Unsupervised Feature Learning from Temporal Data

Rostislav Goroshin[1] goroshin@cims.nyu.edu
Joan Bruna[1] bruna@cims.nyu.edu
Jonathan Tompson[1] tompson@cims.nyu.edu
David Eigen[1] deigen@cs.nyu.edu
Yann LeCun[1] yann@cs.nyu.edu

[2]Department of Mathematics, City College of New York
[1]Courant Institute of Mathematical Sciences, New York University

Abstract

Current state-of-the art object detection and recognition algorithms mainly use supervised training, and most benchmark datasets contain only static images. In this work we study feature learning in the context of temporally coherent video data. We focus on training convolutional features on unlabeled video data, using only the assumption that adjacent video frames contain semantically similar information. This assumption is exploited to train a pooling auto-encoder model regularized by slowness and sparsity. First, we confirm that fully connected networks mainly learn features stable under translation. Insipred by this observation, we proceed to train convolutional slow features which reveal richer invariants that are learned from natural video data.

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, unlabeled data can be used to learn generically useful representations. However the principles which would lead to generically useful representations in the realm of unsupervised learning remain elusive. Temporal coherence is a form of weak supervision, which we exploited to learn generic signal representations that are stable with respect to the variability in natural video, including local deformations. Such representations can then be used to improve deep convolutional architectures on supervised tasks.

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 [21]. This prior has been successfully applied to metric learning, as a regularizer in supervised learning, and in unsupervised learning [8, 15, 21]. A popular assumption in unsupervised learning is that high dimensional data lies on a low dimensional manifold parametrized by the latent variables [1, 18, 20, 6]. 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 non-linearity and a local pooling, are fundamental building blocks in deep convolutional networks, due to their capacity to generate local invariance while preserving discriminative information [13, 2]. By considering a simple generative model based on local translations, we justify that pooling operators are a natural choice for our unsupervised learning architecture. The resulting pooling autoencoder 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.

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) [21, 11]. Works that learn slow features distinguish themselves mainly in three ways: (1) how the features are parameterized, (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 [21]. This parametrization is equivalent to projecting onto a learned basis followed by L^2 pooling. The recent work by Lies et al [14] uses features which are composed of projection onto a learned unitary basis followed by a local L^2 pooling in groups of two [14].

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 Lies et. al, the linear part of the transformation into feature space is constrained to be unitary. However enforcing that the transform be unitary implies that it is invertible *for all inputs*, and not just the data samples which unnecessarily limits its invariance properties. 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 [8] 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 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 constrastive term. This apporach is most similar to the one taken by Kavukcuoglu et al [10] when optimizing group sparsity. However, in our approach the pooling is done over a sparse representation of the input which limits information loss through pooling.

Several other studies combine the slowness prior with independence inducing priors [14, 4]. For a detailed explanation between independence and sparsity see [9]. However, our model maximizes the sparsity of the representation *before* the pooling operator.

This work relates slowness to metric learning and demonstrates that the learned features represent a semantically meaningful metric, which is tested on retrieval and classification tasks. The dominant source of variability in natural videos on small spatial scales is mainly due to local translations. This is why many previous works on slowness learn mainly locally translation invariant features [21, 11, 14]. In this work we introduce the use of convolutional pooling architectures, which are locally translation invariant by design, and show that a richer variety of slow features can be learned.

3 Slowness as Metric Learning

Temporal 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

(a)

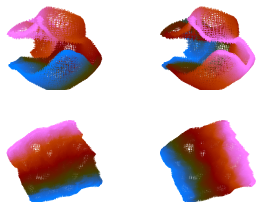

(b)

Figure 1: (a) Three samples from out rotating plane toy dataset. (b) Toy 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.

discrete time setting this prior corresponds to minimizing an L^p norm of the difference of feature vectors corresponding to 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 $z_t = 0 \; \forall \; t$ the second term in the objective 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 a loss function introduced by Hadsell et al as a dimension reduction and data visualization algorithm known as DrLIM [8]. 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:

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

In practice the above loss is minimized by constructing a "Siamese" network with shared weights whose inputs are pairs of samples along with their temporal indices [8]. The loss is minimized with respect to the trainable parameters with stochastic gradient descent via back-propagation. To demonstrate the effect of minimizing Equation 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 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 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 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.

