Department of Computer Science
University of Toronto
http://learning.cs.toronto.edu

6 King's College Rd, Toronto
M5S 3G4, Canada
fax: +1 416 978 1455

May 28, 2015

UTML TR 2015–

Unsupervised Learning of Visual Representations using Videos

Nitish Srivastava
Department of Computer Science, University of Toronto

Abstract

This is a review of unsupervised learning applied to videos with the aim of learning visual representations. We look at different realizations of the notion of temporal coherence across various models. We try to understand the challenges being faced, the strengths and weaknesses of different approaches and identify directions for future work.

Unsupervised Learning of Visual Representations using Videos

Nitish Srivastava
Department of Computer Science, University of Toronto

1 Introduction

Videos are a virtually unlimited source of rich visual information. However, videos when represented as a temporal sequence of pixel intensities lie in an extremely high-dimensional space where distinct factors of variation such as illumination, pose, view point, camera motion, object motion, object identity and object attributes are heavily entangled. We would like to discover computations which can disentangle these factors to produce a representation that is equivariant (or invariant) to subsets of these factors. The hope is that such a representation would provide a generic and expressive interface between low-level visual inputs and high-level "AI". This would make it easy to solve tasks in the AI-set which require access to visual stimuli.

1.1 Motivation

Why videos ? The motivation for working with videos rather than with still images is that a temporally ordered sequence of images contains more information than an unordered one. This additional information lies in the temporal order. This information is valuable because temporal ordering is closely related to semantic structure. Since the semantics of a visual scene usually does not change rapidly over time, we can exploit the fact that any good visual representation must also not vary drastically over time. This provides a strong prior to constrain the search space for good visual representations and suggests that videos should help learn better visual representations. Besides, there are tasks that only operate on videos, such as activity recognition and tracking, where being able to aggregate information across time is crucial. Therefore videos are a natural playground for building representations aimed at solving these tasks. Moreover, if we take inspiration from biological forms of intelligence, we should not ignore the fact that visual systems exist, learn and evolve in an environment of continuously *changing* visual stimuli.

Why unsupervised learning ? Supervised learning has been extremely successful in learning good visual representations that not only produce good results at the task they are trained for, but also transfer well to other tasks and datasets. Therefore, it is natural to extend the same approach to learning video representations. This has led to research in 3D convolutional nets (Ji et al., 2013; Tran et al., 2014), different temporal fusion strategies (Karpathy et al., 2014) and exploring different ways of presenting visual information to convolutional nets (Simonyan and Zisserman, 2014). However, videos are much higher dimensional entities compared to single images. Therefore, it becomes increasingly difficult to do credit assignment and learn long range structure, unless we collect much more labelled

data or do a lot of feature engineering (for example, computing the right kinds of flow features) to keep the dimensionality low. The costly work of collecting more labelled data and the tedious work of doing more clever engineering can go a long way in solving particular problems, but this is ultimately unsatisfying as a machine learning solution. This highlights the need for using unsupervised learning to find and represent structure in videos. Moreover, videos have a lot of structure in them (spatial and temporal regularities) which makes them particularly well suited as a domain for building unsupervised learning models, even from a purely theoretical perspective.

1.2 Challenges

Evaluation The problem that we want to solve is how to find generic and expressive visual representations that serve as an interface to high-level AI. One of the key challenges is the inherent ambiguity of this problem. How do we quantify the notion of a representation being "generic and expressive" in a task-independent way ? How do we know that an interface is good when the thing that uses this interface (AI) is not very well defined ? In order to make progress, we need some meaningful way to quantify the goodness of representations. This is important for two reasons (1) as the source of a learning signal for training models (following the gradient that goes from worse to better parameters) and (2) as a way of comparing distinct models. We can think of (2) in the context of a search space over machine learning models that is being explored by the research community.

One solution is to consider probabilistic generative models and quantify the model's performance by measuring how likely the model is to generate some test data. This method has the advantage of being well-defined and meaningful to compare across distinct models. However it has the disadvantage that it is only applicable to probabilistic generative models. Moreover, it is often hard to compute or even reasonably approximate log likelihood. Often the approximations are either unbiased but high variance or fairly stable but so far from the true value that they are not useful. Also, the more complicated a model we design, the harder it becomes to evaluate its log-likelihood.

An alternative is to consider examples of supervised learning tasks, for example, object recognition or action recognition and try to solve them using the representations learned from unsupervised models. The performance on the supervised task is indicative of the quality of the unsupervised model, especially if the size of the labelled data is rather small. This has the advantage of being applicable to all models, not just probabilistic generative ones. It also fits nicely with the original motivation of enabling AI tasks. The disadvantage is that specific supervised learning tasks may bias our search and direct it away from being truly generic. Also it gets hard to justify this approach because if we really cared about solving a particular supervised problem, there may be better ways to do that, for example, by throwing lots of supervised data at large neural nets.

The third solution is to look for desirable properties of the learned representations. For example slowness, sparsity, high entropy, invariance to certain transforms, the ability to recover the input, ability to retain information about the input, and so on. These can sometimes be incorporated as priors in generative models or enforced directly through loss functions in non-probabilistic settings.

Computational The next challenge is purely computational. Large video datasets are hard to handle. Videos need to be compressed in order to be stored efficiently. When making minibatches they need to be decompressed and shuffled. Random access in a compressed video is not efficient.

Also information is quite sparse, in the sense that most of the time, in most regions of space, nothing interesting happens. There seems to be no easy solutions to handle this challenge, except careful engineering while writing data handlers.

1.3 The Underlying Theme: Temporal Coherence

Most unsupervised machine learning effort applied to visual perception has focused on still images. All those ideas are obviously relevant for videos as well along the spatial dimension. Keeping this in mind, this report will focus on all things temporal and temporal coherence will be an underlying theme.

Coherence across time is a ubiquitous feature of visual sensory data. Things in the world change locally and slowly, if at all. This is a fundamental artifact of living in a low energy, low temperature physical system with localized, embodied objects. Consider an object moving across a visual frame. Things that we might care about, for example object identity, do not vary at all with time. The object's pose varies slowly and smoothly. However low-level features, for example the state of a particular pixel or the response of a linear filter at some patch vary at a much faster rate, especially in the presence of background clutter and camera motion. This indicates that if we could extract features that are coherent across time, we might be able to get rid of the variations that we don't care about and capture the ones that we do. Of course being coherent alone will lead to trivial solutions like being constant. Therefore, the features must also be informative in some way. It should be emphasized that temporal coherence is not just about features changing slowly. It can be generalized to mean that features change predictably. Therefore, temporal coherence is not just about coherence in the value of the features but coherence in the process that explains the features.

One of the earliest mentions of temporal coherence in the context of unsupervised machine learning was by Hinton (1989), page 208, where it is suggested that we could insist that all or part of the code "change as slowly as possible with time" in order to get a signal for self-supervised backpropagation. Since then a huge amount of research has been done in trying to use temporal coherence. All the models that we review in this paper try to use temporal coherence in some way. They differ in how they quantify and capture coherence, how they try to prevent collapse and what tasks they aim to solve. They can be categorized into the following sections.

- Early progress made in designing Hebbian update rules that encourage temporal coherence.
- Models that quantify coherence based on information-theoretic measures such as entropy and mutual information.
- Probabilistic generative models for temporal sequences and image pairs.
- Models that use objective functions designed to encourage slowness and sparsity but are not necessarily motivated by or derived from probabilistic models.
- Models that try to solve various computer vision tasks involving videos. As opposed to typical machine learning models which solve simple tasks by learning rich representations with little domain knowledge, most of these models use simple fixed representations but a lot of domain

knowledge to solve complicated tasks. We review these models with the aim of assessing if they can be extended so that the learning signal can be used to learn the visual representations.

2 Hebbian update rules

The standard Hebbian learning principle (Hebb, 1949) is that units that fire together, wire together. In a computational setting, this principle leads to an update rule for the strength of the synapse (weight w_{ij}) connecting unit j to unit i

$$\Delta w_{ij} = \epsilon y_i (x_j - w_{ij}), \tag{1}$$

where y_i is the activation of the post-synaptic unit i, x_j is the activation of the pre-synaptic unit j and ϵ is a learning rate. The weight decay term is added to keep the weight bounded. Hinton (1989) mentioned that one possible way of incorporating temporal coherence would be to use the post-synaptic activity from the previous time step, instead of the current one. In a similar spirit, Földiák (1991) proposed a modified Hebbian rule which uses the trace of the post-synaptic unit's activity (exponential running average) in place of its current activation.

$$\Delta w_{ij}^{(t)} = \epsilon \bar{y_i}^{(t)} \left(x_j^{(t)} - w_{ij}^{(t)} \right),$$

where $\bar{y_i}^{(t)} = (1 - \delta)\bar{y_i}^{(t-1)} + \delta y_i^{(t)}$. Combined with some lateral competition for turning units "on", this learning rule encourages features to be coherent over time because the features which can stay coherent will have a high running average and therefore will be able to get a large learning signal. This temporal low-pass filtering of the network's activations incorporates an inductive bias that the learned features should be stable over time. This was one of the first attempts to use temporal coherence to learn invariant representations. This method was shown to work in a simple scenario where the input consisted of straight lines in different orientations moving across a visual field. The linear separability of the input made this particular scenario easy. In order to extend this approach to more realistic settings, we would need a multi-layer neural net (or some other powerful non-linear transform) to extract that linearly separable space from raw inputs, but from this work it was not obvious what the learning signal for that non-linear transform would be.

