image of size 32x32 on a background image of size 64x64, whereas there are just 4x4 different ways to position an image of size 61x61 on the same background image. So, we might hypothesize that in this dataset, distinct from the translation and translation plus rotation datasets, changes in scale size offset some of the variation that would be induced by translation. Wiskott & Sejnowski (2002) has also demonstrated, for the case of one-dimensional, synthetic datasets, that learning scale invariance is easier than learning translation invariance. The apparent tension in our results can thus be explained by combining these two pieces together: that some of the variation that would be induced by translation is offset by changes in scale, and this variation caused by scale change, in turn, is easier to accommodate than translation.

Other points worth noting about these results are the increases in test errors observed for the hierarchical models (largest for the hierarchical SFA-4 model) when the training set size becomes smaller (bottom row in Figure 13) and the relatively worse performance of sSFA-4 model (and to some extent, that of sSFA-3 as well) compared with its hierarchical counterpart.

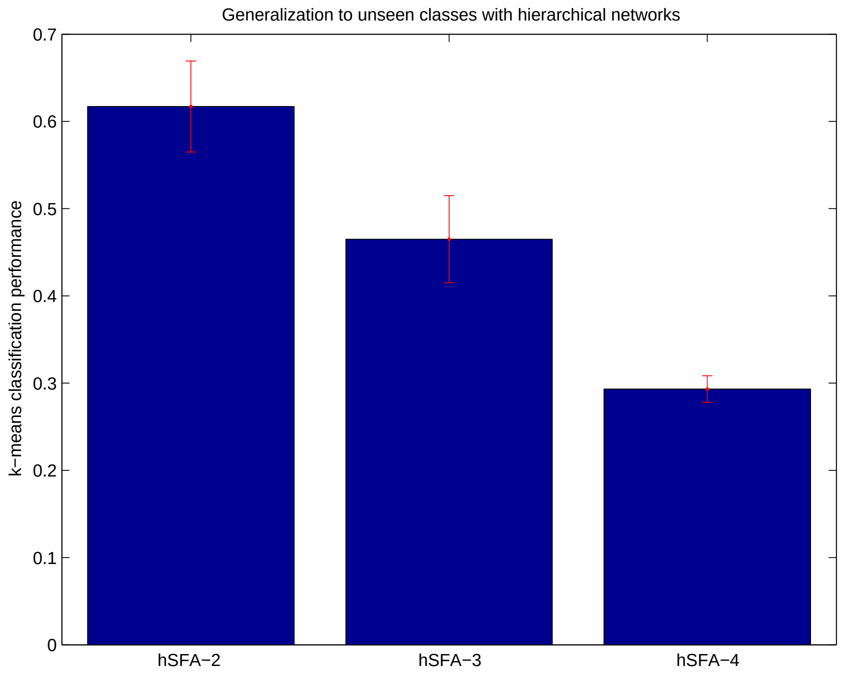


Figure 14: K-means correct classification rates of hierarchical networks of SFA-2, SFA-3 and SFA-4 nodes. Only the best performing models of each type are shown.

Generalization to Unseen Classes

Similar to previous sections and subsections, generalization to unseen classes was assessed by the k-means classification procedure. The test set in this case consisted of all views of 5 letters not used during training, that is, 30 views of each letter presented 200 times at random positions, making a total of 6000 instances per letter. Figure 14 shows the k-means correct classification rates of hierarchical networks of SFA-2, SFA-3 and SFA-4 nodes. The models compared in the graph were trained on 15 views (every second) of each of the 10 training letters presented 200 times at random positions. Again only the best performing models of each type are shown in the figure. A hierarchical network of SFA-2 nodes with 32 input dimensions performed the best, with mean k-means correct classification rate of 0.6171 and standard deviation of 0.0522, which is comparable to the mean k-means correct classification rate achieved with the best performing network SFA-2 model trained on the random translation dataset with the same size: a correct classification rate of 0.6220 with a standard deviation of 0.0935.

Results on COIL-20 Database

COIL-20 database is an object database of 1440 views of 20 objects (72 each, and 5^o changes in viewpoint between adjacent views of the same object). Images are all 128x128 pixel grey-scale images. In this section, we consider the application of slow feature analysis to COIL-20 database. Objects moved randomly across a homogeneous 256x256 background image, simultaneously with a change in viewpoint. These images were then all down-sampled to 64x64 pixels. The pre-processing of these 64x64 images was similar to the one in Einhauser et al. (2005). All images were filtered through a bank of 24 complex Gabor filters (scales: 1/32, 1/16, 1/8, 1/4 pixel^{-1}, orientations: 0^0, 30^0, 60^0, 90^0, 120^0, 150^0), and in a way mimicking the behavior of complex cells in V1, each filtered image was spatially summed (i.e. absolute values of pixels of each filtered image were summed up to yield a single value per filter). The final representation is thus a 24-dimensional vector, each dimension of which is being called a complex cell (CC) unit. These CC units roughly signal the presence or absence of a particular feature in the image regardless of its position. So, this pre-processing already guarantees a significant amount of translation invariance, thus the real task here is to learn viewpoint invariance as in Einhauser et al. (2005). Given the low dimensionality of the input signal, the models to be considered in this section will always be single SFA nodes.

