# Deep Learning of Representations

Yoshua Bengio and Aaron Courville
Dept. IRO, Université de Montréal

January 10th, 2011 - DRAFT

**Abstract**

Unsupervised learning of representations has been found useful in many applications and benefits from several advantages, e.g., where there are many unlabeled exemples and few labeled ones (semi-supervised learning), or where the unlabeled or labeled examples are from a distribution different but related to the one of interest (self-taught learning, multi-task learning, and domain adaptation). Some of these algorithms have successfully been used to learn a hierarchy of features, i.e., to build a deep architecture, either as initialization for a supervised predictor, or as a generative model. Deep learning algorithms can yield representations that are more abstract and better disentangle the hidden factors of variation underlying the unknown generating distribution, i.e., to capture invariances and discover non-local structure in that distribution. This chapter reviews the main motivations and ideas behind deep learning algorithms and their representation-learning components, as well as recent results in this area, and proposes a vision of challenges and hopes on the road ahead.

# 1 Introduction

WHAT THIS CHAPTER IS ABOUT

Follow-up (?)

**Why learning representations?**

Machine learning is about capturing dependencies between random variables and discovering the salient but hidden structure in an unknown distribution (conditional or not), from examples. Current machine learning algorithms tend to be

very dependent on the choice of data representation on which they are applied: with the right representation, almost every learning problem becomes very easy. This is particularly true of non-parametric and kernel-based algorithms, which have been very successful in recent years (?). It is therefore very tempting to ask the questions: can we learn better representations? what makes a good representation? The probabilistic and geometric points of view also help us to see the importance of a good representation. The data density may concentrate in a complicated region in raw input space, but a linear or non-linear change in coordinates could make the learning task much easier, e.g., extracting uncorrelated (PCA) or independent (ICA) factors allows much more compact descriptions of the raw input space density, and this is even more true if we consider the kind of non-linear transformations captured by manifold learning algorithms (?).

Imagine a probabilistic graphical model (?) in which we introduce latent variables which correspond to the true explanatory factors of the observed data. It is very likely that answering questions and learning dependencies in that space is going to be much easier. To make things concrete, imagine a graphical model for images that, given an image of a child throwing a ball to her friend in a park, would have latent variables turning "ON", whose values would directly correspond to these events (e.g., a variable could encode the fact that a person is located at a particular position in a particular pose, another that this person is a child, that she is a female, another that she is in the action of thowing a ball, located at such position, etc.). Starting from this representation, learning to answer questions about the scene would be very easy indeed. A simple linear classifier trained from as few as one or a few examples would probably do the job. Indeed, this is the kind of surprisingly fast supervised learning that humans perform *in settings that are familiar to them*, i.e., in which they had the chance to collect many examples (without needing a label associated to task on which they are finally tested, i.e., this is really the semi-supervised setting in machine learning). In fact, with linguistic associations of these latent variables to words and sentences, humans can answer questions about a new task with zero labeled training examples for the new task, i.e., they are simply doing inference: the new task is specified in a language that is already well connected to these latent variables, and one can thus generalize to new tasks with zero examples (?), if one has set or learned a good representation for tasks themselves.

So that is the objective we are setting: **learning representations that capture the explanatory factors of variation, and help to disentangle them.**

**Why distributed representations?**

A distributed representation is one which has many components or attributes,

and such that many of them can be independently active or varied simultaneously. Hence the number of possible objects that can be represented can grow up to exponentially with the number of attributes that can be simultaneously active. In a dense distributed representation, all of the components can be active simultaneously, whereas in a *sparse representation*, only a few can be (while the others are 0). The other end of the spectrum is a *local representation*, such as a hard clustering, in which each input object is represented by the activation of a single cluster (the one to which the object belongs), i.e., an integer (ranging from 1 to the number of clusters). It is clear that even with a discrete representation, if there are many factors of variation, representing the input object by a set of $N$ attributes (each taking one out of several values, at least 2) provides for a much richer representation, with which up to $2^N$ different objects can be distinguished. With a sparse representation with $k$ active attributes, the number of objects that can be distinguished is on the order of $N$ choose $k$, which also grows faster than exponential in $k$.

**Why deep**?

How should our learned representation by parametrized? A **deep architecture** is one in which there are *multiple levels of representation*, with higher levels built on top of lower levels (the lowest being the original raw input), and higher levels representing more abstract concepts defined in terms of less abstract ones from lower levels. There are several motivations for deep architectures:

- *Brain inspiration*: several areas of the brain, especially those better understood such as visual cortex and auditory cortex, are organized as a deep architecture, with each brain area associated with a level of representation (?).