Mitchison (1991) proposed an anti-Hebbian term designed to remove short-term temporal variations. The term looks like

$$\Delta w_{ij}^{(t)} \propto -\Delta y_i^{(t)} \Delta x_j^{(t)}, \tag{2}$$

where $\Delta y_i^{(t)} = y_i^t - y_i^{t-1}$ and $\Delta x_i^{(t)} = x_i^t - x_i^{t-1}$ are the temporal differentials of the output and input unit respectively. This terms says that if changes in the input over time are positively correlated with changes in the output, the weight connecting them should be lowered. The contribution from this term will be small only when changes in the input over time do not produce a large change in the output. Therefore, the output will be temporally coherent. Until this is achieved, this term would keep pushing the weights to be small. This is essentially a regularization term that encourages temporal coherence since it prefers zero weights over weights that produce large changes in the output over a short period of time.

Stone and Bray (1995) derived a combination of Hebbian and anti-Hebbian updates from a simple observation regarding temporal coherence. They argue that we want the outputs of neurons to have small short-term variance but large long-term variance. This would make them temporally coherent over short time scales but prevent collapse by making them vary a lot over long time scales. Suppose $y^{(t)}$ denotes the output of a neuron at time t, $\tilde{y}^{(t)}$ the short term average (an exponential average with a fast decay) and $\bar{y}^{(t)}$ a similar long term average. We can get the desired behaviour by maximizing the following objective -

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

This objective says that the log ratio of the long-term variance and the short-term variance must be as large as possible. Suppose this neuron is linear and the weight coming into it from input x_j is w_j. Then taking derivatives wrt w_j-

$$\begin{aligned}\frac{\partial F}{\partial w_j} &= \frac{1}{V}\sum_t (y^{(t)} - \bar{y}^{(t)})\left(\frac{\partial y^{(t)}}{\partial w_j} - \frac{\partial \bar{y}^{(t)}}{\partial w_j}\right) - \frac{1}{U}\sum_t (y^{(t)} - \tilde{y}^{(t)})\left(\frac{\partial y^{(t)}}{\partial w_j} - \frac{\partial \tilde{y}^{(t)}}{\partial w_j}\right) \\ &= \frac{1}{V}\langle (y - \bar{y})(x_j - \bar{x}_j)\rangle - \frac{1}{U}\langle (y - \tilde{y})(x_j - \tilde{x}_j)\rangle.\end{aligned}$$

Therefore, we have an update rule that contains two terms - a Hebbian term and an anti-Hebbian term. This can be interpreted as saying that if V is large compared to U then we are in a domain where preventing collapse is not a major concern, therefore the model's learning is mostly anti-Hebbian and aimed towards reducing temporal variability and improving coherence. On the other hand, if V is smaller than U, then collapsing is becoming a concern and the model's learning switches to Hebbian, aimed at increasing the variability of the output in response to the input. This update rule elegantly combines the anti-Hebbian term (Eq. 2) proposed by Mitchison (1991) with standard Hebbian learning. This approach is one of the first examples of the use of a contrastive term to prevent collapse in the context of learning temporal coherence. However, a drawback of this method is that the objective function F is a per-neuron objective. In order to make multiple neurons discover different features we would need some way to decorrelate the outputs of multiple neurons.

Becker (1996) proposed a Hebbian learning rule for Gaussian Mixture Models where the mixing proportions are not fixed parameters, but are functions of some context. The aim was to produce a clustering that is temporally coherent. Consider a Gaussian Mixture Model over K components-

$$\begin{aligned}p(\mathbf{x}) &= \sum_{k=1}^{K} \pi_k y_k(\mathbf{x}), \\ y_k(\mathbf{x}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k).\end{aligned}$$

When the log-likelihood $\sum_{\mathbf{x}} \log(p(\mathbf{x}))$ is optimized with gradient descent, the contribution to the update for the mean $\boldsymbol{\mu}_k$ due to any data case $\mathbf{x}$ is

$$\Delta\mu_{kj} = \epsilon \frac{\pi_k y_k(\mathbf{x})}{\sum_{k'} \pi_{k'} y_{k'}(\mathbf{x})} \left(x_j - \mu_{kj}\right).$$

Comparing this to Eq. 1, this is a Hebbian update rule involving a normalized post-synaptic activation, pre-synaptic action x_j and weight μ_{kj}. Instead of π_k being fixed parameters, Becker (1996) make them a function of some context $\mathbf{c}$ of the current input $\mathbf{x}$.

$$\pi_k(\mathbf{c}) = \frac{\exp(c_k)}{\sum_{k'} \exp(c_{k'})}$$

The context could be a (learned) function of, for example, the preceding sequence of inputs or the preceding cluster probabilities. If the cluster predicted by the context $\pi_k(\mathbf{c})$ agrees with that predicted by the current input $y_k(\mathbf{x})$, then the weight update will be large since the update depends on the product of the two. Therefore, the model is being encouraged to discover clusters such that they can be predicted in agreement with the context signal, which can be seen as enforcing temporal coherence. This model is also an example of a multiplicative interaction between context and current input.

The Hebbian update rule here was derived from trying to optimize a cost function over multiple neurons, unlike the previous work from Földiák (1991), Mitchison (1991) and Stone and Bray (1995). More broadly, the approach in the research community has drifted away from formulating biologically plausible update rules towards designing global loss functions, whether or not they lead to local update rules is of secondary importance. One driving force behind this is that if we have a well-defined objective function then it becomes easier to add complexity to the model, for example, by adding more layers to a network, without having to worry too much about how to make that complexity be useful. In other words, it decouples what we want to achieve from how we want to achieve it. This provides the flexibility of choosing objectives through which desired properties can be enforced, for example, temporal coherence. For generative probabilistic models, the objective function is often the log-likelihood of the data. Another class of objective functions is based on information theoretic measures that try to quantify information or independence in the learned representations. We describe those next.

3 Measuring Information and Independence

A sensible objective of unsupervised representation learning is to learn a representation that is maximally informative about its input. After all, in the absence of a supervised task, we should try to extract as much information as possible about the input. The challenge is how to represent this information and measure it quantitatively. This measurement must be such that it is easy to compute. It would also be nice if it was differentiable so it can be optimized easily. Shannon (1948) showed that entropy of a distribution is a measure of how much information is obtained by observing samples from it. The broad result in this area is that given samples from a distribution, its entropy can be measured if we are willing to make certain assumptions about the distribution. Estimating information gets harder if we do not make assumptions. The research in this area is mostly driven by making sensible assumptions and seeing what properties emerge.

3.1 Mutual Information

Mutual Information between random variables X and Y is defined as

$$I(X,Y) \equiv H(X) + H(Y) - H(X,Y) = H(Y) - H(Y|X),$$

where $H(X) \equiv -\sum_{x \in \mathcal{X}} p(x) \log p(x)$ is the entropy of random variable X which takes values in the domain $\mathcal{X}$. I is high when each variable has a lot of information (high entropy) but given one variable, knowing the other does not give much additional information. Becker and Hinton (1992) proposed that common causes, that affect separate but related parts of some perceptual input, can be found by maximizing the mutual information between the outputs of computational modules each of which only looks at one of the parts. For instance, they showed that mutual information between outputs of modules that look at non-overlapping, spatially adjacent patches of random-dot stereograms can be used to discover the common feature – depth. The idea is to exploit spatial coherence of features such as depth which are likely to be the same or change slowly across the visual input. Becker (1992) applied this idea to temporally adjacent inputs. Here, the common feature was the identity of a 6-dimensional random pattern as it translated (with wrap-around).

The ability to compute the entropy and its derivatives is often the limiting factor in using this objective function. If the domain of the random variable is small, for example, if it is a single binary or multinomial unit, then the entropy can be measured by directly enumerating the space. This was the case in Becker (1992) where the variable was the output of the softmax trying to classify the input. Otherwise, it becomes hard to measure the entropy unless an assumption is made about the distribution of the variable. For example Becker and Hinton (1992) in the case of random-dot stereograms assume that the output of each module is the true depth corrupted with independent Gaussian noise. Under this assumption, computing the mutual information and its derivative become tractable. For two variables a and b which are corrupted versions of the true underlying signal,

$$I = \frac{1}{2} \log \frac{V(a+b)}{V(a-b)}$$

is a good measure of the mutual information between the mean of a and b and the underlying true signal, where V denotes variance. In general, computing entropy is hard. Viola et al. (1995) proposed a method for approximating entropy using Parzen density estimates to avoid the use of any parametric assumption about the true density function. However, the method is hard to extend to very high dimensional spaces because it is hard to get reliable Parzen density estimates in those cases.

Zemel and Hinton (1991) used mutual information to discover viewpoint invariant representations of an object. The model looks at different parts of the object and outputs some representation. The representations extracted at these different locations are made to agree. The motivation is that attributes like position, orientation and object size are common and if the model learns to extract those, then it can have high mutual information.

In all these models, the mutual information between two outputs of a network was used as the objective function to train the network. Next, we will look at models where the mutual information between the input and output of a network is used to train it.

