Unsupervised Deep Learning

by

Rostislav Goroshin

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2015

Professor Yann LeCun

Dedication

Parents and friends

Acknowledgements

Abstract

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Related Work

Chapter 3

Saturating Auto-Encoders

3.1 Introduction

An auto-encoder is a conceptually simple neural network used for obtaining useful data representations through unsupervised training. It is composed of an encoder which outputs a hidden (or latent) representation and a decoder which attempts to reconstruct the input using the hidden representation as its input. Training consists of minimizing a reconstruction cost such as L_2 error. However this cost is merely a proxy for the true objective: to obtain a useful latent representation. Auto-encoders can implement many dimensionality reduction techniques such as PCA and Sparse Coding (SC) [7] [21] [10]. This makes the study of auto-encoders very appealing from a theoretical standpoint. In recent years, renewed interest in auto-encoders networks has mainly been due to their empirical success in unsupervised feature learning [22][23][24][25].

When minimizing only reconstruction cost, the standard auto-encoder does not typically learn any meaningful hidden representation of the data. Well known theoretical

and experimental results show that a linear auto-encoder with trainable encoding and decoding matrices, W^e and W^d respectively, learns the identity function if W^e and W^d are full rank or over-complete. The linear auto-encoder learns the principle variance directions (PCA) if W^e and W^d are rank deficient [7]. It has been observed that other representations can be obtained by regularizing the latent representation. This approach is exemplified by the Contractive and Sparse Auto-Encoders [24] [22] [23]. Intuitively, an auto-encoder with limited capacity will focus its resources on reconstructing portions of the input space in which data samples occur most frequently. From an energy based perspective, auto-encoders achieve low reconstruction cost in portions of the input space with high data density (recently, [1] has examined this perspective in depth). If the data occupies some low dimensional manifold in the higher dimensional input space then minimizing reconstruction error achieves low energy on this manifold. Useful latent state regularizers raise the energy of points that do not lie on the manifold, thus playing an analogous role to minimizing the partition function in maximum likelihood models. In this work we introduce a new type of regularizer that does this explicitly for auto-encoders with a non-linearity that contains at least one flat (zero gradient) region. We show examples where this regularizer and the choice of nonlinearity determine the feature set that is learned by the auto-encoder.

3.2 Latent State Regularization

Several auto-encoder variants which regularize their latent states have been proposed, they include the sparse auto-encoder and the contractive auto-encoder[22][23][24]. The sparse auto-encoder includes an over-complete basis in the encoder and imposes a sparsity inducing (usually L_1) penalty on the hidden activations.

This penalty prevents the auto-encoder from learning to reconstruct all possible points in the input space and focuses the expressive power of the auto-encoder on representing the data-manifold. Similarly, the contractive auto-encoder avoids trivial solutions by introducing an auxiliary penalty which measures the square Frobenius norm of the Jacobian of the latent representation with respect to the inputs. This encourages a constant latent representation except around training samples where it is counteracted by the reconstruction term. It has been noted in [24] that these two approaches are strongly related. The contractive auto-encoder explicitly encourages small entries in the Jacobian, whereas the sparse auto-encoder is encouraged to produce mostly zero (sparse) activations which can be designed to correspond to mostly flat regions of the nonlinearity, thus also yielding small entries in the Jacobian.

3.2.1 Saturating Auto-Encoder through Complementary Nonlinearities

Our goal is to introduce a simple new regularizer which explicitly raises reconstruction error for inputs not near the data manifold. Consider activation functions with at least one flat region; these include shrink, rectified linear, and saturated linear (Figure 3.1). Auto-encoders with such nonlinearities lose their ability to accurately reconstruct inputs which produce activations in the zero-gradient regions of their activation functions. Let us denote the auto-encoding function $x_r = G(x, W)$, x being the input, W the trainable parameters in the auto-encoder, and x_r the reconstruction. One can define an energy surface through the reconstruction error:

$$E_W(x) = ||x - G(x, W)||^2$$

Let's imagine that G has been trained to produce a low reconstruction error at a particular data point x^*. If G is constant when x varies along a particular direction v, then the energy will grow quadratically along that particular direction as x moves away from x^*. If G is trained to produce low reconstruction errors on a set of samples while being subject to a regularizer that tries to make it constant in as many directions as possible, then the reconstruction energy will act as a *contrast function* that will take low values around areas of high data density and larger values everywhere else (similarly to a negative log likelihood function for a density estimator).

The proposed auto-encoder is a simple implementation of this idea. Using the notation $W = \{W^e, B^e, W^d, B^d\}$, the auto-encoder function is defined as

$$G(x, W) = W^d F(W^e x + B^e) + B^d$$

where W^e, B^e, W^d, and B^d are the encoding matrix, encoding bias, decoding matrix, and decoding bias, respectively, and F is the vector function that applies the scalar function f to each of its components. f will be designed to have "flat spots", i.e. regions where the derivative is zero (also referred to as the saturation region).

The loss function minimized by training is the sum of the reconstruction energy $E_W(x) = \|x - G(x, W)\|^2$ and a term that pushes the components of $W^e x + B^e$ towards the flat spots of f. This is performed through the use of a *complementary function* f_c, associated with the non-linearity $f(z)$. The basic idea is to design $f_c(z)$ so that its value corresponds to the distance of z to one of the flat spots of $f(z)$. Minimizing $f_c(z)$ will push z towards the flat spots of $f(z)$. With this in mind, we introduce a penalty of the form $f_c(\sum_{j=1}^{d} W_{ij}^e x_j + b_i^e)$ which encourages the argument to be in the saturation regime of the activation function (f). We refer to auto-encoders which include this

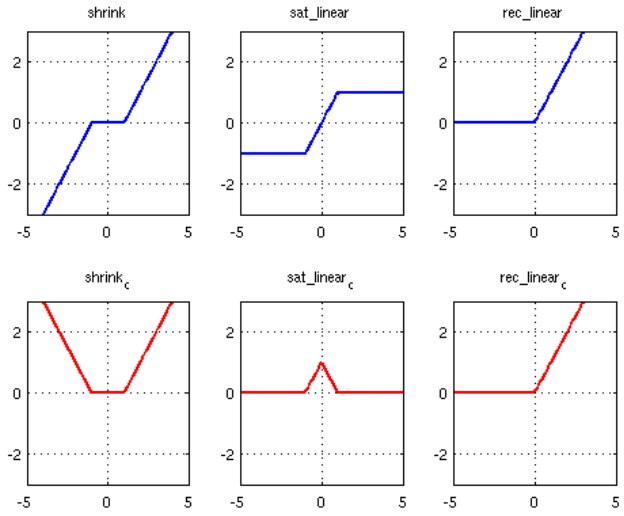


Figure 3.1: Three nonlinearities (top) with their associated complementary regularization functions(bottom).

regularizer as Saturating Auto-Encoders (SATAEs). For activation functions with zero-gradient regime(s) the complementary nonlinearity (f_c) can be defined as the distance to the nearest saturation region. Specifically, let $S = \{z \mid f'(z) = 0\}$ then we define $f_c(z)$ as:

$$f_c(z) = \inf_{z' \in S} |z - z'|. \tag{3.1}$$

Figure 1 shows three activation functions and their associated complementary nonlinearities. The complete loss to be minimized by a SATAE with nonlinearity f is:

$$L = \sum_{x \in D} \frac{1}{2} \|x - \left(W^d F(W^e x + B^e) + B^d\right)\|^2 + \alpha \sum_{i=1}^{d_h} f_c(W_i^e x + b_i^e), \tag{3.2}$$

where d_h denotes the number of hidden units. The hyper-parameter α regulates the trade-off between reconstruction and saturation.