4 Slow-Feature Auto-Encoders

The contrastive term in (1) only acts to avoid the degenerate solution in which G_W is a constant mapping. We propose to replace this contrastive term with a term that penalizes a reconstruction error of both data samples. Denote the functions which serve as the encoder and decoder by G_{W_e} and R_{W_d}, respectively. In the auto-encoder setting the decoder acts on the output of G_{W_e} and thus serves as the approximate inverse of G_{W_e} at each data sample. Let $X \in \mathbb{R}^m$ and $Z \in \mathbb{R}^n$, then $G_{W_e}(X) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $R_{W_d}(X) : \mathbb{R}^n \rightarrow \mathbb{R}^m$. 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

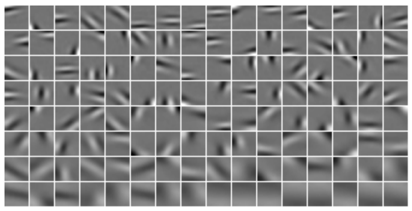


Figure 2: Visualization of the linear features obtained by minimizing Equation 2 on natural movies.

these features will be applied is not known a priori, we would like to preserve as much information about the input as possible. Replacing the contrastive term in (1) with the reconstruction objective and choosing the L^1 norm as the measure of slowness, we obtain:

$$L(x_t, x_{t'}, W) = \sum_{\tau=\{t,t'\}} \|R_{W_d}(G_{W_e}(x_\tau)) - x_\tau\| + \beta |G_{W_e}(x_t) - G_{W_e}(x_{t'})| \, , \tag{2}$$

where β is a hyper-parameter that determines the weight on the slowness of the features.

This architecture may be viewed as a regularized auto-encoder with its latent state regularized by temporal slowness [20, 18, 17]. Since the L^1 norm is known to induce sparsity [17], the choice of L^1 norm in (2) corresponds to an implicit sparse prior on the *difference* between the representations of two temporally adjacent data samples. The implicit prior is that only a small subset of latent variables change between temporally adjacent data samples. Note that since $|z_1 - z_2| \leq |z_1| + |z_2|$, the second term in (2) can be arbitrarily minimized if G simply scales down the input. The reconstruction need not suffer if R scales up the code produced by G. In our experiments $R_{W_d} = W_d$ is a linear map, and this problem can be avoided by normalizing the columns of W_d [17].

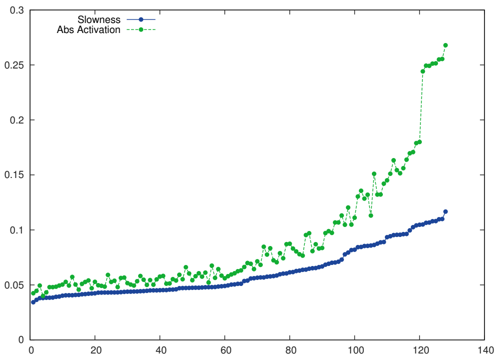
To explore the properties of the minima of (2) for natural data we present experiments on a data set consisting of natural movie patches. The data set consists of approximately 170,000, 20×20 gray scale patches extracted from full resolution movies. Consider the simple case where $G_{W_e} = W_e$ and $R_{W_d} = W_d$, i.e. the encoder and decoder are linear maps. Minimizing the cost over our data-set using stochastic gradient descent yields the features shown in Figure 2. The features are sorted by increasing average slowness (normalized by their average activation) from top left to bottom right. The slowness of the features is inversely proportional to their spatial frequency. This fact can be easily interpreted. Assuming a model of variability given by a family of transformations $\{U_\tau\}_\tau$, which may include local translations, rotations, etc, slow linear features w_i should satisfy

$$\forall \, x, \, \tau \, , \, \langle x, w_i \rangle \approx \langle U_\tau x, w_i \rangle = \langle x, U_\tau^* w_i \rangle \, ,$$