3.2 Infomax

Information maximization was first proposed by Linsker (1988) as the underlying principle for learning in networks in the context of visual systems. The main idea was that networks learn to be so that the maximum amount of information gets transferred across each layer of the network, subject to constraints. So if $\mathbf{x}$ is the input to a neuron and y is its output, the weights will be set so that y conveys as much information as possible about $\mathbf{x}$. This principle is called "infomax".

For example, consider a linear neuron $y = \sum_i w_i x_i + b + \eta$ where η is independent Gaussian noise with mean zero and standard deviation σ. If we assume y to be Gaussian distributed with variance V, then the rate at which information flows through this neuron is $0.5 \log V/\sigma^2$. Therefore, the information can be maximized by increasing the variance. But then the information can be arbitrarily increased by just increasing the scale of the weights. So we can place a constraint on the weights, for example that $||\mathbf{w}||_2^2 = 1$. Since the norm of the weights is bounded, we are asking the model to find a direction $\mathbf{w}$ such that the variance of the projection of the inputs along that direction is maximized, which corresponds to finding the first principle component. If we have a whole bunch of neurons $\mathbf{y}$ and we simultaneously find these directions for each unit while keeping them orthogonal to each other, then would recover Principle Component Analysis (PCA). Imposing a Gaussian assumption on y led us to derive PCA. If we place a different "non-Gaussianity" assumption, then we could discover other representations. A lot of research in the field of Independent Components Analysis (ICA) specializes in making these assumptions and finding interesting representations. Some of these will be reviewed in this paper. However, if we insist on not making any assumptions, the problem becomes much harder to solve. We would typically require some non-parametric density estimator to express the probability distribution over the output space, and use that to measure entropy.

Before we look at am application of infomax to temporal sequences, it is insightful to derive the update rules that fall out of this objective and compare them to the Hebbian rules we saw earlier. A lot of this analysis is borrowed from Bell and Sejnowski (1995). Suppose a neural net takes X as input and produces Y. The aim is to maximize the mutual information between X and Y. This can be formulated as maximizing

$$I(X, Y) = H(Y) - H(Y|X)$$

where $H(Y)$ is the entropy of the output and $H(Y|X)$ is the entropy in Y conditioned on X. If the model that produces Y given X is deterministic, then $H(Y|X)$ diverges to $-\infty$ if Y is real-valued. However, this is fine because H is actually differential entropy and is defined with respect to some noise level (for example, σ for the linear neuron above). Therefore, we can just concern ourselves with maximizing the entropy $H(Y)$. Suppose X and Y are both one dimensional entities and we have an invertible function g that maps X to Y. The input has a probability density function $p_X(x)$. Then the probability density function of the output $p_Y(y)$ is given by

$$p_Y(y) = \frac{p_X(x)}{|\frac{\partial y}{\partial x}|}.$$

The entropy of this probability density function is

$$H(Y) = -E[\log p_Y(y)] = E\left[\log\left|\frac{\partial y}{\partial x}\right|\right] - E[\log p_X(x)].$$

The term $E[\log p_X(x)]$ is a constant, independent of the model. Therefore, we can ignore it during optimization. The derivative of $H(Y)$ wrt any model parameter w is

$$\frac{\partial}{\partial w}H(Y) = E\left[\left(\frac{\partial y}{\partial x}\right)^{-1}\frac{\partial}{\partial w}\left(\frac{\partial y}{\partial x}\right)\right].$$

We can approximate the expectation by the mean over the data distribution. In particular, suppose our model is a single weight w and bias b followed by a sigmoid non-linearity,

$$y = \frac{1}{1+\exp(-(wx+b))}.$$

Then

$$\begin{aligned}\frac{\partial y}{\partial x} &= wy(1-y),\\ \frac{\partial}{\partial w}\left(\frac{\partial y}{\partial x}\right) &= y(1-y)(1+wx(1-2y)),\end{aligned}$$

and the updates are-

$$\begin{aligned}\Delta w &\propto \frac{1}{w} + x(1-2y),\\ \Delta b &\propto 1-2y.\end{aligned}$$

The bias update has the nice interpretation that it will be zero only if the expected value of y is 0.5. This means that the model will center the sigmoid around the mean of the input distribution. The $x(1-2y)$ term in the weight updates will change the weights to try to match them with the scale of the input distribution. The $1/w$ term keeps the w from collapsing to 0. This is a rather interesting situation where we have an anti-Hebbian rule involving the activities and an anti-decay term that prevents collapse. Usually, we have a Hebbian rule involving the activities and a weight decay.

In general, if both X and Y are vectors of length N, then this generalizes to

$$p_Y(\mathbf{y}) = \frac{p_X(\mathbf{x})}{|\det\ J|},$$

where J is the Jacobian whose entries are given by $J_{ij} = \frac{\partial y_i}{\partial x_j}$. This leads to

$$\begin{aligned}\Delta\mathbf{W} &\propto \left[\mathbf{W}^\top\right]^{-1} + (1-2\mathbf{y})\mathbf{x}^\top,\\ \Delta\mathbf{b} &\propto 1-2\mathbf{y}.\end{aligned}$$

Instead of just preventing $\mathbf{W}$ from being 0, the $\left[\mathbf{W}^\top\right]^{-1}$ term now prevents any linear dependence

between the weights, making the different weights discover different features. This is an interesting way in which it prevents collapse. However, this technique only works for full dimensional weights. If the output has fewer units than inputs, the mapping is not invertible and this technique would not apply directly (det $J = 0$).

Bell and Sejnowski (1995) applied this to the source separation problem also known as the cocktail party problem. In this problem, there are N independent temporal signals (N people talking at a party). There are N sensors that are recording the sound. Each sensor receives a different mixture of the N signals. The input signal $\mathbf{x}^{(t)}$ represents the input at the N sensors at time t. The problem is to separate this input signal to recover the N independent signals $\mathbf{y}^{(t)}$ where each dimension corresponds to one of the speakers. By maximizing the mutual information between $\mathbf{x}^{(t)}$ and $\mathbf{y}^{(t)}$ they were able to separate up to 10 independent sources including 6 human speakers, rock music, raucous laughter etc. The temporal problem being solved here is *disentangling* an additive mixture of independent factors of variation. This is much simpler than disentangling the factors of variation in a visual scene (object identity, motion, pose, camera motion etc) because they combine in overwhelmingly intricate ways. But looking at the strengths and limitations of these approaches when applied to a simple problem can potentially help understand when and how they would fail for bigger problems.

3.3 Independent Component Analysis

In this section we review models based on ICA. ICA has been used to learn representations in numerous domains. The principle underlying ICA was first introduced by Herault and Jutten (1986) and stated more clearly by Comon (1994). The idea has since then been explored by a number of researchers (Bell and Sejnowski, 1995; Amari et al., 1996; Hyvärinen and Oja, 1997, 2000). In ICA, given an input $\mathbf{x}$, we would like to transform it into an encoding $\mathbf{y} = \mathbf{W}\mathbf{x}$ such that the components y_j are as statistically independent as possible. The notion of independence has been quantified in numerous ways – correlation, kurtosis, "non-Gaussianity", differential entropy (negentropy), mutual information, etc.

ICA has been used to solve the cocktail party problem. The motivation is that if we can obtain N signals that are as independent as possible, then they should correspond to the N speakers. Jutten and Herault (1991) proposed a neural network architecture to solve this problem. They proposed a set of update rules to learn the network's parameters, but the rules did not correspond to any global loss function. However, they showed that the fixed point of the updates is reached when the correlation between different output dimensions is zero. In this way, this model used second-order statistics of the output signals (i.e. the correlation $\langle y_j^{(t)} y_i^{(t)} \rangle_t$) as a measure of independence. However, since we want $y_i^{(t)}$ and $y_j^{(t)}$ to be independent, not just uncorrelated, all non-linear functions of them must also be uncorrelated. Therefore, the authors extended their approach to second-order statistics of fixed arbitrary non-linear functions of the output signals (i.e., the correlation $\langle f(y_j^{(t)}) g(y_i^{(t)}) \rangle_t$, where f, g are some fixed non-linear functions, for instance, $f(x) = x^3, g(x) = \sin(x)$). This makes the output signals more independent. However, a drawback of their approach is that they only took into account statistics at the same time point t.

Ideally, we would want independence between components across time as well. In other words, we want $y_i^{(t)}$ and $y_j^{(t+\tau)}$ to be independent for all $i \neq j$ and for all $\tau \geq 0$. Molgedey and Schuster

(1994) try to take this account. They choose a single fixed time-delay τ and minimize

$$L = \sum_{i \neq j} \langle y_i^{(t)} y_j^{(t)} \rangle^2 + \sum_{i \neq j} \langle y_i^{(t)} y_j^{(t+\tau)} \rangle^2, \tag{3}$$

while keeping the variance $\langle y_i^{(t)} \rangle^2$ constrained to be 1. Ziehe and Müller (1998) extended this to several time steps τ_k

$$L = \sum_{k=0}^{K} \sum_{i \neq j} \langle y_i^{(t)} y_j^{(t+\tau_k)} \rangle^2,$$

where $\tau_0 = 0$. They also extended it by applying fixed arbitrary non-linear functions to y_i's and y_j's. These models do take into account correlations at other time steps, but they still quantify independence in terms of *correlation*. By assuming that decorrelation and independence are the same thing, we make an implicit Gaussian assumption about the signals in question. This can be ameliorated to some extent by looking at non-linear functions of the signals (Jutten and Herault, 1991) but ultimately these are still second-order statistics, when what we really want is to look at statistics at all orders.