3.3 Effect of the Saturation Regularizer

We will examine the effect of the saturation regularizer on auto-encoders with a variety of activation functions. It will be shown that the choice of activation function is a significant factor in determining the type of basis the SATAE learns. First, we will present results on toy data in two dimensions followed by results on higher dimensional image data.

3.3.1 Visualizing the Energy Landscape

Given a trained auto-encoder the reconstruction error can be evaluated for a given input x. For low-dimensional spaces ($\mathbb{R}^n$, where $n \leq 3$) we can evaluate the reconstruction error on a regular grid in order to visualize the portions of the space which are well represented by the auto-encoder. More specifically we can compute $E(x) = \frac{1}{2}||x - x_r||^2$ for all x within some bounded region of the input space. Ideally, the reconstruction energy will be low for all x which are in the training set and high elsewhere. Figures 3.2 and 3.3 depict the resulting reconstruction energy for inputs $x \in \mathbb{R}^2$, and $-1 \leq x_i \leq 1$. Black corresponds to low reconstruction energy. The training data consists of a one dimensional manifold shown overlain in yellow. Figure 3.2 shows a toy example for a SATAE which uses ten basis vectors and a shrink activation function. Note that adding the saturation regularizer decreases the volume of the space which is well reconstructed, however good reconstruction is maintained on or near the training data manifold. The auto-encoder in Figure 3.3 contains two encoding basis vectors (red), two decoding basis vectors (green), and uses a saturated-linear activation function. The encoding and decoding bases are unconstrained. The unregularized auto-encoder learns an orthogonal basis with a random orientation. The region of the space which is well reconstructed

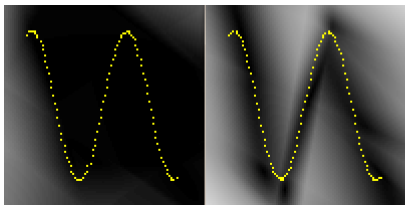


Figure 3.2: Energy surfaces for unregularized (left), and regularized (right) solutions obtained on SATAE-shrink and 10 basis vectors. Black corresponds to low reconstruction energy. Training points lie on a one-dimensional manifold shown in yellow.

corresponds to the outer product of the linear regions of two activation functions; beyond that the error increases quadratically with the distance. Including the saturation regularizer induces the auto-encoder basis to align with the data and to operate in the saturation regime at the extreme points of the training data, which limits the space which is well reconstructed. Note that because the encoding and decoding weights are separate and unrestricted, the encoding weights were scaled up to effectively reduce the width of the linear regime of the nonlinearity.

3.3.2 SATAE-shrink

Consider a SATAE with a shrink activation function and shrink parameter λ. The corresponding complementary nonlinearity, derived using Equation 1 is given by:

$$shrink_c(x) = \begin{cases} abs(x), \ |x| > \lambda \\ \\ 0, \ \text{elsewhere} \end{cases} .$$

Note that $shrink_c(W^e x + b^e) = abs(shrink(W^e x + b^e))$, which corresponds to an L_1 penalty on the activations. Thus this SATAE is equivalent to a sparse auto-encoder with a shrink activation function. Given the equivalence to the sparse auto-encoder we anticipate the same scale ambiguity which occurs with L_1 regularization. This ambiguity

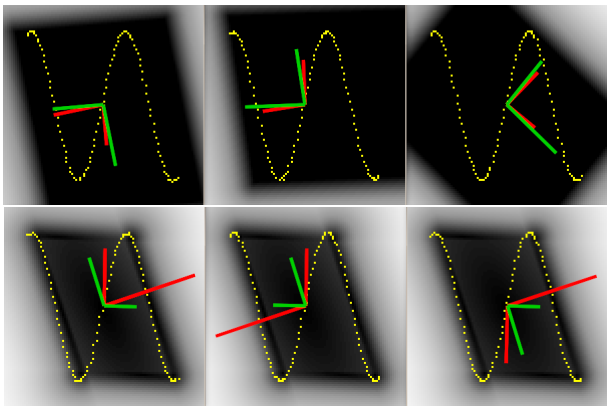


Figure 3.3: SATAE-SL toy example with two basis elements. Top Row: three randomly initialized solutions obtained with no regularization. Bottom Row: three randomly initialized solutions obtained with regularization.

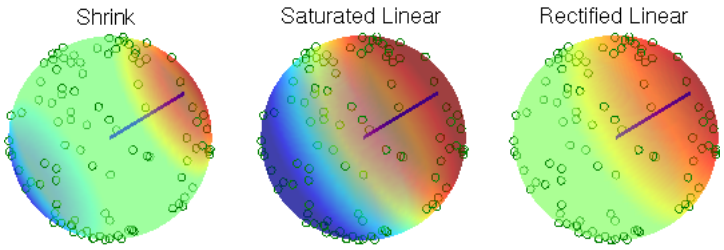


Figure 3.4: Geometric visualization of non-linearities

can be avoided by normalizing the decoder weights to unit norm. It is expected that the SATAE-shrink will learn similar features to those obtained with a sparse auto-encoder, and indeed this is what we observe. Figure 3.6(c) shows the decoder filters learned by an auto-encoder with shrink nonlinearity trained on gray-scale natural image patches. One can recognize the expected Gabor-like features when the saturation penalty is activated. When trained on the binary MNIST dataset the learned basis is comprised of portions of digits and strokes. Nearly identical results are obtained with a SATAE which uses a rectified-linear activation function. This is because a rectified-linear function with an encoding bias behaves as a positive only shrink function, similarly the complementary function is equivalent to a positive only L_1 penalty on the activations.

Figure 3.5: Evolution of two filters with increasing saturation regularization for a SATAE-SL trained on CIFAR-10. Filters corresponding to larger values of α were initialized using the filter corresponding to the previous α. The regularization parameter was varied from 0.1 to 0.5 (left to right) in the top five images and 0.5 to 1 in the bottom five