In a second series of experiments, the effect of ‘distractor’ objects on the training and testing performances of the models were tested. For this condition, a randomly selected distractor object was placed on the background image during both training and testing phases. Partial overlap between distractor and target objects was allowed, but the target image always appeared in the front, with the distractor object being seen occluded by the target in such cases. The remaining pre-processing stages in this condition were the same as in the first condition: images were first down-sampled to 64x64, then filtered through a bank of 24 complex Gabor filters and then spatially summed up to yield a single value per filter. In accordance with Einhauser et al. (2005), the first condition will be referred to as ‘the plain condition’ and the second one as ‘the cluttered condition’.

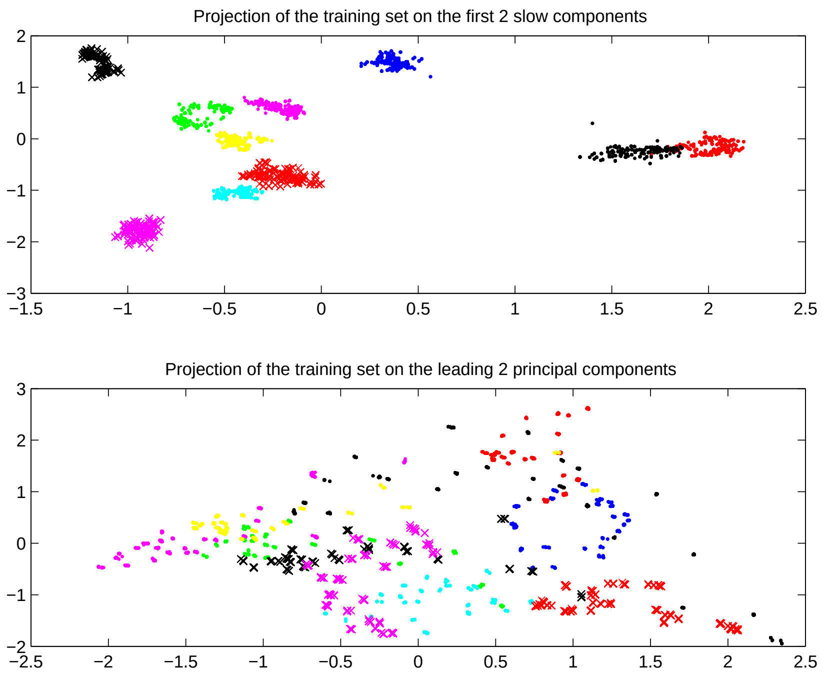


Figure 15: Projection of the training set on the first two slow components (top) and on the first two principal components (bottom) respectively.

Results for the Plain Condition

We first show, as an illustration, in Figures 15 and 16 the projection of the training set on the first two of the seven slow components extracted by a single SFA-3 node with 9 input dimensions, on the first two principal components, and on the most discriminative two dimensions of the 24-dimensional input. The training set in this illustration consisted of 1200 instances each from the first 10 classes, and every third view of each object was shown to the model during training, so that the model saw only 24 views (out of 72) of each object presented 50 times at random positions. Average training error of a (multivariate) linear classifier trained on the 7-dimensional output of this model was 0, that is all 10 classes were pair-wise linearly separable. Indeed using just three of the seven slow components extracted by the SFA-3 model was sufficient for ensuring the pair-wise linear separability of all the classes in the training set.

The most discriminative two dimensions of the input shown in Figure 16 were found by a simple forward feature subset selection procedure. Visually, these figures again illustrate the usefulness of SFA as a pre-processing step for classification algorithms on invariant recognition tasks.