The use of higher order statistics for example kurtosis (4-th order moments) has also been proposed. Kurtosis is usually defined as

$$\frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2} - 3$$

This essentially measures the peakedness of a distribution. The Gaussian distribution has zero kurtosis. Spikier distributions have higher kurtosis and flatter distributions have lower kurtosis. It has been used as a measure of non-Gaussianity. However, it is very sensitive to outliers and is not robust. Cardoso and Souloumiac (1993) used 4th order cumulants for separating non-Gaussian signals.

3.4 Independent Subspace Analysis

ISA (Hyvärinen and Hoyer, 2000) is a generalization of ICA where instead of having each component y_j be independent, the norms of the projection of $\mathbf{x}$ onto subspaces defined by subsets of these of components are independent. More precisely, suppose the projection $\mathbf{y}$ is divided into J subsets of K units each. Let y_{jk} denote feature k in group j, and $\mathbf{w}_{jk}$ denote the weight that connects it to the input $\mathbf{x}$.

The norm of the projection of $\mathbf{x}$ into subspace j is given by

$$a_j(\mathbf{x}) = \left(\sum_{k=1}^{K} \left(\mathbf{w}_{jk}^\top \mathbf{x} \right)^2 \right)^{1/2}.$$

ISA maximizes the following objective

$$L = \frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{J} \log p(a_j(\mathbf{x}^{(n)})) \quad \text{subject to } WW^\top = \mathbf{I}.$$

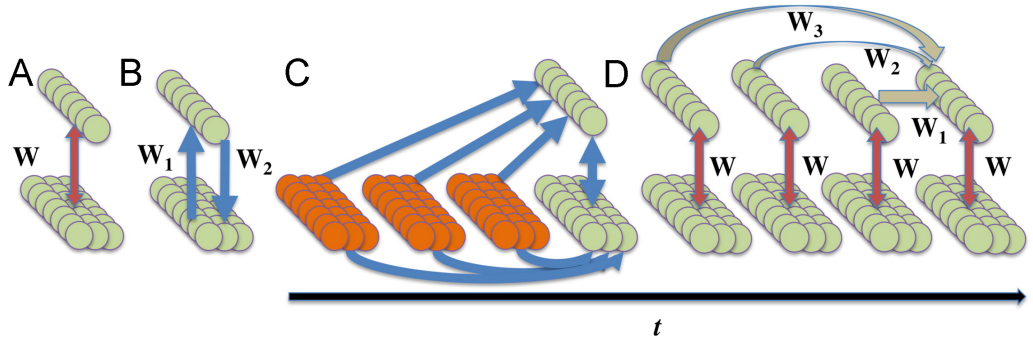


Figure 1: (Figure reproduced from Häusler et al. (2013)) (A) RBM (B) Autoencoder (C) Conditional RBM (D) Temporal RBM.

Here decorrelation is used as a surrogate for independence, although other forms of independence can also be enforced. $p(a_j(\mathbf{x}))$ is chosen to be some sparse distribution. Often this is the exponential distribution,

$$\log p(a_j(\mathbf{x})) \propto -\lambda a_j(\mathbf{x}) + \beta.$$

Le et al. (2011) applied ISA to learn an encoding of spatio-temporal cubes sampled from natural videos.

4 Probabilistic Generative Models of Temporal Sequences

Probabilistic generative models offer a principled framework for expressing unsupervised learning models. The basic idea is to define a probability density function P over the domain of the data $\mathcal{X}$. Given unlabelled data $\mathcal{D}$ we assume that it consists of samples drawn i.i.d from the true unknown data generating distribution. The maximum likelihood objective function is to learn the distribution P such that the probability of drawing the samples in $\mathcal{D}$ from P is as high as possible. A huge number of models have been proposed to understand temporal sequences using probabilistic generative models. In this section, we will focus on models that have been built for visual data or can be potentially applied to them.

4.1 Temporal Boltzmann Machines

The Restricted Boltzmann Machine (RBM) is a simple undirected generative model. Temporal extensions of RBMs have been studied to model temporal sequences. Fig. 1 shows a conditional RBM and a temporal RBM. These models incorporate temporal coherence by conditioning the hidden state of an RBM at any time by the states at previous time steps. However, note that the conditioning is additive in the sense that previous visible and/or hidden states add a bias to the current state. The case of the TRBM is a little more complicated since the proper way to do inference at time t here would be to marginalize the effects of previous timesteps. Sutskever et al. (2009) proposed Recurrent Temporal RBMs (RT-RBMs) which are similar to TRBMs, except that they remove this problem by removing the undirected connection between the hidden and visibles and replacing that by a directed connection from the visible to hidden. The effect of the hidden state on the visible is through a time

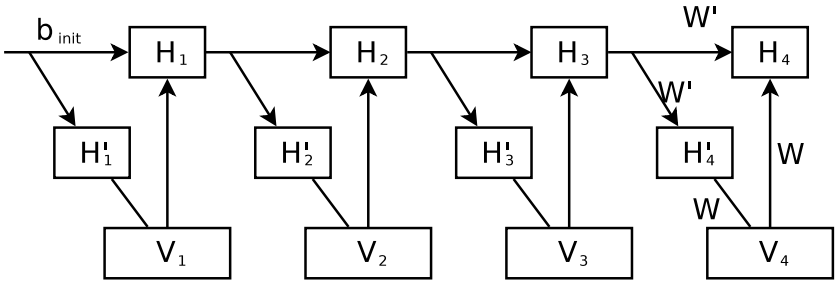


Figure 2: (Figure reproduced from Sutskever et al. (2009)) A Recurrent Temporal Restricted Boltzmann Machine (RTRBM).

delayed recurrent connection. The resulting model is shown in Fig. 2. This was used to model a video of bouncing balls and worked better than a TRBM. Boulanger-Lewandowski et al. (2012) proposed RNN-RBMs (Fig. 3). In this model there is a recurrent neural net running that takes as input the observed temporal sequence. There is also a conditional RBM that looks at the input at each time step. The RBM is conditioned on the states of the RNN from the previous time step. This model separates the hidden state of the RNN from the hidden state of the RBM. Therefore the RNN hidden states model the temporal dynamics while the RBM hidden states model the appearance (conditioned on the temporal dynamics). This tries to separate motion modeling capacity from the appearance modeling one. In the RT-RBM, the same hidden state tries to perform both jobs simultaneously. The RNN-RBM was shown to empirically work better than RT-RBM on the task of modeling polyphonic music.

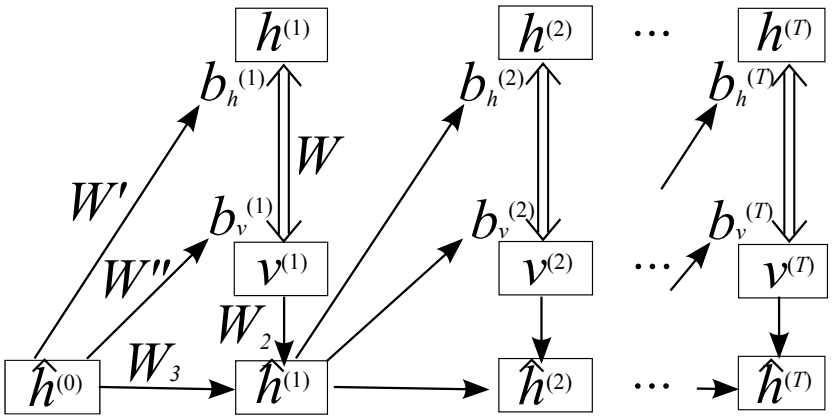


Figure 3: (Figure reproduced from Boulanger-Lewandowski et al. (2012)) A Recurrent Neural Network Restricted Boltzmann Machine (RNN-RBM).

4.2 Gated Restricted Boltzmann Machines and Autoencoders

Most of the models that we have considered till now only use additive interactions between the information coming from the previous time steps to better model the data at the current time step. Models

that take use multiplicative interactions have also been proposed. Gated Restricted Boltzmann Machines (GRBMs) were proposed by Taylor and Hinton (2009) to model human motion-capture data. Memisevic and Hinton (2010) applied GRBMs for learning the representation of the *transformation* that happens between pairs of temporally adjacent video frames. The motivation was to encode the *transform* and not the image that is being acted upon by the transform, so that motion can be separated from appearance. This model was applied in a convolutional way by Taylor et al. (2010). Memisevic (2011) proposed Gated Autoencoders which are same as GRBMs, except that the generative probabilistic model is replaced by an autoencoder with sparsity. Memisevic (2013) has a review of these related models. Michalski et al. (2014) extend this idea by modelling not just pairs but longer images sequences. In this case, instead of just representing the velocity-like features, higher-order features corresponding to higher-order time derivatives of the motion, such as acceleration are also modelled.