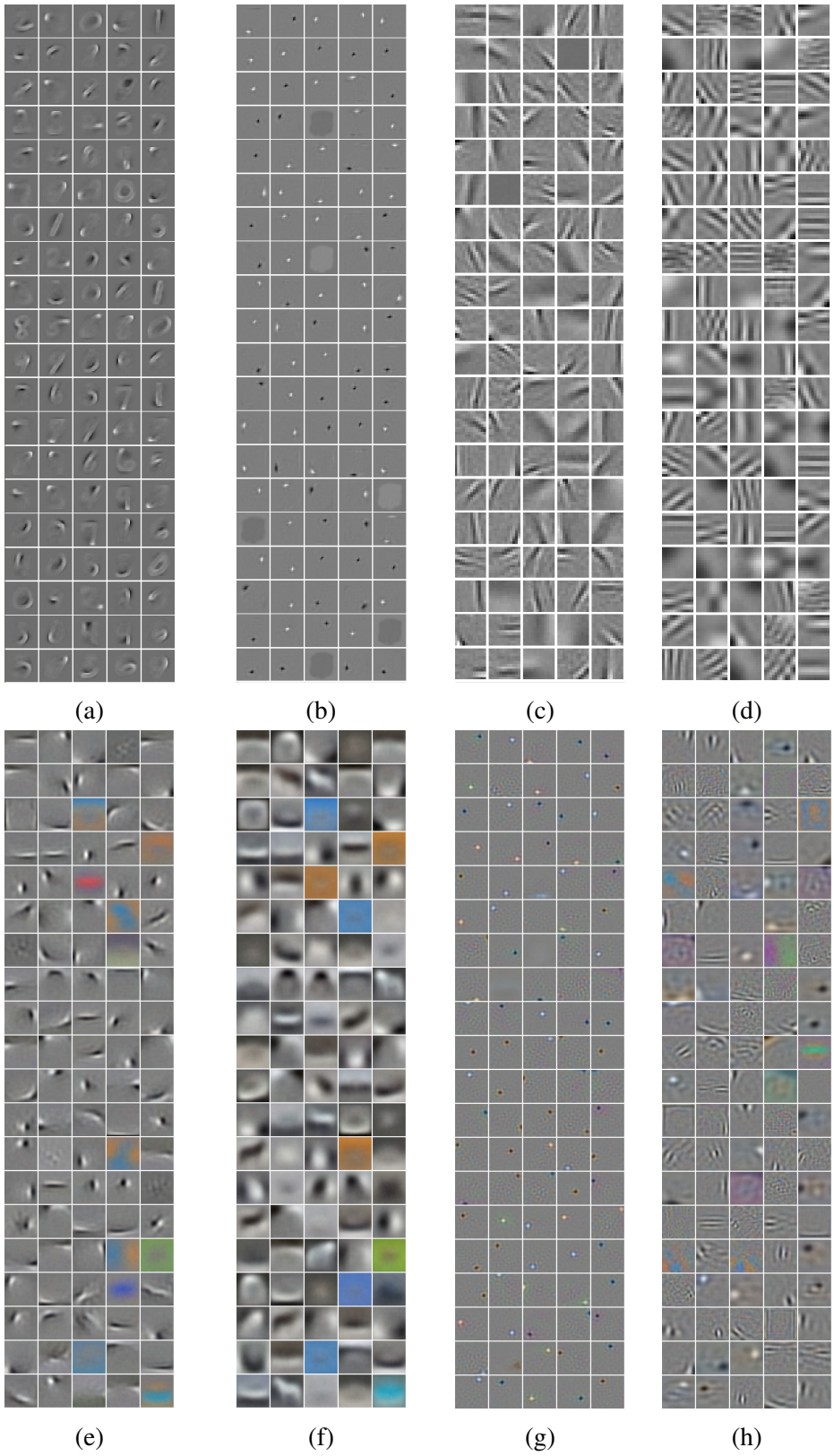


Figure 3.6: Basis elements learned by the SATAE using different nonlinearities on: 28x28 binary MNIST digits, 12x12 gray scale natural image patches, and CIFAR-10. (a) SATAE-shrink trained on MNIST, (b) SATAE-saturated-linear trained on MNIST, (c) SATAE-shrink trained on natural image patches, (d) SATAE-saturated-linear trained on natural image patches, (e)-(f) SATAE-shrink trained on CIFAR-10 with $\alpha = 0.1$ and $\alpha = 0.5$, respectively, (g)-(h) SATAE-SL trained on CIFAR-10 with $\alpha = 0.1$ and $\alpha = 0.6$, respectively.

3.3.3 SATAE-saturated-linear

Unlike the SATAE-shrink, which tries to compress the data by minimizing the number of active elements; the SATAE saturated-linear (SATAE-SL) tries to compress the data by encouraging the latent code to be as close to binary as possible. Without a saturation penalty this auto-encoder learns to encode small groups of neighboring pixels. More precisely, the auto-encoder learns the identity function on all datasets. An example of such a basis is shown in Figure 3.6(b). With this basis the auto-encoder can perfectly reconstruct any input by producing small activations which stay within the linear region of the nonlinearity. Introducing the saturation penalty does not have any effect when training on binary MNIST. This is because the scaled identity basis is a global minimizer of Equation 2 for the SATAE-SL on any binary dataset. Such a basis can perfectly reconstruct any binary input while operating exclusively in the saturated regions of the activation function, thus incurring no saturation penalty. On the other hand, introducing the saturation penalty when training on natural image patches induces the SATAE-SL to learn a more varied basis (Figure 3.6(d)).

3.3.4 Experiments on CIFAR-10

SATAE auto-encoders with 100 and 300 basis elements were trained on the CIFAR-10 dataset, which contains small color images of objects from ten categories. In all of our experiments the auto-encoders were trained by progressively increasing the saturation penalty (details are provided in the next section). This allowed us to visually track the effect of the saturation penalty on individual basis elements. Figure 3.6(e)-(f) shows the basis learned by SATAE-shrink with small and large saturation penalty, respectively. Increasing the saturation penalty has the expected effect of reducing the

number of nonzero activations. As the saturation penalty increases, active basis elements become responsible for reconstructing a larger portion of the input. This induces the basis elements to become less spatially localized. This effect can be seen by comparing corresponding filters in Figure 3.6(e) and (f). Figures 3.6(g)-(h) show the basis elements learned by SATAE-SL with small and large saturation penalty, respectively. The basis learned by SATAE-SL with a small saturation penalty resembles the identity basis, as expected (see previous subsection). Once the saturation penalty is increased small activations become more heavily penalized. To increase their activations the encoding basis elements may increase in magnitude or align themselves with the input. However, if the encoding and decoding weights are tied (or fixed in magnitude) then reconstruction error would increase if the weights were merely scaled up. Thus the basis elements are forced to align with the data in a way that also facilitates reconstruction. This effect is illustrated in Figure 3.5 where filters corresponding to progressively larger values of the regularization parameter are shown. The top half of the figure shows how an element from the identity basis ($\alpha = 0.1$) transforms to a localized edge ($\alpha = 0.5$). The bottom half of the figure shows how a localized edge ($\alpha = 0.5$) progressively transforms to a template of a horse ($\alpha = 1$).

3.4 Experimental Details

Because the regularizer explicitly encourages activations in the zero gradient regime of the nonlinearity, many encoder basis elements would not be updated via backpropagation through the nonlinearity if the saturation penalty were large. In order to allow the basis elements to deviate from their initial random states we found it necessary to progressively increase the saturation penalty. In our experiments the weights

obtained at a minimum of Equation 2 for a smaller value of α were used to initialize the optimization for a larger value of α. Typically, the optimization began with $\alpha = 0$ and was progressively increased to $\alpha = 1$ in steps of 0.1. The auto-encoder was trained for 30 epochs at each value of α. This approach also allowed us to track the evolution of basis elements as a function of α (Figure 3.5). In all experiments data samples were normalized by subtracting the mean and dividing by the standard deviation of the dataset. The auto-encoders used to obtain the results shown in Figure 3.6 (a),(c)-(f) used 100 basis elements, others used 300 basis elements. Increasing the number of elements in the basis did not have a strong qualitative effect except to make the features represented by the basis more localized. The decoder basis elements of the SATAEs with shrink and rectified-linear nonlinearities were reprojected to the unit sphere after every 10 stochastic gradient updates. The SATAEs which used saturated-linear activation function were trained with tied weights. All results presented were obtained using stochastic gradient descent with a constant learning rate of 0.05.

3.5 Discussion

In this work we have introduced a general and conceptually simple latent state regularizer. It was demonstrated that a variety of feature sets can be obtained using a single framework. The utility of these features depend on the application. In this section we extend the definition of the saturation regularizer to include functions without a zero-gradient region. The relationship of SATAEs with other regularized auto-encoders will be discussed. We conclude with a discussion on future work.