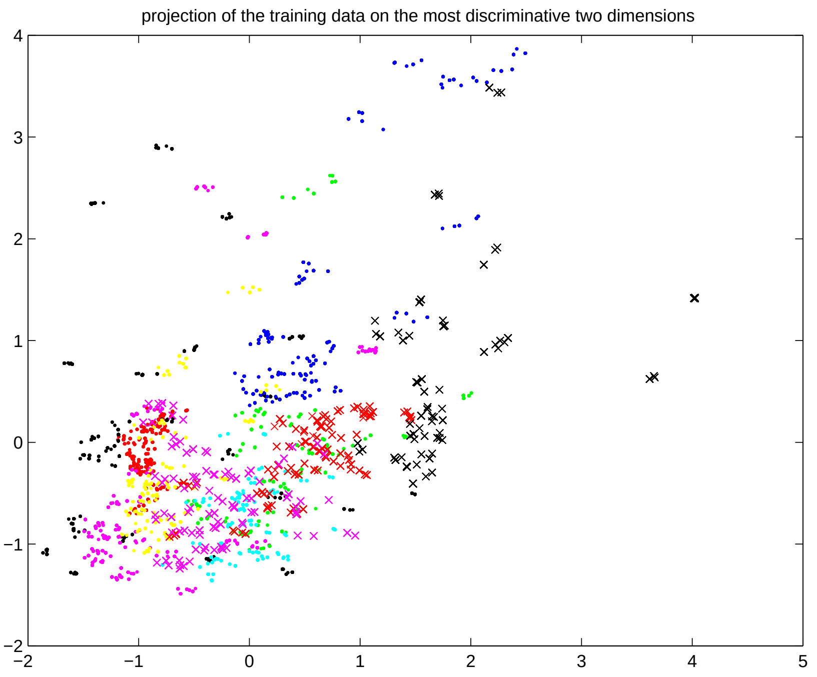


Figure 16: Projection of the training set on the most discriminative two dimensions of the input.

Invariance of the Slow Component Responses

In a separate experiment, we compared the invariance of the complex cell (CC) and slow component responses across different views of the same object. We trained a single SFA-3 node with 9 input dimensions on a training set consisting of 1800 instances each from the first ten objects in the database, i.e. every second view (out of 72 views) of the first ten objects was presented to the model 50 times at random positions during training. Then, the responses of 24 complex cell (CC) units, and 7 slow components to all views of all objects in the database (including the views and objects that were not in the training set), each presented 50 times at random positions, were computed.

As in Einhauser et al. (2005), for each view of each object, normalized average responses of a unit (CC or slow component) were computed by averaging the responses of the unit over 50 presentations of that view at different positions and normalizing this to zero mean and unit variance over the whole dataset. Figure 17 shows the normalized average responses of a selected complex cell (black) to each view of the 10 objects from the training set. Shown in red is the similarly normalized average responses of the first slow component to each view of the same 10 objects.

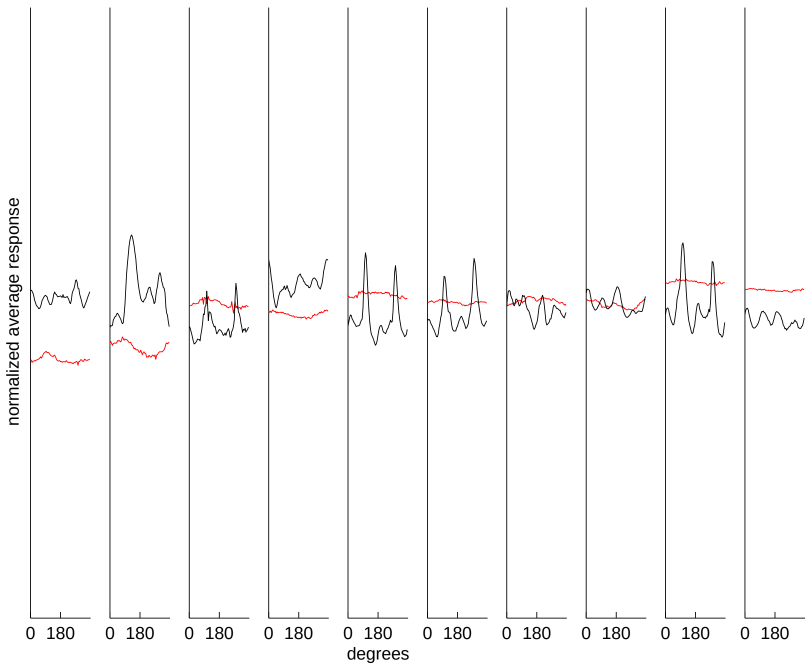


Figure 17: Normalized average responses of a selected complex cell (black) and the first slow component (red) to each view (72 views in total for each object) of the 10 objects from the training set. Different views of the objects are indexed by their angle of rotation in depth, from 0^0 to 360^0 with increments of 5^0 between adjacent views.

With the first slow component responses, the increase in the invariance of the responses to different views of the same object is clearly visible, compared to those of the selected complex cell. Whereas the complex cell exhibits several peaks and troughs in its response during the presentation of different views of the same object, the first slow component gives a more or less stable response to different views of the same object.

This invariance can be quantified by an invariance index as in Einhauser et al. (2005): one minus the standard deviation of the normalized average responses of the unit to different views of the same object. Figure 18 plots the mean invariance index for the complex cell units (CC) averaged over all 24 units, versus the mean invariance index for the slow components averaged over the seven slow components. Each dot corresponds to a different object. Again, it can be seen that most of the dots are above the diagonal indicating that on average the responses of the slow components to different views of the same object are more invariant than the responses of the complex cell units (CCs).