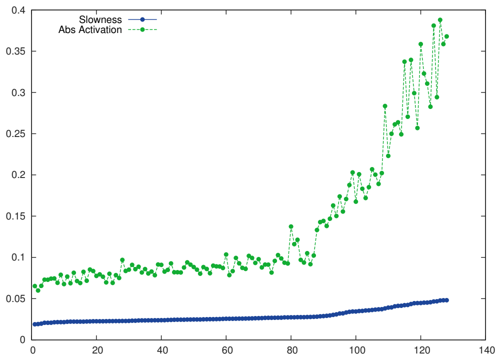
where U_τ^* is the adjoint operator of U_τ. It follows that slow linear features should be near eigenvectors of *any* transformation in $\{U_\tau\}_\tau$, which explains why low frequency information is the only linear slow feature representation. It is well known that the lowest frequencies carry most of the energy in natural images [19, 16]. As a consequence, the average activation of a particular feature will be inversely proportional to its frequency. Thus slowness is achieved in linear maps by simply reducing the activation of each feature as much as possible, while still reconstructing the input. Figure 3a shows a plot of the average slowness (blue) and average absolute activation (green) of each unit. When normalized by their average activations, the slowest features actually correspond to the lowest spatial frequencies (the mean being the slowest feature). This suggests that linear maps simply do not have the capacity to produce non-trivial slow features in natural videos.

5 Slowness with Pooling Operators

In order to address the lack of capacity of linear operators for forming slow features, we ask: what is the optimal architecture of G_{W_e} for extracting slow features? Slow features are by definition invariant to temporal changes. In natural video and on small spatial scales these changes mainly correspond to local translations and deformations [2]. Invariances to such changes can be achieved using approriate pooling operators [2, 13]. Such operators are at the heart of the most successful supervised feature learning architectures [12].

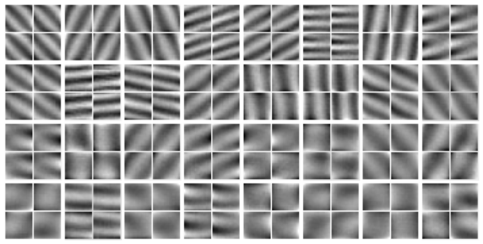


(a)

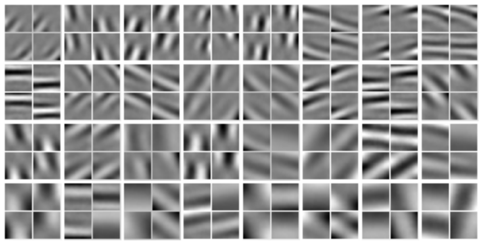


(b)

Figure 3: Relationship between slowness (blue curve) and activations (green curve) of the 128 code units without (a) and with (b) pooling.



(a)



(b)

Figure 4: Pooled decoder dictionaries learned without (a) and with (b) the L_1 penalty using (3).

Inspired by these results, let G_{W_e} be a two stage operator comprised of a learned linear map followed by a local pooling. Let the N hidden activations, $h = W_e x$, 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 nothing precludes us from using any norm, including $p = \infty$ which corresponds to max pooling. 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 [3] which is not possible with our linear inverse map W_d. Instead of reconstructing from z we reconstruct from the hidden representation h. This is the same approach taken in group sparse coding [10]. In order to promote independence among the features we additionally add a sparsifying L^1 penalty on the hidden activations. To facilitate truly sparse activations a rectifying point-wise non-linearity $f(\cdot)$ is applied to the hidden activations. Including the rectifying non-linearity becomes critical for learning sparse inference in a hugely redundant dictionary, e.g. convolutional dictionaries [7]. Denote the vector of hidden activations for the sample at time t as $h_t = f(W_e x_t)$. The complete loss functional is:

$$L(x_t, x_{t'}, W) = \sum_{\tau=\{t,t'\}} \left(\|W_d h_\tau - x_\tau\| + \alpha |h_t|\right) + \beta \sum_{i=1}^{K} \left| \|h_t\|^{P_i} - \|h_{t'}\|^{P_i} \right| \tag{3}$$

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

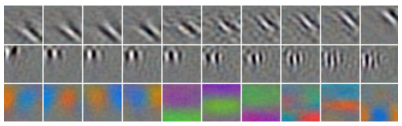


Figure 5: Pooling over a large number of units often leads to features that emulate convolution.

and the second stage recombines these partitions into temporally stable groups. Since the loss of information is uncontrolled after the pooling operator, ensuring that the pooling is done over sparse activation maps is a means of implicitly limiting the information lost by pooling.