3.5.1 Extension to Differentiable Functions

We would like to extend the saturation penalty definition (Equation 1) to differentiable functions without a zero-gradient region. An appealing first guess for the complimentary function is some positive function of the first derivative, $f_c(x) = |f'(x)|$ for instance. This may be an appropriate choice for monotonic activation functions which have their lowest gradient regions at the extrema (e.g. sigmoids). However some activation functions may contain regions of small or zero gradient which have negligible extent, at the extrema for instance. We would like our definition of the complimentary function to not only measure the local gradient in some region, but to also measure it's extent. For this purpose we employ the concept of average variation over a finite interval. We define the average variation of f at x in the positive and negative directions at scale l, respectively as:

$$\begin{aligned}\Delta_l^+ f(x) &= \frac{1}{l}\int_x^{x+l} |f'(u)|du = |f'(x)| * \Pi_l^+(x) \\ \Delta_l^- f(x) &= \frac{1}{l}\int_{x-l}^{x} |f'(u)|du = |f'(x)| * \Pi_l^-(x).\end{aligned}$$

Where $*$ denotes the continuous convolution operator. $\Pi_l^+(x)$ and $\Pi_l^-(x)$ are uniform averaging kernels in the positive and negative directions, respectively. Next, define a directional measure of variation of f by integrating the average variation at all scales.

$$\begin{aligned}M^+ f(x) &= \int_0^{+\infty} \Delta_l^+ f(x) w(l) dl = \left[\int_0^{+\infty} w(l)\Pi_l^+(x)dl\right] * |f'(x)| \\ M^- f(x) &= \int_0^{+\infty} \Delta_l^- f(x) w(l) dl = \left[\int_0^{+\infty} w(l)\Pi_l^-(x)dl\right] * |f'(x)|.\end{aligned}$$

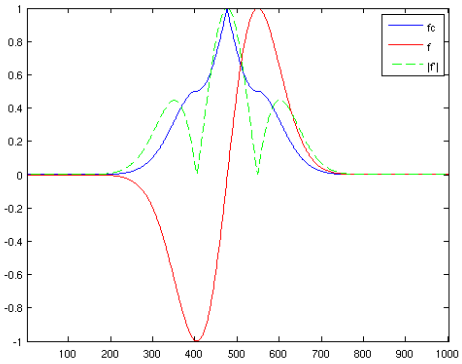


Figure 3.7: Illustration of the complimentary function (f_c) as defined by Equation 3 for a non-monotonic activation function (f). The absolute derivative of f is shown for comparison.

Where $w(l)$ is chosen to be a sufficiently fast decreasing function of l to insure convergence of the integral. The integral with which $|f'(x)|$ is convolved in the above equation evaluates to some decreasing function of x for Π^+ with support $x \geq 0$. Similarly, the integral involving Π^- evaluates to some increasing function of x with support $x \leq 0$. This function will depend on $w(l)$. The functions $M^+f(x)$ and $M^-f(x)$ measure the average variation of $f(x)$ at all scales l in the positive and negative direction, respectively. We define the complimentary function $f_c(x)$ as:

$$f_c(x) = min(M^+f(x), M^-f(x)). \tag{3.3}$$

An example of a complimentary function defined using the above formulation is shown in Figure 3.7. Whereas $|f'(x)|$ is minimized at the extrema of f, the complimentary function only plateaus at these locations.

3.5.2 Relationship with the Contractive Auto-Encoder

Let h_i be the output of the i^{th} hidden unit of a single-layer auto-encoder with point-wise nonlinearity $f(\cdot)$. The regularizer imposed by the contractive auto-encoder (CAE) can be expressed as follows:

$$\sum_{ij}\left(\frac{\partial h_i}{\partial x_j}\right)^2 = \sum_{i}^{d_h}\left(f'(\sum_{j=1}^{d} W_{ij}^e x_j + b_i)^2 \|W_i^e\|^2\right),$$

where x is a d-dimensional data vector, $f'(\cdot)$ is the derivative of $f(\cdot)$, b_i is the bias of the i^{th} encoding unit, and W_i^e denotes the i^{th} row of the encoding weight matrix. The first term in the above equation tries to adjust the weights so as to push the activations into the low gradient (saturation) regime of the nonlinearity, but is only defined for differentiable activation functions. Therefore the CAE indirectly encourages operation in the saturation regime. Computing the Jacobian, however, can be cumbersome for deep networks. Furthermore, the complexity of computing the Jacobian is $O(d \times d_h)$, although a more efficient implementation is possible [24], compared to the $O(d_h)$ for the saturation penalty.

3.5.3 Relationship with the Sparse Auto-Encoder

In Section 3.2 it was shown that SATAEs with shrink or rectified-linear activation functions are equivalent to a sparse auto-encoder. Interestingly, the fact that the saturation penalty happens to correspond to L_1 regularization in the case of SATAE-shrink agrees with the findings in [10]. In their efforts to find an architecture to approximate inference in sparse coding, Gregor et al. found that the shrink function is particularly compatible with L_1 minimization. Equivalence to sparsity only for some activation functions suggests that SATAEs are a generalization of sparse auto-encoders. Like the spar-

sity penalty, the saturation penalty can be applied at any point in a deep network for the same computational cost. However, unlike the sparsity penalty the saturation penalty is adapted to the nonlinearity of the particular layer to which it is applied.

Chapter 4

Convolutional Sparse Inference

Chapter 5

Learning Spatiotemporally Coherent Metrics

5.1 Introduction

Is it possible to characterize "good" representations without specifying a task a priori? If so, does there exist a set of generic priors which lead to these representations? In recent years state-of-the-art results from supervised learning suggest that the most powerful representations for solving specific tasks can be learned from the data itself. It has been hypothesized that large collections of unprocessed and unlabeled data can be used to learn generically useful representations. However the principles which would lead to these representations in the realm of unsupervised learning remain elusive. Temporal coherence is a form of weak supervision, which we exploit to learn generic signal representations that are stable with respect to the variability in natural video, including local deformations.

Our main assumption is that data samples that are temporal neighbors are also likely

to be neighbors in the latent space. For example, adjacent frames in a video sequence are more likely to be semantically similar than non-adjacent frames. This assumption naturally leads to the slowness prior on features which was introduced in SFA ([26]).

This prior has been successfully applied to metric learning, as a regularizer in supervised learning, and in unsupervised learning ([11, 20, 26]). A popular assumption in unsupervised learning is that high dimensional data lies on a low dimensional manifold parametrized by the latent variables as in [2, 24, 25, 9]. In this case, temporal sequences can be thought of as one-dimensional trajectories on this manifold. Thus, an ensemble of sequences that pass through a common data sample have the potential to reveal the local latent variable structure within a neighborhood of that sample.

Non-linear operators consisting of a redundant linear transformation followed by a point-wise nonlinearity and a local pooling, are fundamental building blocks in deep convolutional networks. This is due to their capacity to generate local invariance while preserving discriminative information ([18, 4]). We justify that pooling operators are a natural choice for our unsupervised learning architecture since they induce invariance to local deformations. The resulting pooling auto-encoder model captures the main source of variability in natural video sequences, which can be further exploited by enforcing a convolutional structure. Experiments on YouTube data show that one can learn pooling representations with good discrimination and stability to observed temporal variability. We show that these features represent a metric which we evaluate on retrieval and classification tasks.

5.2 Contributions and Prior Work

The problem of learning temporally stable representations has been extensively studied in the literature, most prominently in Slow Feature Analysis (SFA) and Slow Subspace Analysis (SSA) ([26, 15, 13]). Works that learn slow features distinguish themselves mainly in three ways: (1) how the features are parametrized, (2) how the trivial (constant) solution is avoided, and (3) whether or not additional priors such as independence or sparsity are imposed on the learned features.

The features presented in SFA take the form of a nonlinear transformation of the input, specifically a quadratic expansion followed by a linear combination using learned weights optimized for slowness ([26]). This parametrization is equivalent to projecting onto a learned basis followed by L_2 pooling. The recent work by [19] uses features which are composed of projection onto a learned unitary basis followed by a local L_2 pooling in groups of two.