4.3 RNN Autoencoders

Recurrent neural nets, especially the ones based on Long Short Term Memory (LSTMs)(Hochreiter and Schmidhuber, 1997) have recently become very popular for modeling sequential data. The RNN encoder-decoder framework was proposed by Sutskever et al. (2014) for machine translation. This was applied by Srivastava et al. (2015) for learning a model that represents a video sequence (features from a convolutional net or image patches) by encoding it with an LSTM. The model is trained to decode the state at the last time step of the encoder using another LSTM to predict the future and/or reconstruct the input sequence back. Ranzato et al. (2014) also proposed a model that tries to predict future image patches. They make an interesting innovation by representing the target using a one-of-K encoding to avoid using an L_2 loss when reconstructing the pixels.

4.4 Sparse Coders

Sparse coding was introduced by Olshausen and Field (1996) in the context of learning first level features from static frames. They showed that these features are similar to the features empirically observed in the V1 region of mammalian brains. Sparse coding can be seen as a directed generative model which has latent variables $\mathbf{h}$. In the simplest case, the generative model is that the code $\mathbf{h}$ is drawn from a prior distribution which is sparse, and the data is a linear function of $\mathbf{h}$, plus additive Gaussian noise.

$$\begin{aligned}\mathbf{h} &\sim P(\mathbf{h}) \\ \mathbf{x} &\sim \mathcal{N}(\mathbf{W}\mathbf{h}, \sigma^2)\end{aligned}$$

Learning can be formulated as optimizing the following objective

$$L(\mathbf{W}, \{\mathbf{h}_n\}_{n=1}^N) = ||\mathbf{W}\mathbf{h}_n - \mathbf{x}_n||^2 - \sigma^2 \log P(\mathbf{h}_n),$$

where $\log P(\mathbf{h})$ could be $||\mathbf{h}||_2^2$ (Gaussian prior), $||\mathbf{h}||_1$ (Laplace prior), or some other creative prior. The main intuition behind increasing sparseness is that is not feasible for biological neurons

to be highly active all the time. Even though a high entropy state is most efficient for information transmission, when we trade off information flow rate with the cost of action potential generation required for firing, having a small number of neurons firing is optimal. In order to encourage temporal coherence we can now design suitable priors. Willmore and Tolhurst (2001) give a detailed analysis of some of these priors.

Hyvärinen et al. (2004) propose an interesting sparse coder model that learns spatio-temporal "bubbles" of activity. They enforce three things – sparsity, a topology on the filter space and temporal coherence. Fig. 4 shows the patterns of activity this model produces, compared to models that enforce only some of the three things.

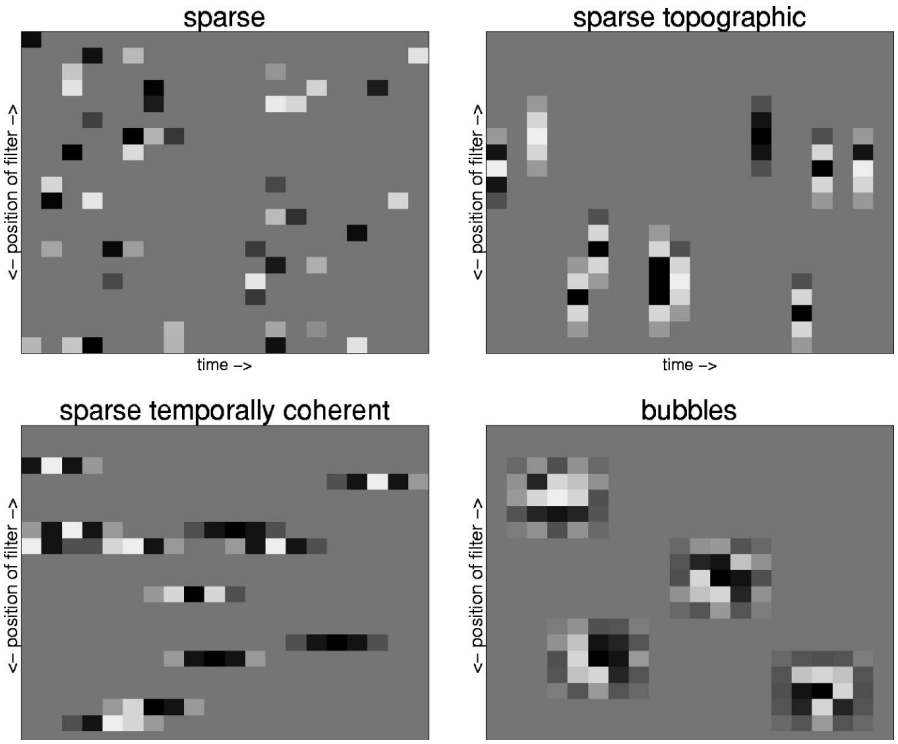


Figure 4: (Figure reproduced from Hyvärinen et al. (2004)) Spatio-temporal bubbles.

Häusler et al. (2013) proposed a model where they use a sparse autoencoder to pretrain temporal weights for a TRBM. In a recent preprint, Goroshin et al. (2015) proposed a sparse autoencoder that incorporates temporal coherence to regularize the learned code. The loss function is

$$L(\mathbf{x}^{(t)}, \mathbf{x}^{(t')}, \theta) = \sum_{\tau=t,t'} \left(||f_{\text{dec}}(\mathbf{h}^{(\tau)}) - \mathbf{x}^{(\tau)}||^2 + \alpha|\mathbf{h}^{(\tau)}| \right) + \beta \sum_{i=1}^{K} \left| ||\mathbf{h}^{(t)}||^{P_i} - ||\mathbf{h}^{(t')}||^{P_i} \right|,$$

where t, t' are adjacent frames in a video, θ denotes the parameters of the encoder f_{enc} and decoder f_{dec}, $\mathbf{h}^{(\tau)} = f_{\text{enc}}(\mathbf{x}^{(\tau)})$ is the encoding of the input and $||\mathbf{h}^{(\tau)}||^{P_i}$ denotes an L_2 pooling operation over the region P_i. The first term asks the model to reconstruct the input from the code and the code to L_1 sparse. The second term encourages temporal coherence but in the pooled space.

4.5 Sparse-Coder for Separating Amplitude and Phase

Cadieu and Olshausen (2012) proposed a model which aims to disentangle appearance and motion by explicitly representing features in terms of an amplitude and a phase. This can be seen as a grouping and reparameterization operation. In a standard sparse coder, each feature is independent. But here we bundle features into groups of two. Each group is now represented in polar coordinates – an amplitude a and phase ϕ, as opposed to the more standard way of representing them as two independent components. In both cases we have two degrees of freedom. The advantage of separating amplitude and phase is that over a short time scale, motion will create smooth changes in amplitude and phase but more abrupt changes if the features were represented independently. The phase helps impose a smooth local topology on the feature space. Therefore, instead of each feature just representing how strongly that feature is present, it now represents that along with where that feature is present with respect to its own local coordinate frame. This separation of what and where makes this model very appealing.

The model can be elegantly expressed as a sparse coder with complex-valued basis functions and complex-valued coefficients. Consider sparse-coding an I-dimensional input signal $\mathbf{x}$. Suppose we choose to have J basis functions and we want the input signal to be a sparse linear combination of these J basis functions weighted by some coefficients. In a standard sparse coder, each basis function would be an I-dimensional vector of weights $\mathbf{w}_j$. However, in this case, each basis function has an amplitude $\mathbf{w}_j$ and phase $\boldsymbol{\theta}_j$, both of which are I-dimensional vectors. Therefore, the weight connecting feature j to input dimension k is $w_{jk}e^{i\theta_{jk}}$. The coefficient corresponding to basis function j is z_j. It has an amplitude a_j and phase ϕ_j and can be represented as the complex number $a_j e^{i\phi_j}$. The generative model of the data is that for each dimension x_k,

$$\begin{aligned} x_k &= Re\left[\sum_j z_j^* w_{kj}\right] + \eta \\ &= \sum_j a_j w_{ij}\cos(\theta_{kj} - \phi_j) + \eta \end{aligned}$$

where $*$ denotes complex conjugate, Re denotes the real part of that complex number and η denotes independent Gaussian noise. Note that this model can further be expanded as-

$$\begin{aligned} x_k &= \sum_j a_j w_{kj}(\cos\phi_j \cos\theta_{kj} + \sin\phi_j \sin\theta_{kj}) + \eta \\ &= \sum_j (a_j \cos\phi_j)(w_{kj}\cos\theta_{kj}) + \sum_j (a_j \sin\phi_j)(w_{kj}\sin\theta_{kj}) + \eta \end{aligned}$$

Therefore, if each feature and weight is reparameterized by projecting into Cartesian coordinates $(r, \theta) \rightarrow (r\cos\theta, r\sin\theta)$, then the model looks similar to a standard sparse-coder. Therefore, till this point, all we have done is make groups of two and reparametrize in polar coordinates, without changing the model at all. What makes this model interesting and different from standard sparse coding is that we can now impose priors separately on the phase and amplitude. When projected

into Cartesian coordinates, this would induce a dependence within the two features in each group which is not present in standard sparse coding. In particular, we can choose to have a uniform prior on phase and enforce sparsity and slowness on the amplitude.

$$P(a_j^{(t)}|a_j^{(t-1)}) \propto \exp\left(-\lambda Sp(a_j^{(t)}) - \beta Sl(a_j^{(t)}, a_j^{(t-1)})\right),$$
$$Sp(a_j^{(t)}) = \log\left(1 + \left(\frac{a_j^{(t)}}{\sigma}\right)^2\right),$$
$$Sl(a_j^{(t)}, a_j^{(t-1)}) = \left(a_j^{(t)} - a_j^{(t-1)}\right)^2.$$

Sp is the sparsity prior on the amplitudes (Cauchy distribution). Sl is the slowness prior which enforces temporal coherence. λ and β control the importance of these priors.