Minimizing Equation 3 with $\alpha = 0$ results in the learned decoder basis shown in Figure 4a. Here a dictionary of 512 basis elements was trained which pooled in non-overlapping groups of four resulting in 128 output features. Only the slowest 32 groups are shown in Figure 4a. The average slowness and activations of each group are plotted in Figure 3b. Not only are the pooled features much slower, but their slowness is less dependent on their corresponding activations. 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 the translations are the principal source of variation at small spatial scales. This is a well known phenomenon, and the Fourier basis emerges in other models which maximize slowness using the L^2 pooling operator [14]. Minimizing Equation 3 with $\alpha > 0$ results in a more localized basis depicted in Figure 4b. This basis is more consistent with a local deformation model as opposed to a global one. For a detailed study on the trade-off between slowness and independence in feature learning see [14]. Pooling with overlap leads to coherence between features over longer extents. For example, the top of Figure 5 shows a groups of ten features which were obtained by pooling in groups of four with overlap. This group is well approximated by the translation of a single oriented edge, confirming that translations dominate the temporal variability even over larger spatial extents. This motivates lead us to experiment with translation invariant architecture in order to learn more varied and powerful invariance groups.

6 Slow Feature Learning using Convolutional Architectures

By replacing all linear operators in our model with convolutional filter banks translation invariance need not be learned [13]. In all other respects the convolutional model is identical to the fully connected model described in the previous section. Let the linear stage of the encoder consist of a filter bank which takes C input color channels 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 [5]. Spatial pooling induces local translation invariance in the feature maps. Pooling across feature maps allows the network to potentially learn feature gropus that are stable with respect to more general deformations.

The convolutional architecture was trained on contrast normalized video data consisting of $150,000$ frames extracted from YouTube videos, and tested on $20,000$ frames from different videos. Each color frame was down-sampled to a 32×32 spatial resolution in order to match the resolution of the CIFAR-10 dataset. Typical examples of features obtained by training fully connected architectures on the same data are shown in Figure 5. Note the factorization of color from structural features that is also commonly observed in the lower layers of networks trained with supervision [12].

In the following experiments, models with filter banks consisting of 32 kernels with 9×9 spatial support were trained. The L^2 pooling layer computes the local modulus *across* feature maps in non-overlapping groups of four. Thus the output feature vector z consists of eight 32×32 feature maps. Figures 6a and 6b show the decoders of two convolutional models trained with $\beta = 0.1, 0.5,$

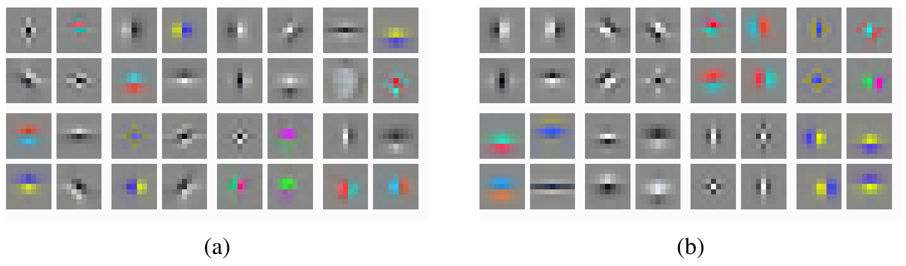


Figure 6: Pooled convolutional dictionaries trained with (a)β = 0.1 and (b) β = 0.5 slowness regularization strengths.