Slow feature learning methods also differ in the way that they avoid the trivial solution of learning to extract constant features. Constant features are perfectly slow (invariant), however they are not informative (discriminative) with respect to the input. All slow feature learning methods must make a trade-off between the discriminability and stability of the learned features in order to avoid trivial solutions. Slow Feature Analysis introduces two additional constraints, namely that the learned features must have unit variance and must be decorrelated from one another. In the work by [19], the linear part of the transformation into feature space is constrained to be unitary. Enforcing that the transform be unitary implies that it is invertible *for all inputs*, and not just the data samples. This unnecessarily limits the invariance properties of the transform and precludes the possibility of learning over-complete bases. Since the pooling operation

following this linear transform has no trainable parameters, including this constraint is sufficient to avoid the trivial solution. Metric learning approaches ([11]) can be used to perform dimensionality reduction by optimizing a criteria which minimizes the distance between temporally adjacent samples in the transformed space, while repelling non-adjacent samples with a hinge loss, as explained in Section 5.3. The margin based contrastive term in DrLIM is explicitly designed to only avoid the constant solution and provides no guarantee on how informative the learned features are. Furthermore since distances grow exponentially due to the curse of dimensionality, metric based contrastive terms can be trivially satisfied in high dimensions.

Our approach uses a reconstruction criterion as a contrastive term. This approach is most similar to the one taken by [14] when optimizing group sparsity. In this work group-sparsity is replaced by slowness, and multiple layers of convolutional slow features are trained.

Several other studies combine the slowness prior with independence inducing priors [19, 6, 27]. For a detailed discussion on the connection between independence and sparsity see [12]. However, our model maximizes the sparsity of the representation *before* the pooling operator. Our model can be interpreted as a sparse auto-encoder additionally regularized by slowness through a local pooling operator.

In this work we introduce the use of convolutional pooling architectures for slow feature learning. At small spatial scales, local translations comprise the dominant source of variability in natural video; this is why many previous works on slowness learn mainly locally translation-invariant features ([26, 15, 19]). However, convolutional pooling architectures are locally translation-invariant by design, which allows our model to learn features that capture a richer class of invariances, beyond translation. Finally, we demonstrate that nontrivial convolutional dictionaries can be learned in the unsu-

(a) (b)

Figure 5.1: (a) Three samples from our rotating plane toy dataset. (b) Scatter plot of the dataset plotted in the output space of G_W at the start (top) and end (bottom) of training. The left side of the figure is colored by the yaw angle, and the right side by roll, 0° blue, 90° in pink.

pervised setting using only stochastic gradient descent (on mini-batches), despite their huge redundancy — that is, without resorting to alternating descent methods or iterative sparse inference algorithms.

5.3 Slowness as Metric Learning

coherence can be exploited by assuming a prior on the features extracted from the temporal data sequence. One such prior is that the features should vary slowly with respect to time. In the discrete time setting this prior corresponds to minimizing an L^p norm of the difference of feature vectors for temporally adjacent inputs. Consider a video sequence with T frames, if z_t represents the feature vector extracted from the frame at time t then the slowness prior corresponds to minimizing $\sum_{t=1}^{T} \|z_t - z_{t-1}\|_p$. To avoid the degenerate solution $z_t = z_0$ for $t = 1...T$, a second term is introduced which encourages data samples that are *not* temporal neighbors to be separated by at least a distance of m-units in feature space, where m is known as the margin. In the temporal setting this corresponds to minimizing $max(0, m - \|z_t - z_{t'}\|_p)$, where $|t - t'| > 1$. Together the two terms form the loss function introduced in [11] as a dimension

reduction and data visualization algorithm known as DrLIM. Assume that there is a differentiable mapping from input space to feature space which operates on *individual* temporal samples. Denote this mapping by G and assume it is parametrized by a set of trainable coefficients denoted by W. That is, $z_t = G_W(x_t)$. The per-sample loss function can be written as:

$$L(x_t, x_{t'}, W) = \begin{cases} \|G_W(x_t) - G_W(x_{t'})\|_p, & \text{if } |t - t'| = 1 \\ \max(0, m - \|G_W(x_t) - G_W(x_{t'})\|_p) & \text{if } |t - t'| > 1 \end{cases} \tag{5.1}$$

In practice the above loss is minimized by constructing a "Siamese" network ([3]) with shared weights whose inputs are pairs of samples along with their temporal indices. The loss is minimized with respect to the trainable parameters with stochastic gradient descent via back-propagation. To demonstrate the effect of minimizing Equation 5.1 on temporally coherent data, consider a toy data-set consisting of only one object. The data-set is generated by rotating a 3D model of a toy plane (Figure 5.1a) by 90° in one-degree increments around two-axes of rotation, generating a total of 8100 data samples. Input images (96×96) are projected into two-dimensional output space by the mapping G_W. In this example the mapping $G_W(X) : \mathbb{R}^{9216} \rightarrow \mathbb{R}^2$. We chose G_W to be a fully connected two layer neural network. In effect this data-set lies on an intrinsically two-dimensional manifold parametrized by two rotation angles. Since the sequence was generated by continuously rotating the object, temporal neighbors correspond to images of the object in similar configurations. Figure 5.1b shows the data-set plotted in the output space of G_W at the start (top row) and end (bottom row) of training. The left and right hand sides of Figure 5.1b are colorized by the two rotational angles, which are never explicitly presented to the network. This result implies that G_W has learned a mapping in which the latent variables (rotation angles) are linearized. Furthermore, the

gradients corresponding to the two rotation angles are nearly orthogonal in the output space, which implies that the two features extracted by G_W are independent.

5.4 Slow Feature Pooling Auto-Encoders

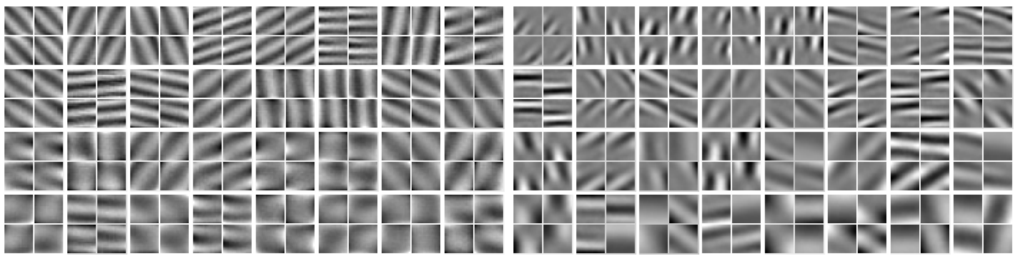
The second contrastive term in Equation 5.1 only acts to avoid the degenerate solution in which G_W is a constant mapping, it does not guarantee that the resulting feature space is informative with respect to the input. This discriminative criteria only depends on pairwise distances in the representation space which is a geometrically weak notion in high dimensions. We propose to replace this contrastive term with a term that penalizes the reconstruction error of both data samples. Introducing a reconstruction terms not only prevents the constant solution but also acts to explicitly preserve information about the input. This is a useful property of features which are obtained using unsupervised learning; since the task to which these features will be applied is not known a priori, we would like to preserve as much information about the input as possible.