In this model, Cadieu and Olshausen (2012) use a one-dimensional phase ϕ. There is no reason why this cannot be extended to higher dimensional phases. This would provide an expressive feature representation where the phase can represent the pose of a feature. Essentially, we are providing a local coordinate system to each feature. Objects can then be detected by finding agreement in the relative pose of their parts. This is a very promising research direction that I am currently exploring.

4.6 Contrastive Backpropagation

Feed-forward neural networks can be used to define generative probabilistic models. This can be done simply by defining a real-valued smooth function E of the activations in a network and calling it an energy. Any function will do, as long as it can be well-defined energy, meaning that it should be lower bounded and smooth. This energy function E can be used to define a probability density function on the space of data vectors $\mathbf{x}$

$$p(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{\sum_{\mathbf{x}'} e^{-E(\mathbf{x}')}}.$$

One natural way to think of $E(\mathbf{x})$ is as a measure of how much $\mathbf{x}$ violates the constraints that are present in the system that we are modeling. If $\mathbf{x}$ satisfies all the constraints, it has a low energy and a high probability. Contrastive Backpropagation (Hinton et al., 2006) is an extremely general method for training such models. The main idea is that if we could sample from this distribution, we can find examples that the model thinks should have high probability. Then during training, we can change the model's parameters to lower the probability of those examples and raise the probability of the true data. These two effects will cancel out when the things we sample from the model have the same distribution as the true data, which is when we stop training. In Contrastive Backpropagation, samples are obtained by running a Markov chain, starting from samples from the data distribution. In principle, if the Markov chain is run for a long time, the chain should forget its initial position and end up in a true sample from the distribution. The mixing of the Markov chain is often slow for powerful models and in order to speed up the process, Hybrid Monte-Carlo (HMC) sampling is used which tries to simulate the dynamics of a particle moving on the energy surface. In practice, only a

few steps of this Markov chain are run (as many as computationally feasible) and it is argued that that since we start from the data distribution, a few steps are often enough to detect a systematic change in the distribution which is enough to provide the right gradient direction.

Although Contrastive Backpropagation does not place any serious restrictions on the input space or kind of network or energy function, it is currently not a popular method for unsupervised learning. This is primarily because learning requires sampling from the energy landscape of the model defined on the high-dimensional input space, which is very slow in spite of using HMC. The particles moving on the energy landscape are prone to getting stuck and never reaching large regions of the domain, leaving them with high probabilities. This approach was shown to work by Mnih and Hinton (2005) for modeling the coordinates of a moving arm containing several segments. The constraints in this case are that the length of each arm segment is fixed and the coordinates must respect this. Contrastive Backpropagation can be expected to work well in this case because the constraint is a very sharp defining characteristic of this system.

Hinton et al. (2006) mention that this can be applied to modeling temporal sequences by feeding them into a recurrent neural net and defining a smooth energy function on the hidden activities of the network. While any smooth energy function will do, we could enforce desirable properties by designing more appropriate energy functions. For example, if we wish to encourage temporal coherence, we can penalize the states for differing too much over successive time steps. If we want the features in the RNN to be different, we can encourage decorrelation of the outputs of the hidden units. We can have a lot of creativity in designing the energy function. This offers a promising direction for future research. Although the computational disadvantages of this method are still daunting, they can now be overcome to some extent by just using faster computers.

5 Objective Functions for Slowness

In this section, we review models that try to explicitly quantify slowness. Slowness is measured by some global objective function which is optimized to train the model.

5.1 Slow Feature Analysis

Wiskott and Sejnowski (2002) proposed Slow Feature Analysis (SFA) as a method of finding temporally coherent features. For any chosen J, SFA tries to find the J-dimensional projection of the input signal that minimizes the mean squared rate of change of the projected signal over time, under the constraint that the J dimensions have zero mean and identity covariance over time. More precisely, the problem can be stated as follows. Suppose $\mathbf{x}^{(t)}$ is an I-dimensional time-varying input signal. We would like to find J functions, $\{g_1(\mathbf{x}), \ldots, g_J(\mathbf{x})\}$ such that if we apply those functions on the input, the resulting J-dimensional output sequence $\mathbf{y}^{(t)}$, with $y_j^{(t)} = g_j(\mathbf{x}^{(t)})$ has the property that-

$$\langle \left(\Delta y_j^{(t)}\right)^2 \rangle \text{ is minimal}$$

subject to the constraints that for each j

$$\begin{aligned} \langle y_j \rangle &= 0, && \text{(zero mean)} \\ \langle y_j^2 \rangle &= 1, && \text{(unit variance)} \\ \langle y_j y_{j'} \rangle &= 0 \quad \forall j' < j, && \text{(decorrelation)} \end{aligned}$$

where $\langle \bullet \rangle$ indicates the average over time and $\Delta y_j^{(t)} = y_j^{(t)} - y_j^{(t-1)}$. This objective says that changes in $\mathbf{y}^{(t)}$ over time must be small. Being constant would obviously minimize this objective if there were no constraints. Therefore, to prevent collapse, we demand that each dimension of $\mathbf{y}^t$ must have zero mean and unit variance over time. Additionally, we would like the J functions to be different. Therefore, we also demand that each successive feature be uncorrelated with all previous ones. This also defines a natural ordering of the features. Each successive feature, must obey an additional constraint that it should be uncorrelated with the previous ones. Therefore, the first feature is the slowest varying and each successive one becomes less slow.

In general, this optimization problem is hard to solve, but it can solved if the g's are constrained to be of particular forms. For example, if the g's are all linear, then they can be found by computing the temporal deltas of the input signal, whitening them, doing PCA and then picking the J lowest eigen values. Instead of applying this directly to the input signal, we could also first apply a fixed non-linear projection on each frame of the signal and then do slow feature analysis on the projected signal. This would map the signal to a higher dimensional and hopefully richer space where interesting slow features can be found. This is the formulation proposed by Wiskott and Sejnowski (2002) and explained through multiple examples. The g's become non-linear but the non-linear part is fixed, not learned.

This brings us to the main drawback of this approach which is that we need to engineer the non-linear projection to expose interesting features. It is not clear how to do this for a general input sequence. The SFA part is just applying an affine transformation to this engineered space to find the directions along which the data is changing the slowest. Arguably, most of the heavy lifting needs to be done by the non-linear projection. The situation is somewhat analogous to that faced in designing good kernel functions for use in Support Vector Machines. Another drawback, as mentioned by the authors, is that the computational complexity becomes prohibitive as size of the non-linear space (say, D) increases because we need to compute a covariance matrix of size $D \times D$ and apply PCA to it which takes $O(D^3)$ time. Moreover, this is a batch method which makes it hard to apply to very large datasets.

Wiskott and Sejnowski (2002) also show how SFA modules can be stacked to create hierarchical networks. Just like in a multilayer neural network, each subsequent SFA layer looks at the previous layer's features and optionally increases its field of view (similar to convolutional networks). If we think of the input as a fast-changing sequence, each subsequent layer can be seen as a progressively slowed down higher-level representation of the layer below.

The authors highlight an important aspect of the SFA algorithm that it is "simple and guaranteed to find the optimal solution in one shot", as opposed to the other methods that we have seen till now that involve some form of iterative learning which is prone to getting stuck in local optima or

on plateaus. Also, multiple decorrelated slow features can be extracted simultaneously. The way decorrelation has been achieved, if at all, in all other models that we have seen is through some form of competition, for example, in the case of Becker (1996) it was achieved by normalizing the post-synaptic activations over a bunch of neurons. The authors here argue that this not really learning different features, but a single one-of-K encoded feature that exhibits K states. This method has been successfully applied to a number of applications such as estimating driving forces of a dynamical system (Wiskott, 2003), blind source separation (Sprekeler et al., 2014) and as feature extractor for reinforcement learning (Legenstein et al., 2010).

5.2 Probabilistic Interpretation of SFA

SFA was introduced as the solution to an optimization problem and was not necessarily motivated from a generative probabilistic standpoint. Turner and Sahani (2007) showed that SFA can be seen as inference for a linear Gaussian dynamical system under certain constraints. The prior over $\mathbf{y}^{(t)}$ is

$$\begin{aligned} p(\mathbf{y}^{(t)}|\mathbf{y}^{(t-1)}, \boldsymbol{\lambda}, \boldsymbol{\sigma}) &= \prod_{j=1}^{J} p(y_j^{(t)}|y_j^{(t-1)}, \lambda_j, \sigma_j^2), \\ p(y_j^{(t)}|y_j^{(t-1)}, \lambda_j, \sigma_j^2) &= \mathcal{N}(\lambda_j y_j^{(t-1)}, \sigma_j^2), \\ p(y_j^{(1)}|\sigma_{j,1}^2) &= \mathcal{N}(0, \sigma_{j,1}^2). \end{aligned}$$

The conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is

$$p(\mathbf{x}^{(t)}|\mathbf{y}^{(t)}, \mathbf{W}, \sigma_x) = \mathcal{N}(\mathbf{W}^{-1}\mathbf{y}^{(t)}, \sigma_x^2\mathbf{I}).$$

If we set $\sigma_{j,1}^2 = 1$, $\sigma_x \to 0$, $\sigma_j^2 = 1 - \lambda_j^2$ and $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_J \to 0$, then maximum likelihood learning in this model would be same as the optimization problem being solved in SFA. Therefore, SFA is equivalent to a linear Gaussian dynamical system with a simple component-wise diffusion process in the latent space. This analysis makes explicit some of the assumptions being made implicitly in the original SFA formulation. It also makes it easy to relax certain conditions or add more to create different SFA variants.