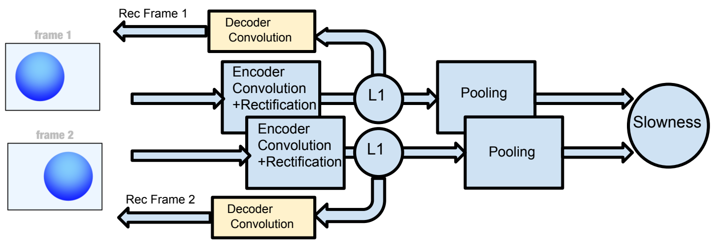
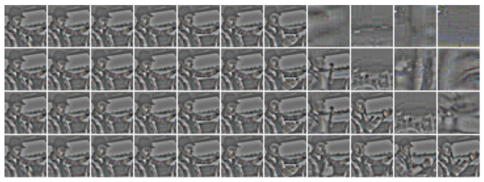


Figure 7: Block diagram of the convolutional model trained on pairs of frames showing the Siamese architecture.

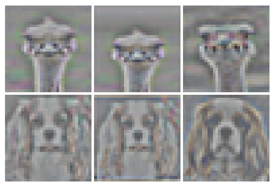
respectively, and a constant value of α. The filter bank trained with $\beta = 0.1$ exhibits almost no coherence within each pool group; the filters are not visually similar nor do they tend to co-activate at spatially neighboring locations. Most groups in the filter bank trained with $\beta = 0.5$ tend be visually similar, corresponding to similar colors and/or geometric structures.

To verify the connection between slowness and metric learning, we evaluate the metric properties of the learned features. It is well known that L^2 distance in the extrinsic (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 the temporal neighbors of the query frame. Figure 8a shows the top ten query results for a single frame (left column) in three spaces. The top row shows the top ten neighbors in pixel space. The following three rows show the neighbors computed in the feature spaces of models trained with progressively increasing slowness penalty (β). Note that the last row appears to contain the most frames from the same video sequence. Figure 8b shows the result of a query in the CIFAR-10 dataset. The left column shows two query images selected from the test set, the middle column shows the nearest neighbors in the metric space learned by our top performing model, and the right column shows the nearest neighbors in pixel space. Although the nearest neighbor in pixel space also belongs to the same category, the nearest neighbors retrieved in the learned metric are much more similar in appearance to the queries.

To measure the temporal coherence of a representation Φ, we consider for a given query example x_{t^*} the set of neighbors $\mathcal{N}(t^*, \delta) = \{t' \,;\, \|\Phi(x_{t^*}) - \Phi(x_{t'})\| \leq \delta\}$ as an estimator of the set of true temporal neighbors $\mathcal{N}_{true}(t^*) = \{t' \,;\, |t' - t^*| \leq \epsilon\}$. We fix $\epsilon = 4$ and we vary δ to obtain a precision-recall curve for each representation Φ, evaluated on a test set of 20,000 contrast normalized YouTube frames. Figure 9a shows that the models trained with slowness achieve better precision at high recall, although they have slightly worse precision at low recall than the input representation. This fact confirms that the resulting features are stable with respect to temporal variability, at the expense of a minimum loss of discriminability. We compare this to models trained with an L_1 penalty imposed on the output of the pooled representation, corresponding to convolutional group sparsity [10]. These models do not make use of the temporal coherence assumption, and are referred to as GSC 1-6 in Figure 9. Since the variability present in natural videos is likely to be uninformative when it comes to object classification, one might expect that the slow-feature representation also

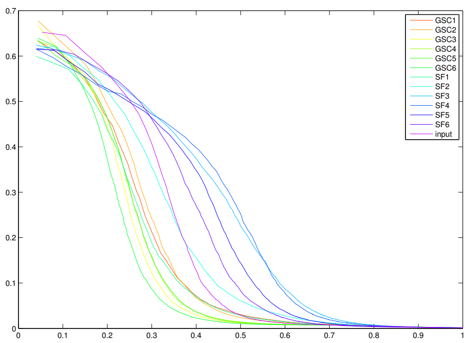


(a)

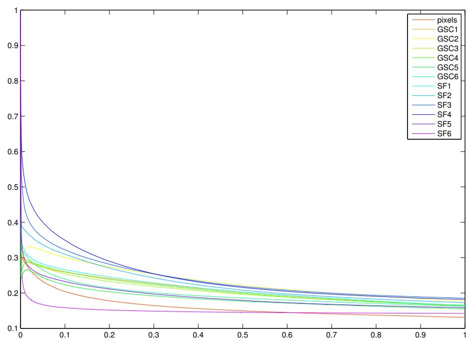


(b)

Figure 8: Query results in the (a) video and (b) CIFAR-10 datasets.