What is the optimal architecture of G_W for extracting slow features? Slow features are invariant to temporal changes by definition. In natural video and on small spatial scales these changes mainly correspond to local translations and deformations. Invariances to such changes can be achieved using appropriate pooling operators [4, 18]. Such operators are at the heart of deep convolutional networks (ConvNets), currently the most successful supervised feature learning architectures [17]. Inspired by these observations, let G_{W_e} be a two stage encoder comprised of a learned, generally over-complete, linear map (W_e) and rectifying nonlinearity $f(\cdot)$, followed by a local pooling. Let the N hidden activations, $h = f(W_e x)$, be subdivided into K potentially overlapping neighborhoods denoted by P_i. Note that biases are absorbed by expressing the input x in homogeneous

coordinates. Feature z_i produced by the encoder for the input at time t can be expressed as $G^i_{W_e}(t) = \|h_t\|_p^{P_i} = \left(\sum_{j \in P_i} h_{tj}^p\right)^{\frac{1}{p}}$. Training through a local pooling operator enforces a local topology on the hidden activations, inducing units that are pooled together to learn complimentary features. In the following experiments we will use $p = 2$. Although it has recently been shown that it is possible to recover the input when W_e is sufficiently redundant, reconstructing from these coefficients corresponds to solving a phase recovery problem [5] which is not possible with a simple inverse mapping, such as a linear map W_d. Instead of reconstructing from z we reconstruct from the hidden representation h. This is the same approach taken when training group-sparse auto-encoders [14]. In order to promote sparse activations in the case of over-complete bases we additionally add a sparsifying L_1 penalty on the hidden activations. Including the rectifying nonlinearity becomes critical for learning sparse inference in a hugely redundant dictionary, e.g. convolutional dictionaries [10]. The complete loss functional is:

$$L(x_t, x_{t'}, W) = \sum_{\tau=\{t,t'\}} \left(\|W_d h_\tau - x_\tau\|^2 + \alpha|h_\tau|\right) + \beta \sum_{i=1}^{K} \left|\|h_t\|^{P_i} - \|h_{t'}\|^{P_i}\right| \tag{5.2}$$

Figure 5.3 shows a convolutional version of the proposed architecture and loss. This combination of loss and architecture can be interpreted as follows: the sparsity penalty induces the first stage of the encoder, $h = f(W_e x)$, to approximately infer sparse codes in the analysis dictionary W_e; the slowness penalty induces the formation of pool groups whose output is stable with respect to temporal deformations. In other words, the first stage partitions the input space into disjoint linear subspaces and the second stage recombines these partitions into temporally stable groups. This can be seen as a sparse auto-encoder whose pooled codes are additionally regularized by slowness.



(a) (b)

Figure 5.2: Pooled decoder dictionaries learned without (a) and with (b) the L_1 penalty using (5.2).

5.4.1 Fully-Connected Architecture

To gain an intuition for the properties of the minima of Equation 5.2 for natural data, an auto-encoder was trained on a small dataset consisting of natural movie patches. This data set consists of approximately 170,000, 20×20 gray scale patches extracted from full resolution movies. Minimizing Equation 5.2 with $\alpha = 0$ results in the learned decoder basis shown in Figure 5.2a. Here a dictionary of 512 basis elements was trained whose outputs were pooled in non-overlapping groups of four resulting in 128 output features. Only the slowest 32 groups are shown in Figure 5.2a. The learned dictionary has a strong resemblance to the two-dimensional Fourier basis, where most groups are comprised of phase shifted versions of the same spatial frequency. Since translations are an invariant of the local modulus of the Fourier transform, the result of this experiment is indicative of the fact that translations are the principal source of variation at small spatial scales. Minimizing Equation 5.2 with $\alpha > 0$ results in a more localized basis depicted in Figure 5.2b. This basis is more consistent with a local deformation model as opposed to a global one.

5.4.2 Convolutional Architecture

By replacing all linear operators in our model with convolutional filter banks and including spatial pooling, translation invariance need not be learned [18]. In all other respects the convolutional model is conceptually identical to the fully connected model described in the previous section. One important difference between fully-connected and convolutional dictionaries is that the later can be massively over-complete, making sparse inference potentially more challenging. Nevertheless we found that non-trivial dictionaries (see Figure 5.5d) can be learned using purely stochastic optimization, that is, without a separate sparse inference phase. Let the linear stage of the encoder consist of a filter bank which takes C input feature maps (corresponding to the 3 color channels for the first stage) and produces D output feature maps. Correspondingly, the convolutional decoder transforms these D feature maps back to C color channels. In the convolutional setting slowness is measured by subtracting corresponding spatial locations in temporally adjacent feature maps. In order to produce slow features a convolutional network must compensate for the motion in the video sequence by producing *spatially* aligned activations for *temporally* adjacent samples. In other words, in order to produce slow features the network must implicitly learn to track common patterns by learning features which are invariant to the deformations exhibited by these patterns in the temporal sequence. The primary mechanism for producing these invariances is pooling in space and across features [8]. Spatial pooling induces local translation invariance. Pooling across feature maps allows the network to potentially learn feature groups that are stable with respect to more general deformations. Intuitively, maximizing slowness in a convolutional architecture leads to *spatiotemporally* coherent features.

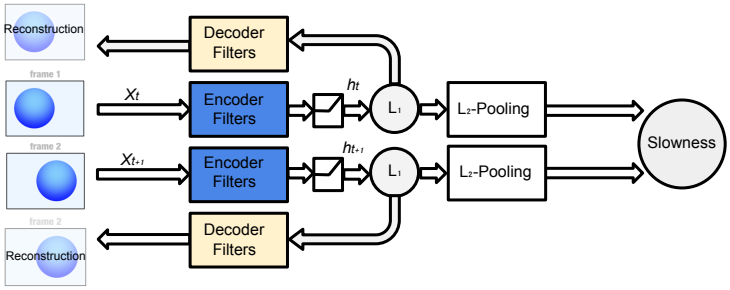


Figure 5.3: Block diagram of the Siamese convolutional model trained on pairs of frames.

5.5 Experimental Results

To verify the connection between slowness and metric learning, we evaluate the metric properties of the learned features. It is well known that distance in the extrinsic (input pixel) space is not a reliable measure of semantic similarity. Maximizing slowness corresponds to minimizing the distance between adjacent frames in code space, therefore neighbors in code space should correspond to temporal neighbors. This claim can be tested by computing the nearest neighbors to a query frame in code space, and verifying whether they correspond to its temporal neighbors. However, the features must also be discriminative so as not to collapse temporally distant samples. In order to make this trade-off in a principled manner, a dataset comprised of short natural scenes was collected. Hyper-parameters are selected which maximize the so called "temporal coherence" of the features which define the metric. We define the temporal coherence of a metric $G_W(\cdot)$ as the area under the precision-recall curve, where precision is defined as the proportion of the nearest neighbors that come from the same scene, and recall is defined as the proportion of frames recalled from that scene. In our experiments, we used the middle frame from each scene as the query.

However, temporal coherence can be a very weak measure of discriminability; it

merely requires that scenes be easy to disambiguate in feature space. If the scenes are quite distinct, then maximizing temporal coherence directly can lead to weakly discriminative features (e.g. color histograms can exhibit good temporal coherence). We therefore evaluate the learned features on a more demanding task by assessing how well the metric learned from the YouTube dataset transfers to a classification task on the CIFAR-10 dataset. Average class-based precision is measured in feature space by using the test set as the query images and finding nearest neighbors in the training set. Precision is defined as the proportion of nearest neighbors that have the same label. As on the YouTube dataset we evaluate the average precision for the nearest 40 neighbors. The CIFAR dataset contains considerably more interclass variability than the scenes in our YouTube dataset, nevertheless many class instances are visually similar.

approximately $150,000$ frames extracted from YouTube videos. Of these, approximately $20,000$ frames were held out for testing. The training and test set frames were collected from separate videos. The videos were automatically segmented into scenes of variable length (2-40 frames) by detecting large L_2 changes between adjacent frames. Each color frame was down-sampled to a 32×32 spatial resolution and the entire dataset was ZCA whitened [16]. Six scenes from the test set are shown in Figure 5.4 where the first scene (top row) is incorrectly segmented.

We compare the features learned by minimizing the loss in Equation 5.2 with the features learned by minimizing DrLIM (Equation 5.1) and group sparsity (Equation 5.3) losses. Once trained, the convolution, rectification, and pooling stages are used to transform the dataset into the feature space. We use cosine distance in feature space to determine the nearest neighbors and select hyperparmeters for each method which maximize the temporal coherence measure.