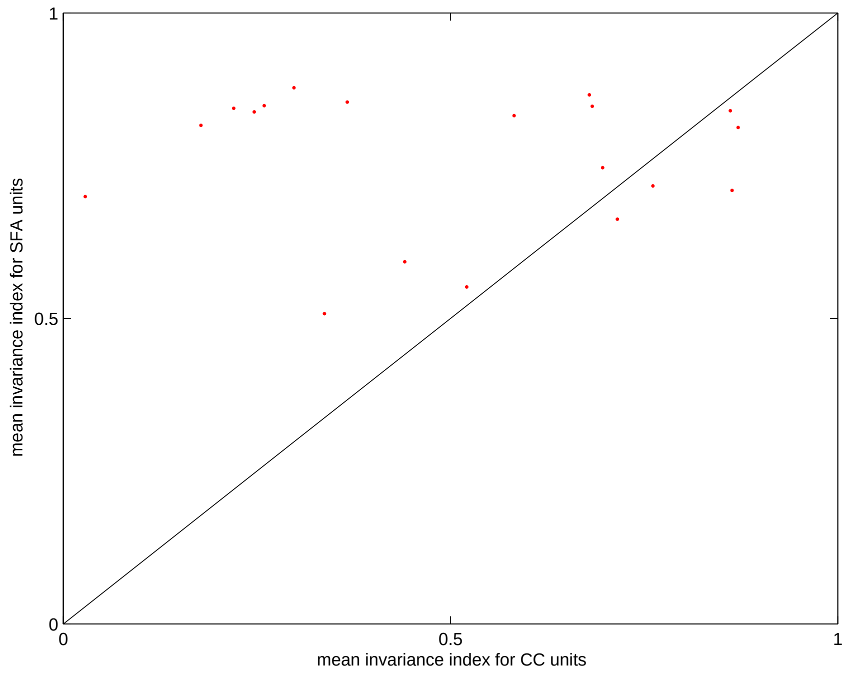


Figure 18: Mean invariance index for the complex cell units (CC) averaged over all 24 units, versus mean invariance index for the slow components averaged over all 7 slow components extracted by the SFA-3 algorithm.

Generalization to Unseen Instances

For testing generalization to unseen instances, the models were presented only with certain views of the objects. In the first experiment, every third view of each of the 10 training objects was retained for training. So the models saw only 24 views of each letter presented 50 times at random positions. In a second condition, every sixth view of each object was retained for training, so that the models saw only 12 views of each object presented 50 times at random positions. Finally, in the third condition, every ninth view was retained for training, so that the models saw only 8 views of each letter presented 50 times at random positions. In each condition, all views of the training objects not used for training were included in the test set.

For each condition, linear classifiers were trained on the outputs of the models. For each of the models compared below, the outputs were always 7-dimensional, that is, for SFA models, we retained only the first 7 slow components; for the PCA model, the leading 7 principal components were retained; and for the complex cell (CC) model, the most discriminative 7 complex cell responses were used as the input to the linear classifier (again there is no learning involved in the latter case). Average classification errors of the linear classifiers were computed both on the training and test sets.

Figure 19 shows the training and test errors of five different types of models for the three conditions mentioned above, corresponding to different training set sizes. The models compared in the figure are: single SFA-2, SFA-3 and SFA-4 models, PCA model with 7 components, and a model consisting of the 7 most discriminative of the complex cells. For the latter case, the most discriminative complex cells were again found by a forward feature selection routine. In addition, for the SFA models, dependence of the classification performance on the input dimensionality is graphed as well. In all conditions (except the last one where the training set consisted of every ninth view of each object presented 50 times at different positions), SFA models perform better than the PCA and CC models. Among SFA models, more complex models have lower training and test errors for a given number of input dimensions. However, especially with smaller training sets, more complex models (SFA-3 and SFA-4) very quickly begin to show signs of overfitting, as the number of input dimensions is increased. When the training set consists of just 400 instances per class (every ninth view of each object presented 50 times at different positions), presumably because of the insufficient number of training instances, PCA model performs better than all the other models on the test set for a large range of input dimensions.