(a)



(b)

Figure 9: Precision-Recall curves corresponding to the YouTube (a) and CIFAR-10 (b) dataset. We compare contrast normalized pixel features, convolutional Group Sparse Coding models (GSC-β) for different values of β, and convolutional slow feature model (3) (SF-β) also for different β. In both datasets, the best performing model corresponds to $\beta = 0.5$.

provides a metric with improved consistency within object categories. This claim is then evaluated in the CIFAR-10 object classification dataset, by replacing $\mathcal{N}_{true}(t^*)$ with the label indicator $\mathcal{N}_{true}(i) = \{i' \,;\, l(x_i) = l(\tilde{x}'_i)\}$, where $l(i)$ is the label of the test example x_i and $l(\tilde{x}_{i'})$ is the label of training example $\tilde{x}_{i'}$. Figure 9b shows again that the metric corresponding to slow-feature models achieves better precision at high recall.

Pre-training is another popular alternative to leverage the information extracted from unsupervised learning to supervised tasks. We performed experiments on CIFAR-10 using a deep convolutional network, consisting in 3 layers of interleaved convolutions and L_2 pooling, followed by a fully connected layer. By initializing the first convolutional layer with the slow-feature convolutional model trained on $170k$ YouTube video frames, we reduced the test error from 24% to $\mathbf{22\%}$. Although still far from the state-of-the-art results, these preliminary results show that temporal coherence has the potential to further improve purely supervised deep architectures.

7 Discussion

Video data provides a virtually infinite source of information to learn meaningful and complex visual invariances. Slow feature learning has a long history at attempting to explain the mechanisms for learning such invariances. In this work we provide an auto-encoder formulation of the problem, and show that the resulting models indeed become more stable to naturally occurring temporal variability, while keeping enough discriminative power.

While temporal slowness is an attractive prior for good visual features, in practice it involves optimizing conflicting objectives, namely 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 information lost at each layer. This could be possible by replacing the linear decoder of model (3) with a non-linear decoder which can be used to reconstruct the input from pooled representations.

References

[1] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. Technical report, University of Montreal, 2012.

[2] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1872–1886, 2013.

[3] Joan Bruna, Arthur Szlam, and Yann LeCun. Signal recovery from pooling representations. In *ICML'2014*.

[4] Charles F. Cadieu and Bruno A. Olshausen. Learning intermediate-level representations of form and motion from natural movies. *Neural Computation 2012*.

[5] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *ICML'2013*.

[6] Rostislav Goroshin and Yann LeCun. Saturating auto-encoders. In *ICLR*, 2013.

[7] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *ICML'2010*.

[8] Raia Hadsell, Soumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR'2006*.

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

[10] Koray Kavukcuoglu, MarcAurelio Ranzato, Rob Fergus, and Yann LeCun. Learning invariant features through topographic filter maps. In *CVPR'2009*.

[11] Christoph Kayser, Wolfgang Einhauser, Olaf Dummer, Peter Konig, and Konrad Kding. Extracting slow subspaces from natural videos leads to complex cells. In *ICANN'2001*.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, volume 1, page 4, 2012.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[14] Jorn-Philipp Lies, Ralf M Hafner, and Matthias Bethge. Slowness and sparseness have diverging effects on complex cell learning. 10.

[15] Hossein Mobahi, Ronana Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *ICML'209*.

[16] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[17] M.A Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. *Advances in neural information processing systems*, 20:1185–1192, 2007.

[18] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Galrot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, 2011.

[19] Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001.

[20] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. Technical report, University of Montreal, 2008.

[21] Laurenz Wiskott and Terrence J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation 2002*.