We trained two layers of our model using greedy layer-wise training [2]. The first

layer model contains a filter bank consisting of 64 kernels with 9×9 spatial support. The first L_2 pooling layer computes the local modulus volumetrically, that is *across* feature maps in non-overlapping groups of four and spatially in 2×2 non-overlapping neighborhoods. Thus the output feature vector of the first stage (z_1) has dimensions $16 \times 16 \times 16$ (4096). Our second stage consists of 64 5×5 convolutional filters, and performs 4×4 spatial pooling producing a second layer code (z_2) of dimension $64 \times 4 \times 4$ (1024). The output of the second stage corresponds to a dimension reduction by a factor of three relative to the input dimension.

Identical one and two-layer architectures were trained using the group sparsity prior, similar to [14]. As in the slowness model, the two layer architecture was trained greedily. Using the same notation as Equation 5.2, the corresponding loss can be written as:

$$L(x_t, W) = \sum_{\tau} \|W_d h_\tau - x_\tau\|^2 + \alpha \|h_\tau\|^{P_i} \tag{5.3}$$

Finally, identical one and two-layer architectures were also trained by minimizing the DrLIM loss in Equation 5.1. Negative pairs, corresponding to temporally non-adjacent frames, were independently selected at random. In order to achieve the best temporal precision-recall performance, we found that each mini-batch should consist of a large proportion of negative to positive samples (at least five-to-one). Unlike the auto-encoder methods, the two layer architecture was trained jointly rather than greedily.

results on the YouTube dataset for a single frame (left column) in eight spaces. The top row shows the nearest neighbors in pixel space. The second row shows the nearest neighbors in pixel space after ZCA whitening. The next six rows show the nearest neighbors in feature space for one and two layer feature transformations learned with slowness, group sparsity, and DrLIM. The resulting first-layer filters and precision-recall

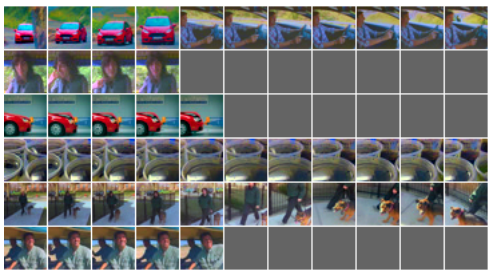
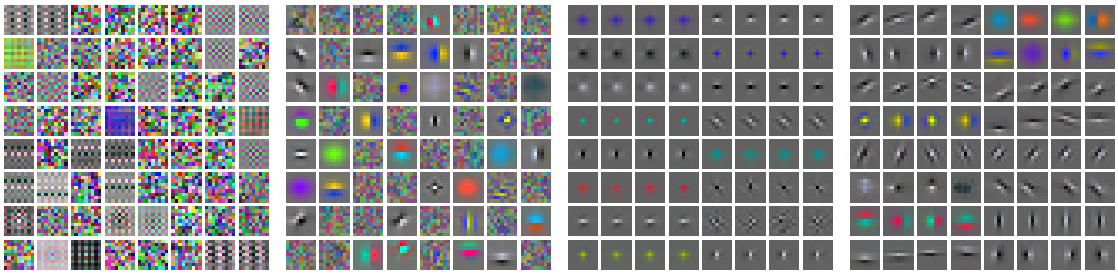


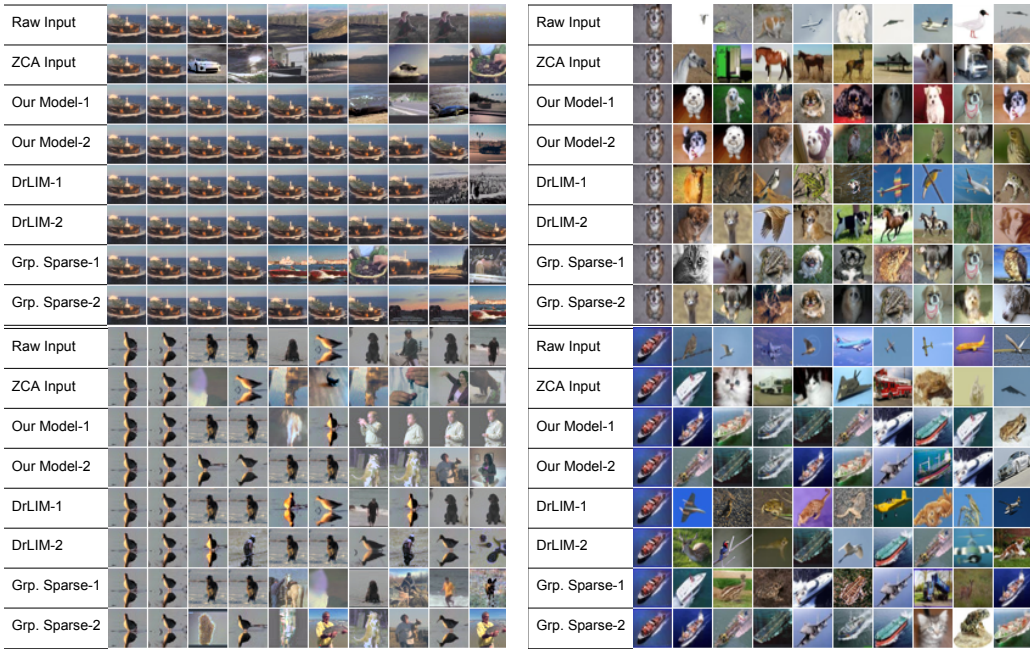
Figure 5.4: Six scenes from our YouTube dataset



(a) (b) (c) (d)

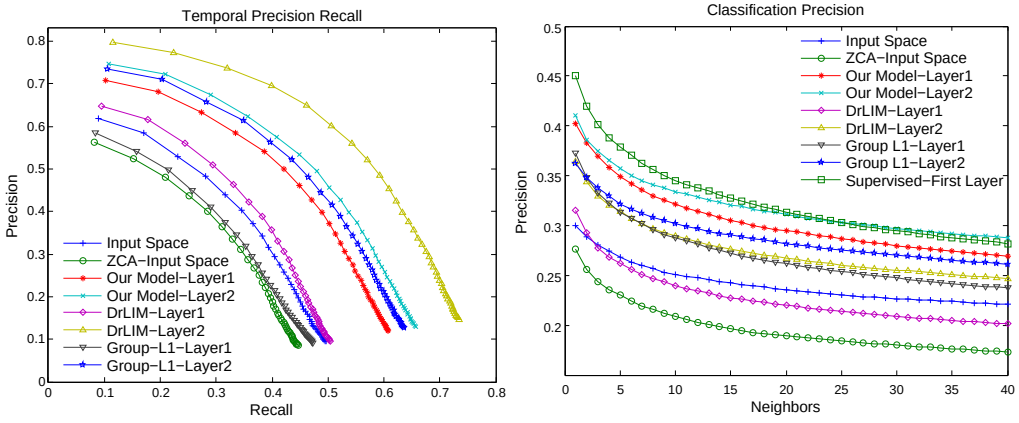
Figure 5.5: Pooled convolutional dictionaries (decoders) learned with: (a) DrLIM and (b) sparsity only, (c) group sparsity, and (d) sparsity and slowness. Groups of four features that were pooled together are depicted as horizontally adjacent filters.

curves are shown in Figures 5.5 and 5.7, respectively. Figures 5.5b and 5.5d show the decoders of two one-layer models trained with $\beta = 0, 2$, respectively, and a constant value of α. The filter bank trained with $\beta = 0$ exhibits no coherence within each pool group; the filters are not visually similar nor do they tend to co-activate at spatially neighboring locations. Most groups in the filter bank trained with slowness tend be visually similar, corresponding to similar colors and/or geometric structures. The features learned by minimizing the DrLIM loss (Equation 5.1), which more directly optimizes temporal coherence, have much more high frequency content than the filters learned with any of the auto-encoder methods. Nevertheless, some filters within the same pool group exhibit similar geometric and color structure (Figure 5.5a). The features learned with a group-sparsity regularizer leads to nearly identical features (and nearly identical



(a) (b)

Figure 5.6: Query results in the (a) video and (b) CIFAR-10 datasets. Each row corresponds to a different feature space in which the queries were performed; numbers (1 or 2) denote the number of convolution-pooling layers.