Generalization to Unseen Classes

Similar to previous sections and subsections, generalization to unseen classes was assessed by the k-means classification procedure. In different conditions, the training set consisted of every second view (1800 instances per class) of 5, 10 and 15 objects from the database. A single SFA-3 node of input dimensionality 9 (with 7 extracted slow components, or outputs) was used for assessing the generalization performance. For comparison, generalization performance using the most discriminative 7 complex cell responses as the input to the k-means algorithm was computed (there was no learning involved in this case). For the latter case, the most discriminative complex cells were again found by a (greedy) forward feature selection algorithm. Figure 21 (upper panel) compares the k-means correct classification rates of the SFA-3 model

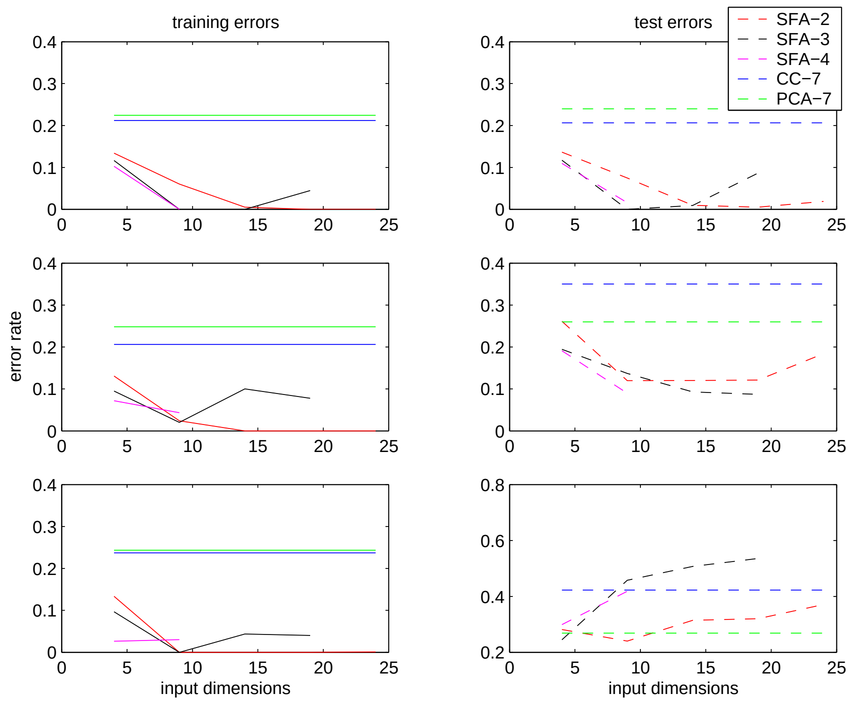


Figure 19: Results for the plain condition: training (left column) and test errors (right column) of five different types of models for different training set sizes: 1200 instances per object (top row), 600 instances per object (middle row), 400 instances per object (bottom row).

(of input dimensionality 9, and with 7 extracted slow components, or outputs) with those of the most discriminative 7 complex cell units (CC) under different training and test set sizes: 15 objects: 5 objects used for training, the remaining for testing; 10 objects: 10 objects used for training, the remaining for testing; 5 objects: 15 objects used for training, the remaining for testing. In all three conditions, SFA-3 model can be observed to have significantly better generalization to unseen classes. As the training set size increases (and the test set size decreases), accuracy rate increases for both models. In the condition where 1800 instances each from 10 objects were used as our training set, and all the views of the remaining 10 objects were used as our test set, the SFA-3 model had a correct classification rate of 0.7747 (with a standard deviation of 0.0296), and the most discriminative 7 complex cells had a correct classification rate of 0.6851 (with a standard deviation of 0.0101). Note that chance levels would be 0.067 for a test set consisting of 15 objects, 0.1 for 10 test objects and 0.2 for 5 test objects.

Results for the Cluttered Condition

In the cluttered condition, both training and testing were made in the presence of distractors. Distractors were always chosen randomly from the whole dataset. Otherwise, experimental procedures for the cluttered condition were the same as in the plain condition.

Generalization to Unseen Instances

Similar to the plain condition, for testing generalization to unseen instances, models were presented only with certain views of the objects. In the first experiment, every third view of each of the 10 training objects was retained for training. So the models saw only 24 views of each object presented 50 times at random positions. In a second condition, every sixth view of each object was retained for training, so that the models saw only 12 views of each object presented 50 times at random positions. Finally, in the third condition, every ninth view was retained for training, so that the models saw only 8 views of each letter presented 50 times at random positions. In each condition, all views of the training objects not used for training were included in the test set.

For each condition, linear classifiers were trained on the outputs of the models. For each of the models compared below, the outputs were always 7-dimensional, that is, for SFA models, we retained only the first 7 slow components; for the PCA model, the leading 7 principal components were retained; and for the complex cell (CC) model, the most discriminative 7 complex cell responses were used as the input to the linear classifier (again there is no learning involved in the latter case). Average classification errors of the linear classifiers were computed both on the training and test sets.

Figure 20 shows the training and test errors of five different types of models for the three conditions mentioned above, corresponding to different training set sizes. The models compared in the figure are: single SFA-2, SFA-3 and SFA-4 models, PCA model with 7 components, and a model consisting of the 7 most discriminative of the complex cells. For the latter case, the most discriminative complex cells were again found by a forward feature selection routine. In addition, for the SFA models, dependence of the classification performance on the input dimensionality is graphed as well.