- *Computational complexity*: as discussed in (?), some computational complexity results suggest that some functions which can be represented compactly with a deep architecture would require an exponential number of components if represented with a shallow (e.g. 2-level) architecture. Of course, depending on the task and the types of computation performed at each layer, the sufficient depth will vary. It is therefore important to have algorithms that can accomodate different depths and choose depth empirically.

- *Statistical efficiency and sharing of statistical strength*. First, if deeper architectures can be more efficient in terms of number of computational units (to represent the same function), that in principle means that the number of parameters than need to be estimated is smaller, which gives rise to greater

statistical efficiency. Another way to see this is to consider the sharing of statistical strength that occurs when different components of an architecture are *re-used* for different purposes (e.g., in the computation for different outputs, or different tasks, or in the computation for different intermediate features). Since the parameters of a component are used for different purposes, they share statistical strength among the different examples (or parts of examples) that rely on these parameters. This is similar and related to the sharing of statistical strength that occurs in distributed representations. For example, if the parameters of one hidden unit of an RBM are "used" for many examples (because that unit turns on for many examples), then there is more information available to estimate those parameters. When a new configuration of the input is presented, it may not correspond to any of those seen in the training set, but its "components" (possibly represented at a higher level of abstraction in intermediate representations) may have been seen previously.

- *Cognitive arguments and engineering arguments*. Humans very often organize ideas and concepts in a modular way, and at multiple levels. Concepts at one level of abstraction are defined in terms of lower-level concepts (for example it is possible to write a dictionary whose definitions depend only of a very small number of core words). Similarly, that is also how problems are solved and systems built by engineers: they typically construct a chain or a graph of processing modules, with the output of one feeding the inputs of another. The inputs and outputs of these modules are intermediate representations of the raw signals that enter the system. They are designed thanks to human ingenuity. We would like to add to human ingenuity the option to *learn such decompositions and intermediate representations*.

- *Sharing of statistical strength for multi-task learning, semi-supervised learning, self-taught learning, and out-of-domain generalization.* Sharing of statistical strength is a core idea behind many advances in machine learning. Components and parameters are shared across tasks in the case of multi-task learning, and deep architectures are particularly well suited for multi-task learning (?). Similarly semi-supervised learning exploits statistical sharing between the tasks of learning the input distribution $P(X)$ and learning the conditional distribution $P(Y|X)$. Because deep learning algorithms often rely heavily on unsupervised learning, they are well suited to exploit this particular form of statistical sharing. A very related form of sharing

occurs in self-taught learning (?), whereby we consider unlabeled training data from $P(X|Y)$ for a set of classes $Y$'s but really care about generalizing to tasks $P(Y|X)$ for a different set of $Y$'s. Recent work showed that deep learners benefit more from the self-taught learning and multi-task learning frameworks than shallow learners (?) This is also a form of out-of-domain generalization, for which deep learners are also well suited, as shown in (?) for pattern recognition and in (?) for natural language processing (sentiment analysis).

**Why semi-supervised or unsupervised learning**?

An important prior exploited in many deep learning algorithms (such as those based on greedy layer-wise pre-training, detailed below, sec. **??**) is the following: *representations that are useful for capturing $P(X)$ can be useful (at least in part, or as initialization) for capturing $P(Y|X)$.* As discussed above, this is beneficial as a statistical sharing strategy, and especially so because $X$ is usually very high-dimensional and rich, compared to $Y$, i.e., it can contain very detailed structure that can be relevant to predicting $Y$ given $X$. See (?) for a discussion of the advantages brought by this prior, and a comprehensive set of experiments showing how it helps not only as a regularizer, but also to find better training error when the training set is large, i.e., *to find better local minima of the generalization error (as a function of the parameters).* This is an important consideration because deep learners have a highly non-convex training criterion (whether it be supervised or unsupervised) and the greedy layer-wise initialization strategy, based on unsupervised pre-training, can make a huge difference.

More generally, the very task of learning representations with the objective of sharing statistical strenghts across tasks, domains, etc. begs for unsupervised learning, modeling for all the variables observed (including the $Y$'s) and good representations for their joint distribution. Consider an agent immersed in a stream of observations of $X$'s and $Y$'s, where the *set* of values that $Y$ can take is non-stationary, i.e., new classes can appear on which we will later want to make predictions. Unsupervised learning is a way to collect as much information as possible ahead of time about all those observations, so as to later be in the best possible position in order to respond to new requests, possibly from very few labels associated with a new class $Y$. Unsupervised learning is what ultimately need if we consider multi-task / semi-supervised / self-taught learning and the number of possible tasks or classes becomes very large or unbounded.