(a) (b)

Figure 5.7: Precision-Recall curves corresponding to the YouTube (a) and CIFAR-10 (b) dataset.

Model	Optimization	Temporal AUC	Class AUC
Our Model Layer1	—	0.262	0.296
Our Model Layer2	Greedy	0.300	**0.310**
DrLIM Layer1	—	0.188	0.221
DrLIM Layer2	Joint	**0.378**	0.268
Group L_1 Layer1	—	0.231	0.266
Group L_1 Layer2	Greedy	0.285	0.281

activations) within each pool group (Figure 5.5c). This is not surprising because group sparsity promotes co-activation of the features within each pool group, by definition. We have also tried including an individual sparsity prior, as in Equation 5.2, in order to encourage independence among the pooled features. However this has lead to significantly worse temporal-coherence performance.

Figure 5.6b shows the result of two queries in the CIFAR-10 dataset. The corresponding precision-recall curves are shown in Figure 5.7b. One-layer DrLIM (4096 dimensional) exhibit poor performance in both the temporal and class-based recall tasks. In contrast, jointly trained two-layer DrLIM features (1024 dimensional) exhibit excellent temporal coherence, outperforming all other models by a large margin. Although better than the first layer, second layer features perform significantly worse on the CIFAR task than even the first-layer features learned by our model. Furthermore, the nearest neighbors in both the one and two-layer feature spaces learned with DrLIM are often neither visually nor semantically similar (see Figure 5.6b). The conclusion which can be drawn from this result is that ***directly maximizing temporal coherence alone is not a sufficient condition for achieving a semantically (or even visually) coherent features***. However, combining it with reconstruction and sparsity, as in our model, yields the most semantically discriminative features. Although significantly better than the features learned with DrLIM, the features learned with group sparsity exhibit slightly weaker temporal coherence and significantly worse class-based recall. Note that since

all the features within a pool group are practically identical, the invariants captured by the pool groups are limited to local translations due to the spatial pooling. As a final comparison, we trained a four layer ConvNet with supervision on CIFAR-10, this network achieved approximately 80% classification accuracy on the test set. The architecture of the first two stages of the ConvNet is identical to the architecture of the first and second unsupervised stages. The precision curve corresponding to the first layer of the ConvNet is shown in Figure 5.7b, which is matched by our-model's second layer at high recall.

5.6 Conclusion

Video data provides a virtually infinite source of information to learn meaningful and complex visual invariances. While temporal slowness is an attractive prior for good visual features, in practice it involves optimizing conflicting objectives that balance invariance and discriminability. In other words, perfectly slow features cannot be informative. An alternative is to replace the small temporal velocity prior with small temporal acceleration, leading to a criteria that *linearizes* observed variability. The resulting representation offers potential advantages, such as extraction of both locally invariant and locally covariant features. Although pooling representations are widespread in visual and audio recognition architectures, much is left to be understood. In particular, a major question is how to learn a stacked pooling representation, such that its invariance properties are boosted while controlling the amount of information lost at each layer. This could be possible by replacing the linear decoder of the proposed model with a non-linear decoder which can be used to reconstruct the input from pooled representations. Slow feature learning is merely one way to learn from temporally coherent data. In

this work we have provided an auto-encoder formulation of the problem and shown that the resulting features are more stable to naturally occurring temporal variability, while maintaining discriminative power.

Chapter 6

Learning to Linearize under Uncertainty

Chapter 7

Adversarial Inpainting

Chapter 8

Conclusion

Bibliography

[1] Guillaume Alain and Yoshua Bengio. "What regularized auto-encoders learn from the data-generating distribution". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3563–3593.

[2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. *Representation Learning: A Review and New Perspectives*. Tech. rep. University of Montreal, 2012.

[3] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. "Signature verification using a Siamese time delay neural network". In: *International Journal of Pattern Recognition and Artificial Intelligence* 7.04 (1993), pp. 669–688.

[4] Joan Bruna and Stéphane Mallat. "Invariant scattering convolution networks". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.8 (2013), pp. 1872–1886.

[5] Joan Bruna, Arthur Szlam, and Yann LeCun. "Signal Recovery from Pooling Representations". In: *ICML*. 2014.

[6] Charles F. Cadieu and Bruno A. Olshausen. "Learning Intermediate-Level Representations of Form and Motion from Natural Movies". In: *Neural Computation* (2012).

[7] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[8] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. “Maxout Networks”. In: *ICML*. 2013.

[9] Rostislav Goroshin and Yann LeCun. “Saturating Auto-Encoders”. In: *ICLR*. 2013.

[10] Karol Gregor and Yann LeCun. “Learning fast approximations of sparse coding”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 399–406.

[11] Raia Hadsell, Soumit Chopra, and Yann LeCun. “Dimensionality Reduction by Learning an Invariant Mapping”. In: *CVPR*. 2006.

[12] Hyvärinen, Aapo, Karhunen, Juha, Oja, and Erkki. *Independent component analysis*. Vol. 46. John Wiley & Sons, 2004.

[13] Aapo Hyvärinen, Jarmo Hurri, and Jaakko Väyrynen. “Bubbles: a unifying framework for low-level statistical properties of natural image sequences”. In: *JOSA A* 20.7 (2003), pp. 1237–1252.

[14] Koray Kavukcuoglu, MarcAurelio Ranzato, Rob Fergus, and Yann LeCun. “Learning Invariant Features through Topographic Filter Maps”. In: *CVPR*. 2009.

[15] Christoph Kayser, Wolfgang Einhauser, Olaf Dummer, Peter Konig, and Konrad Kding. “Extracting Slow Subspaces from Natural Videos Leads to Complex Cells”. In: *ICANN’2001*. 2001.

[16] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. MA thesis. University of Toronto, 2009.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*. Vol. 1. 2. 2012, p. 4.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-Based Learning Applied to Document Recognition”. In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324.

[19] Jorn-Philipp Lies, Ralf M Hafner, and Matthias Bethge. “Slowness and Sparseness Have Diverging Effects on Complex Cell Learning”. In: 10 (3 2014).

[20] Hossein Mobahi, Ronan Collobert, and Jason Weston. “Deep Learning from Temporal Coherence in Video”. In: *ICML*. 2009.

[21] Bruno A Olshausen and David J Field. “Sparse coding with an overcomplete basis set: A strategy employed by V1?” In: *Vision research* 37.23 (1997), pp. 3311–3325.

[22] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. “Efficient learning of sparse representations with an energy-based model”. In: *Advances in neural information processing systems*. 2006, pp. 1137–1144.

[23] Marc Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. “Unsupervised learning of invariant feature hierarchies with applications to object recognition”. In: *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[24] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. “Contractive auto-encoders: Explicit invariance during feature extraction”. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 833–840.

[25] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1096–1103.

[26] Laurenz Wiskott and Terrence J. Sejnowski. “Slow Feature Analysis: Unsupervised Learning of Invariances”. In: *Neural Computation* (2002).

[27] Will Zou, Shenghuo Zhu, Kai Yu, and Andrew Y Ng. “Deep learning of invariant features via simulated fixations in video”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 3212–3220.