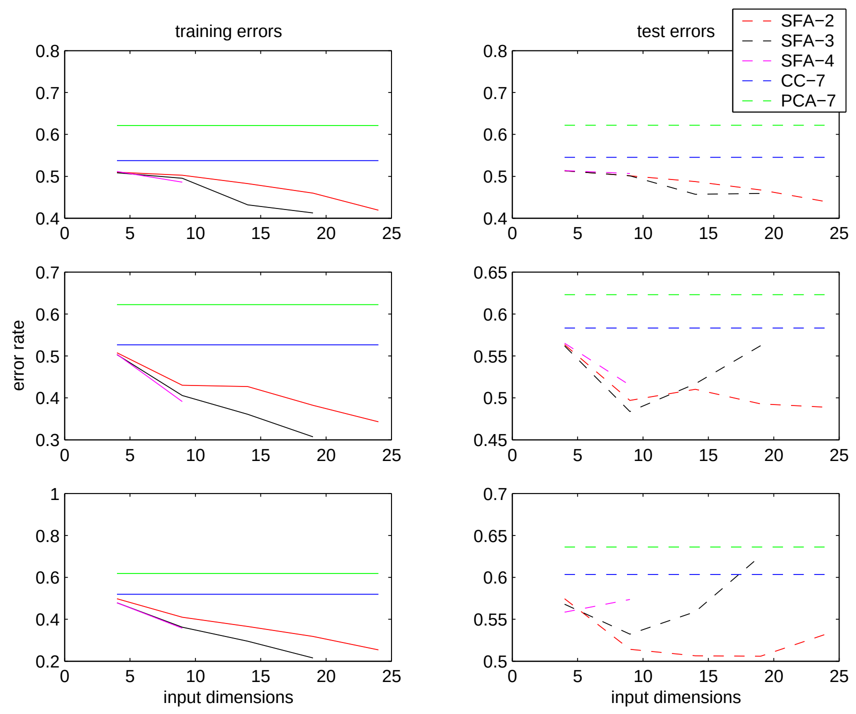


Figure 20: Results for the cluttered condition: training (left column) and test errors (right column) of five different types of models for different training set sizes: 1200 instances per object (top row), 600 instances per object (middle row), 400 instances per object (bottom row).

One can immediately notice that both training and test errors are overall significantly lower for the cluttered condition compared to the plain condition. For instance training errors are around 0.2 for the best of the SFA models, and the test errors are never better than about 0.48. Besides that, some of the similar trends observed in the plain condition can be observed here in the cluttered condition as well. For instance, SFA models outperform PCA and CC models alike in all the conditions. Among SFA models, more complex models have, in general, lower training and test errors for a given number of input dimensions. However, again especially with smaller training sets, more complex models (SFA-3 and SFA-4) very quickly begin to show signs of overfitting, as the number of input dimensions is increased. Compare, for instance, the black (SFA-3) and red (SFA-4) dashed lines (right column) in the second and third rows in Figure 20.

Generalization to Unseen Classes

Similar to the plain condition, generalization to unseen classes was assessed by the k-means classification procedure. In different conditions, the training set consisted of every second view (1800 instances per class) of 5, 10 and 15 objects from the database. A single SFA-3 node of input dimensionality 9 (with 7 extracted slow components, or outputs) was used for assessing the generalization performance. Again for comparison, generalization performance using the most discriminative 7 complex cell responses as the input to the k-means algorithm was computed (there was no learning involved in this case). For the latter case, the most discriminative complex cells were again found by a (greedy) forward feature selection algorithm. Figure 21 (upper panel) compares the k-means correct classification rates of the SFA-3 model (of input dimensionality 9, and with 7 extracted slow components, or outputs) with those of the most discriminative 7 complex cell units (CC) under different training and test set sizes: 15 objects: 5 objects used for training, the remaining for testing; 10 objects: 10 objects used for training, the remaining for testing; 5 objects: 15 objects used for training, the remaining for testing. Again in all three conditions, SFA-3 model can be observed to have better generalization to unseen classes. In conformity with the results for the plain condition, as the training set size increases (and the test set size decreases), accuracy rate increases for both models. However, similar to what we have observed for generalization to unseen instances, overall accuracy rates are significantly lower for the cluttered condition compared to the plain condition. For instance, in the condition where 1800 instances each from 10 objects were used as our training set, and all the views of the remaining 10 objects were used as our test set, the SFA-3 model had a correct classification rate of 0.2428 (with a standard deviation of 0.0220), and the most discriminative 7 complex cells had a correct classification rate of 0.2218 (with a standard deviation of 0.0126).