5.3 Relationship between SFA and ICA

Blaschke et al. (2006) try to understand the relationship between SFA and ICA. Their main result is that linear SFA is formally equivalent to second-order ICA with time delay one. We have looked at time-delayed ICA in Eq. 3. SFA is equivalent to setting $\tau = 1$. This helps relate two different notions of temporal coherence – one coming from slowness of change in a representation over time and the other coming from statistical independence of different components of a representation over time. If independence in measured through correlation, as is the case in second-order ICA, the two notions become the same.

5.4 Contrastive Hinge-loss Objectives

Mobahi et al. (2009) proposed a loss function that encourages coherence by trying to pull temporally consecutive frames close and push other pairs of frames farther than some margin, where the distances are measured in the L_1 sense in some learned feature space.

$$L_{\text{coh}}(\mathbf{x}_1, \mathbf{x}_2, \theta) = \begin{cases} ||f_\theta(\mathbf{x}_1) - f_\theta(\mathbf{x}_2)||_1, & \text{if } \mathbf{x}_1, \mathbf{x}_2 \text{ are consecutive,} \\ \max(0, \delta - ||f_\theta(\mathbf{x}_1) - f_\theta(\mathbf{x}_2)||_1), & \text{otherwise.} \end{cases}$$

Here f_θ is a convolutional neural network and θ denotes all its parameters. This coherence objective is jointly minimized with a supervised classification loss to learn the parameters θ. The feature space is the penultimate layer of the network.

In a recent preprint, Wang and Gupta (2015) propose a similar model. They optimize the following loss function

$$L(\theta) = \sum_{\mathbf{x} \in \mathcal{D}} \max\left(0, \delta + D_\theta(\mathbf{x}, \mathbf{x}^+) - D_\theta(\mathbf{x}, \mathbf{x}^-)\right),$$

where

$$D_\theta(\mathbf{x}_1, \mathbf{x}_2) = 1 - \frac{f_\theta(\mathbf{x}_1) \cdot f_\theta(\mathbf{x}_2)}{||f_\theta(\mathbf{x}_1)||||f_\theta(\mathbf{x}_2)||}$$

is the cosine distance measure. The loss function operates on triplets $(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-)$ where $\mathbf{x}$ is some image patch, $\mathbf{x}^+$ is an image patch found after tracking $\mathbf{x}$ for 30 frames and $\mathbf{x}^-$ is a hard negative example. The motivation is similar to the previous model, in the sense that this model also pulls together image patches that we believe to be semantically close and pushes other image patches away. The measure of the distances is slightly different, as is the fact that this is applied to tracked image patches and not entire frames. In another recent preprint, Ramanathan et al. (2015) learn temporal embeddings by extending the Skip-gram model that was proposed for learning word embeddings (Mikolov et al., 2013). Given an unlabelled set of videos V, they minimize the following objective

$$L(\theta) = \sum_{\mathbf{v} \in V} \sum_{\mathbf{x} \in \mathbf{v}} \sum_{\mathbf{x}^- \in \overline{N(\mathbf{x})}} \max\left(0, 1 - (f_\theta(\mathbf{x}) - f_\theta(\mathbf{x}^-)) \cdot \mathbf{h}_{\theta,\mathbf{x}}\right),$$

where $\mathbf{x}$ is a frame in video $\mathbf{v}$, $\mathbf{x}^-$ is a frame outside the neighbourhood of $\mathbf{x}$ (from the same or different video), f_θ is a deep neural net and $\mathbf{h}_{\theta,\mathbf{x}}$ is the context of frame $\mathbf{x}$. The authors compare different kinds of context but what they found to work best was the average embedding of frames in the neighbourhood of $\mathbf{x}$

$$\mathbf{h}_{\theta,\mathbf{x}} = \frac{1}{|N(\mathbf{x})|} \sum_{\mathbf{x}' \in N(\mathbf{x})} f_\theta(\mathbf{x}').$$

The idea is that similarity between an image and its context $f_\theta(\mathbf{x}) \cdot \mathbf{h}_{\theta,\mathbf{x}}$ must be greater than that between a far away image and the context $f_\theta(\mathbf{x}^-) \cdot \mathbf{h}_{\theta,\mathbf{x}}$ by some margin. In other words, the context must predict the frame better than it predicts other frames.

All these models essentially try to enforce coherence in the learned representation by demanding that nearby frames have similar representations and prevent collapse by contrasting with far away frames. The main drawback of these methods is that these objective functions are often easy to

optimize without learning good high-level representations. For example, if we want adjacent frames to be close, low-level translation invariance can give us that by learning Gabors and pooling. There is not much incentive to learn good high-level features, the kind of which are necessary for supervised learning on ImageNet.

6 Solving Vision Tasks

In this section, we review computer vision tasks that can serve as objectives from learning visual representations from unlabelled videos.

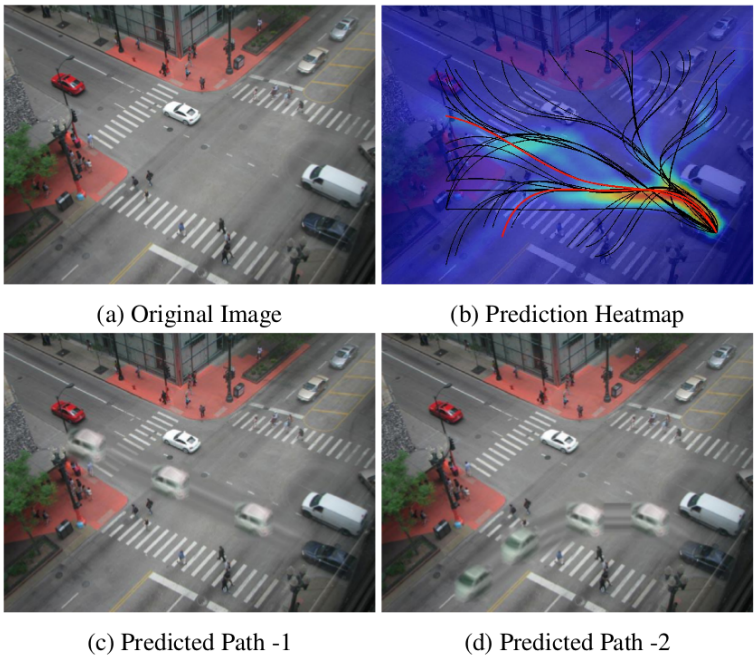
6.1 Tracking

Figure 5: Figure reproduced from Stavens and Thrun (2010). Different views of the same object through time obtained by tracking.

Stavens and Thrun (2010) proposed an unsupervised model to learn visual features using videos. They first track objects using low-level features (Harris corner features + optical flow). By doing this they collect sets of patches where all patches in the same set are of the same object being tracked through a video. Fig. 5 shows four such sets. These sets are used as ground truth for training high-level features using a contrastive loss function which pulls all members of the same set close together in high-level feature space and pushes members of different sets farther away. This is similar in spirit to Mobahi et al. (2009) but in that case the high-level features were computed on entire frames. In this case, objects are first tracked and then used to extract high-level features. The main idea here is that tracking algorithms that use low-level cues like optical flow can be used to track objects and then high-level features can be learned from the resulting data. These high-level features can now help learn better trackers and that can track objects across more variations of pose, occlusion and clutter. The data from that can be used to learn even more high-level representations. Therefore, we can bootstrap the learning of high-level visual representations from low-level ones using unlabelled videos. In this way the goal of tracking can help learn visual representations from unlabelled videos.

6.2 Predicting Future Motion and Appearance

Walker et al. (2014) propose an algorithm for predicting the trajectory and appearance of moving objects in a visual scene. Their algorithm takes a static image as input and outputs a probability distribution over which things will move, what their trajectories will be and what their appearance will be. The model is trained using only unlabelled data. An example output from the model is shown in Fig. 6. This line of research is motivated by the fact that when humans look at a static



(a) Original Image (b) Prediction Heatmap

(c) Predicted Path -1 (d) Predicted Path -2

Figure 6: Results from Walker et al. (2014) (Reproduced from their paper).

scene, they not only understand the objects and their pose but also anticipate the state of the scene in the near future. This mental prediction helps humans in planning their actions, providing a strong evolutionary advantage. Therefore, human brains must be representing visual information in a way that makes it easy to predict the future. Therefore, being able to predict the future might be a reasonable objective function for learning visual representations.

As a representative example of a computer vision approach, we can look at the algorithm proposed by Walker et al. (2014). They learn three things -

- A representation of object appearance. This is a one-of-K encoding obtained by clustering mid-level patches. This essentially clusters image patches into K clusters based on mid-level features (Singh et al., 2012). There might be different clusters for the same object, for example one for frontal views of cars and another for the side view of cars.