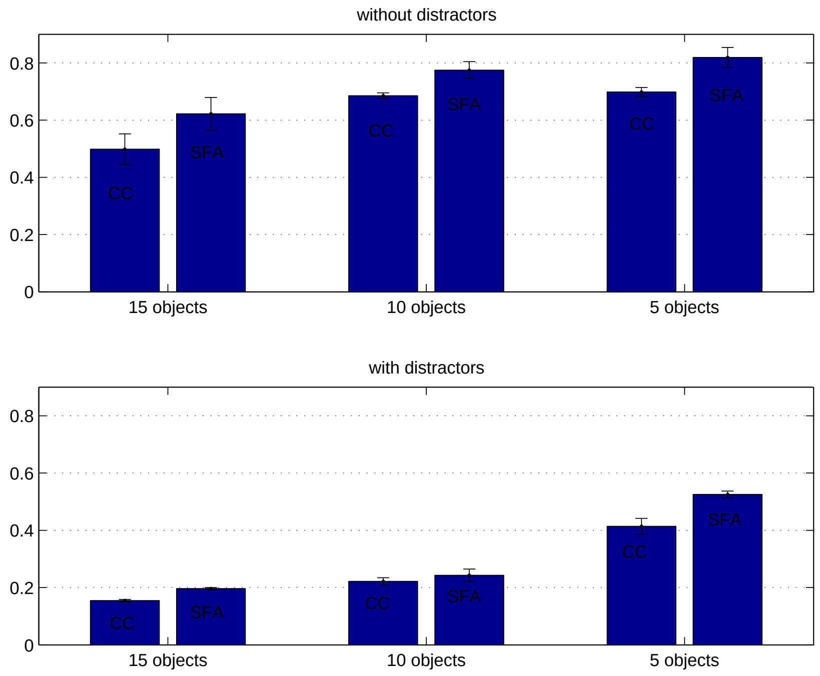


Figure 21: K-means correct classification rates of a single SFA-3 node of input dimensionality 9, with 7 extracted slow components compared with those of the best 7 complex cell units (CC) under different training and test set sizes: 15 objects: 5 objects used for training, the remaining for testing; 10 objects: 10 objects used for training, the remaining for testing; 5 objects: 15 objects used for training, the remaining for testing. Upper panel shows the results for the plain condition, the lower panel for the cluttered condition.

CHAPTER 6

DISCUSSION

We have tested various SFA implementing models on two different datasets: one dataset consisting of letters undergoing various transformations, and the COIL-20 object database. Several observations can be made from the results that we have obtained by our simulations.

First, SFA models can achieve quite satisfactory classification performances even with very simple classifiers. We have intentionally used simple classifiers (linear classifiers and an unsupervised classification procedure based on k-means clustering) to see if SFA itself can extract useful features for invariant object recognition tasks. Using more sophisticated classifiers might yield even better performances. Our results, thus, demonstrate the usefulness of SFA as perhaps a pre-processing step for classification algorithms on invariant recognition tasks.

Secondly, in conformity with the findings of Wiskott & Sejnowski (2002), some of the transformations have been observed to be easier to learn than others. For instance, learning scale invariance was easier than learning translation invariance or rotation invariance. This might be related to the intrinsic difficulty of the respective learning problems, or to an artifact in the generation of our data.

Thirdly, among SFA models tested here, for a given number of input dimensions, more complex models generally outperform simpler SFA models. As an instance of this general pattern, network models tend to do better than single node SFA models with the same number of input dimensions. However, especially with small training sets, more complex models (SFA-3 and SFA-4) very quickly begin to show signs of overfitting, as the number of input dimensions is increased. This is related to the exponential increase in the dimensionality of the expansion space, as the SFA model becomes more complex.

Fourthly, in general, SFA models require very large datasets for successful training. This makes it especially harder to apply SFA to high-dimensional dataset, such as images.

Fifthly, and related to the previous observation, for most cases, explicit expansion into a high-dimensional space used in SFA is impractical. Similar to Martinez & Bray (2003), a kernel-based version of SFA might be developed to remedy this. This would also make it easier to apply SFA to high-dimensional dataset, and therefore lift some of the memory costs involved in working with very large datasets required for the training of SFA.

Lastly, as we have observed in the cluttered condition for the COIL-20 database, SFA performs significantly worse in the presence of distractor objects. But, real-world visual stimuli always involve some degree of clutter, so this makes SFA, in its present form, unsuitable for more realistic visual learning scenarios. Indeed, all temporal coherence approaches are known to have problems with cluttered images (Wallis & Rolls, 1997). Therefore, it might be necessary to include further spatial continuity constraints, as well as temporal ones, to be able to accommodate such cases.

Other questions worth pursuing, but not pursued here include:

- Testing whether incorporation of further objectives besides temporal slowness, such as independence or sparseness as in Franzius et al. (2007) can improve the performance of the models. Experiments with models in which independent component analysis (ICA) nodes were cascaded after SFA nodes did not always yield consistent results and therefore were not included here.

- Interpretation of the functions learned at each SFA node. For a single SFA-2 node, it is possible to analytically determine its optimal excitatory and inhibitory stimuli, as in Berkes and Wiskott (2005). However, for SFA-3 or SFA-4 (or yet more complex) nodes, and similarly for network models, it is not clear, at the moment, how to determine the stimuli they prefer the most or the least, or the invariances they exhibit.

REFERENCES

Barlow, H. B. (1961). Possible principles underlying the transformation of sensory messages. In W. Rosenblith (Ed.), *Sensory Communication* (pp. 217-234). Cambridge, MA: MIT Press.

Berkes, P. (2003). Sfa_tk: Slow feature analysis toolkit for Matlab (v.1.0.1). Retrieved May 12, 2008, from http://itb.biologie.hu-berlin.de/~berkes/software/sfa-tk/sfa-tk.shtml.

Berkes, P. (2006). Temporal slowness as an unsupervised learning principle - self-organization of complex-cell receptive fields and application to pattern recognition. PhD Thesis, retrieved May 12, 2008, from http://edoc.hu-berlin.de/, urn:nbn:de:kobv:11-10058759.

Berkes, P., & Wiskott, L. (2005). Slow feature analysis yields a rich repetoire of complex cell properties. *Journal of Vision*, 5, 579-602.

Biederman, I., & Cooper, E. E. (1991). Evidence for complete translational and reectional invariance in visual object priming. *Perception*, 20, 585-593.

Biederman, I., & Cooper E. E. (1992). Size invariance in visual object priming. *Journal of Experimental Psychology: Human Perception and Performance*, 18 (1), 121-133.

Blaschke, T., Berkes, P., & Wiskott, L. (2006). What is the relationship between slow feature analysis and independent component analysis? *Neural Computation*, 18 (10), 2495-2508.

Blaschke, T., Zito, T., & Wiskott, L. (2007). Independent slow feature analysis and nonlinear blind source separation. *Neural Computation*, 19 (4), 994-1021.

Bray, A., & Martinez, D. (2002). Kernel-based extraction of slow features:

Complex cells learn disparity and translation invariance from natural images. *NIPS 2002 proceedings.*

Dill, M., & Fahle, M. (1998). Limited translation invariance of human visual pattern recognition. *Perception and Psychophysics*, 60 (1), 65-81.

Deco, G., & Schurmann, B. (2001). Predictive coding in the visual vortex by a recurrent network with Gabor receptive fields. *Neural Processing Letters*, 14, 107-114.

Einhäuser, W., Hipp, J., Eggert, J., Körner, E., & König, E. (2005). Learning viewpoint invariant object representations using a temporal coherence principle. *Biological Cybernetics*, 93 (1), 79-90.

Foldiak, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, 3, 194-200.

Franzius, M., Sprekeler, H., & Wiskott, L. (2007). Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3 (8), e166, retrieved May 12, 2008, from doi:10.1371/journal.pcbi.0030166.

Fukushima, K., Miyake, S., & Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Trans. on Systems, Man, and Cybernetics*, 13, 826-834.

Hoyer, P., & Hyvarinen, A. (2000). Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, 11, 191-210.

Martinez, D., & Bray, A. (2003). Nonlinear blind source separation using kernels. *IEEE Transactions on Neural Networks*, 14 (1), 228-236.

Nene, S. A., Nayar, S. K., & Murase, H. (1996). Columbia object image library (COIL-20). Retrieved May 12, 2008, from http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php.

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37, 3311-3325.

Sprekeler, H., Michaelis, C., & Wiskott, L. (2007). Slowness: An objective for spike-timing-dependent plasticity? *PLoS Computational Biology*, 3 (6), e112, retrieved May 12, 2008, from doi:10.1371/journal.pcbi.0030112.

Stone, J., & Bray, A. (1995). A learning rule for extracting spatio-temporal invariances. *Network: Computation in Neural Systems*, 6 (3), 429-436.

Stone, J. V. (1996). Learning perceptually salient visual parameters using spatiotemporal smoothness constraints. *Neural Computation*, 8, 1463-1492.

Tovee, M. J., Rolls, E. T., & Azzopardi, P. (1994). Translation invariance in the responses to faces of single neurons in the temporal visual cortical areas of the alert macaque. *Journal of Neurophysiology*, 72 (3), 1049-1060.

Wallis, G., & Rolls, E. (1997). A model of invariant object recognition in the visual system. *Progress in Neurobiology,* 51, 167-194.

Wiskott, L. (2003). How does our visual system achieve shift and size invariance? *CogPrints*, retrieved May 12, 2008, from http://cogprints.org/3321/1/Wiskott2003.pdf.

Wiskott, L., & Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14 (4), 715-770.