- A transition matrix that contains the probability of transitioning from one cluster to another. This is estimated by simply counting how often these transitions occur in data.

- A reward function that uses the features and context of an image region to determine how likely it is to contain each cluster (for example, road-like regions are likely to contain things that look like cars, but sidewalk regions are unlikely to do so). This is again estimated by counting how often a particular kind of region gets occupied by a particular kind of image patch.

These components are learned piece-wise and put together to create an algorithm that given a static image, goes to each patch in that image, maps it to the nearest cluster and uses the transition matrix and reward function to estimate a distribution over how this patch will move in the future and what its appearance will look like. Even though this model learns a very simple representation and

uses very simple learning schemes, it is interesting in the way it lays out the different factors involved in predicting trajectories and stitches them into a model. The model can be extended by replacing the one-of-K image representation with a distributed one, the HMM by a temporal sequence model which uses a distributed state, going to each patch by an attention model, etc.

The focus in most of the work in this area has been in accurately modeling the trajectories and scene affordances given some fixed hand-engineered visual representation. For example, Kitani et al. (2012) represent objects as points and formulate the problem in terms of reinforcement learning in POMDPs focusing on the geometry of the scene; Yuen and Torralba (2010) model optical flow trajectories; Walker et al. (2014) use a dictionary of mid-level patches. Recently, Vondrick et al. (2015) used features from deep convolutional networks (Krizhevsky et al., 2012; Szegedy et al., 2014) trained on the ImageNet dataset (Deng et al., 2009) as targets for future prediction from static images. However, in all these cases the representation is first fixed and then the task is solved. An interesting future research direction is making these systems end-to-end by training the feature extraction model along with the task being solved.

6.3 Learning to Play Games

Learning good visual representations is a pre-requisite for playing games that require the player to take decisions based on visual input. Therefore visual feature learning can be incorporated into a Reinforcement Learning setting. A very ambitious and fruitful line of research is in this area is being explored. Mnih et al. (2015) showed that an agent can be trained to play simple Atari games by simply looking at the pixels and getting the game score as input. This is different from the completely unsupervised setting where we just have video frames as input. However, the supervision here comes from the environment and once we can design realistic environments, there is no more human intervention needed. Being able to play other games that require understanding 3D geometry, pose and motion from realistic images can help learn representations that might transfer well into the real world.

7 Conclusion

In this paper we reviewed models and tasks that have been used to learn visual representations from videos. We tried to understand how the notion of temporal coherence is captured by these models using different realizations of statistical independence, slowness and sparsity.

Arguably, one of the main drawback of most machine learning models in this area is that they are too simple and general. A lot of advances may be possible by using just the right amount of domain knowledge. For instance the biggest advance in recent years in machine learning applied to vision came from convolutional neural nets which are structured to incorporate the local nature of visual information and the fact that statistics of image patches are invariant to position. Giving even more structure to the models, for example, by using richer neurons (e.g. including phase instead of just activations), incorporating attention and tracking as primitives and building models to explain occlusion are all fruitful research directions.

References

S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In *Advances in Neural Information Processing Systems*, pages 757–763. MIT Press, 1996.

Suzanna Becker. Learning to categorize objects using temporal coherence. In *Advances in Neural Information Processing Systems 5, NIPS*, pages 361–368, 1992.

Suzanna Becker. Learning temporally persistent hierarchical representations. In *NIPS*, pages 824–830. MIT Press, 1996.

Suzanna Becker and Geoffrey E. Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

Anthony J. Bell and Terrence J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

Tobias Blaschke, Pietro Berkes, and Laurenz Wiskott. What is the relation between slow feature analysis and independent component analysis? *Neural Computation*, 18(10):2495–2508, 2006.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML*, 2012.

Charles F. Cadieu and Bruno A. Olshausen. Learning intermediate-level representations of form and motion from natural movies. *Neural Computation*, 24(4):827–866, 2012.

J. F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings-F*, 140(6):362–370, 1993.

Pierre Comon. Independent component analysis, a new concept? *Signal Process.*, 36(3):287–314, April 1994.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Peter Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised feature learning from temporal data. *CoRR*, abs/1504.02518, 2015.

Chris Häusler, Alex Susemihl, and Martin P. Nawrot. Natural image sequences constrain dynamic receptive fields and imply a sparse code. *Brain Research*, 1536(0):53 – 67, 2013.

Donald O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, 1949.

J. Herault and C. Jutten. Space or time adaptive signal processing by neural network models. *AIP Conference Proceedings*, 151(1):206–211, 1986.

Geoffrey E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(13):185 – 234, 1989.

Geoffrey E. Hinton, Simon Osindero, Max Welling, and Yee Whye Teh. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive Science*, 30(4):725–731, 2006.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

A. Hyvärinen and E. Oja. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

Aapo Hyvärinen and Patrik Hoyer. Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12(7):1705–1720, 2000.

Aapo Hyvärinen and Erkki Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

Aapo Hyvärinen, Jarmo Hurri, and Jaakko V"ayrynen. A unifying framework for natural image statistics: spatiotemporal activity bubbles. pages 801–806, 2004.

Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231, Jan 2013.

Christian Jutten and Jeanny Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1 – 10, 1991.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

Kris M. Kitani, Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV*, pages 201–214, 2012.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

Q. V. Le, W. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011.

Robert A. Legenstein, Niko Wilbert, and Laurenz Wiskott. Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 2010.

Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

Roland Memisevic. Gradient-based learning of higher-order image features. In *ICCV*, pages 1591–1598. IEEE, 2011.

Roland Memisevic. Learning to relate images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1829–1846, 2013.

Roland Memisevic and Geoffrey E. Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, 22(6):1473–1492, 2010.

Vincent Michalski, Roland Memisevic, and Kishore Konda. Modeling deep temporal dependencies with recurrent grammar cells. In *Advances in Neural Information Processing Systems 27*, pages 1925–1933. Curran Associates, Inc., 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

G Mitchison. Removing time variation with the anti-hebbian differential synapse. *Neural Computation*, 3(3):312–320, Sept 1991.

Andriy Mnih and Geoffrey Hinton. Learning nonlinear constraints with contrastive backpropagation. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 1302–1307, 2005.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.

Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 737–744, New York, NY, USA, 2009. ACM.

L. Molgedey and H. G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Physical Review Letters*, 72:3634–3637, 1994.

B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

Vignesh Ramanathan, Kevin Tang, Greg Mori, and Li Fei-Fei. Learning temporal embeddings for complex video analysis. *CoRR*, abs/1505.00315, 2015.

Marc'Aurelio Ranzato, Arthur Szlam, Joan Bruna, Michaël Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. *CoRR*, abs/1412.6604, 2014.

Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27: 379–423, 623–656, 1948.

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014.

Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part II*, ECCV'12, pages 73–86, Berlin, Heidelberg, 2012. Springer-Verlag.

Henning Sprekeler, Tiziano Zito, and Laurenz Wiskott. An extension of slow feature analysis for nonlinear blind source separation. *Journal of Machine Learning Research*, 15:921–947, 2014.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015.

David Stavens and Sebastian Thrun. Unsupervised learning of invariant features using video. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 1649–1656, 2010.

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

Ilya Sutskever, Geoffrey E. Hinton, and Graham W. Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems 21*, pages 1601–1608. Curran Associates, Inc., 2009.

Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

Graham W. Taylor and Geoffrey E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1025–1032, New York, NY, USA, 2009. ACM.

Graham W. Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, pages 140–153, Berlin, Heidelberg, 2010. Springer-Verlag.

Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

Richard E. Turner and Maneesh Sahani. A maximum-likelihood interpretation for slow feature analysis. *Neural Computation*, 19(4):1022–1038, 2007.

Paul A. Viola, Nicol N. Schraudolph, and Terrence J. Sejnowski. Empirical entropy manipulation for real-world problems. In *NIPS*. MIT Press, 1995.

Carl Vondrick, Aditya Khosla, Hamed Pirsiavash, Tomasz Malisiewicz, and Antonio Torralba. Visualizing object detection features. *CoRR*, abs/1502.05461, 2015.

Jacob Walker, Abhinav Gupta, and Martial Hebert. Patch to the future: Unsupervised visual prediction. In *Computer Vision and Pattern Recognition*, 2014.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. *CoRR*, abs/1505.00687, 2015.

B. Willmore and D. J. Tolhurst. Characterizing the sparseness of neural codes. *Network*, 12:255–270, 2001.

Laurenz Wiskott. Estimating driving forces of nonstationary time series with slow feature analysis. *CoRR*, abs/cond-mat/0312317, 2003.

Laurenz Wiskott and Terrence J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

Jenny Yuen and Antonio Torralba. A data-driven approach for event prediction. In *Proceedings of the 11th European Conference on Computer Vision: Part II*, ECCV'10, pages 707–720, Berlin, Heidelberg, 2010. Springer-Verlag.

Richard S. Zemel and Geoffrey E. Hinton. Discovering viewpoint-invariant relationships that characterize objects. In *Advances in Neural Information Processing Systems 3*, pages 299–305. Morgan-Kaufmann, 1991.

A. Ziehe and K.-R. Müller. TDSEP – an efficient algorithm for blind separation using time structure. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proc. of the 8th International Conference on Artificial Neural Networks, ICANN'98*, Perspectives in Neural Computing, pages 675 – 680, Berlin, 1998. Springer Verlag.