# 2 Deep Learning of Representations: A Review and Recent Trends

## 2.1 Greedy Layerwise Pre-Training of Unsupervised and Supervised Models

The following basic recipe was introduced in 2006 (?, ?, ?, ?):

1. Let $h_0(x) = x$ be the lowest-level representation of the data, given by the observed raw input $x$.

2. For $\ell = 1$ to $L$
   Train an unsupervised learning model taking as observed data the training examples represented at level $\ell$, and producing after training representations $h_\ell(x) = R_\ell(h_{\ell-1}(x))$ at the next level, based on the representation $h_{\ell-1}(x)$ from the previous level.

From this point on, several variants have been explored in the literature. For supervised learning with fine-tuning (the most common variant (?, ?, ?)):

3 Initialize a supervised predictor whose first stage is the parametrized representation function $h_L(x)$, followed by a linear or non-linear predictor as the second stage.

4 Fine-tune the supervised predictor with respect to a supervised training criterion, based on a labeled training set of $(x, y)$ pairs, and optimizing the parameters in both the representation stage and the predictor stage.

Another supervised variant involves using all the levels of representation as input to the predictor, keeping the representation stage fixed, and optimizing only the predictor parameters (?, ?)

3 Train a supervised learner taking as input $(h_k(x), h_1(x), \ldots, h_L(x))$ for some choice of $0 \leq k \leq L$, using a labeled training set of $(x, y)$ pairs.

Finally, there is a common unsupervised variant, e.g. for training deep auto-encoders (?) or a Deep Boltzmann Machine (?):

3 Initialize an unsupervised model of $x$ based on the parameters of all the stages.

4 Fine-tune the unsupervised model predictor with respect to a unsupervised training criterion, based on the training set of examples $x$.

## 2.2 Undirected Graphical Models and Boltzmann Machines

The first unsupervised learning algorithm (?, ?) that has been proposed for training each level of the above algorithm (step 2) is based on a Restricted Boltzmann Machine (?), which is an undirected graphical model that is a particular form of Boltzmann Machine (?). An undirected graphical model for observed variable $x$ based on latent variable $h$ is specified by an *energy function* $\text{Energy}(x, h)$:

$$P(x, h) = \frac{e^{-\text{Energy}(x,h)}}{Z}$$

where $Z$ is a normalization constant called the partition function. A Boltzmann machine is one where $\text{Energy}(x, h)$ is a second-order polynomial in $(x, h)$, e.g.,

$$\text{Energy}(x, h) = h'Wx + h'Uh + x'Vx + b'h + c'x$$

and in general both $x$ and $h$ are considered to be binary vectors, which makes $Z$ untractable except when both $x$ and $h$ have very few components. The coefficients $\theta = (W, U, V, b, c)$ of that second-order are the parameters of the model. Given an observed $x$, the inference $P(h|x)$ is generally intractable but can be estimated by sampling from a Monte-Carlo Markov Chain (MCMC), e.g. by Gibbs sampling, or using loopy belief approximations, variational or mean-field approximations. Even though computing the energy is easy, marginalizing over $h$ in order to compute the likelihood $P(x)$ is generally intractable, so that the exact log-likelihood gradient is also intractable. However, several algorithms have been proposed in recent years to estimate the gradient, all based on the following decomposition into the so-called "positive phase part" ($x$ is fixed to the observed value, the term tends to decrease the associated energies) and "negative phase part" (both $x$ and $h$ are sampled according to $P$, and the term tends to increase their energy):

$$\frac{\partial}{\partial \theta}(-\log P(x)) = E_h[\frac{\partial \text{Energy}(x, h)}{\partial \theta}|x] - E_{x,h}[\frac{\partial \text{Energy}(x, h)}{\partial \theta}].$$

## 2.3 The Restricted Boltzmann Machine and Its Training Algorithms

The Restricted Boltzmann Machine (RBM) is one without lateral interactions, i.e., $U = 0$ and $V = 0$. In turns out that positive phase part of the gradient can be computed exactly and tractably in the easier special case of the RBM, because $P(h|x)$ factorizes into $\prod_i P(h_i|x)$. Similarly $P(x|h)$ factorizes into $\prod_j P(x_j|h)$,

which makes it possible to apply blocked Gibbs sampling (sampling $h$ given $x$, then $x$ given $h$, again $h$ given $x$, etc.).

RBMs are typically trained by stochastic gradient descent, using a noisy (and generally biased) estimator of the above log-likelihood gradient. The first gradient estimator that was proposed for RBMs is the Contrastive Divergence estimator (?, ?), and it has a particularly simple form: the negative phase gradient is obtained by starting a very short chain (usually just one step) at the observed $x$ and replacing the above expectations by the corresponding samples. In practice, it has worked very well for unsupervised pre-training meant to initialize each layer of a deep supervised (?, ?, ?) or unsupervised (?) neural network.

Another common way to train RBMs is based on the Stochastic Maximum Likelihood (SML) estimator (?) of the gradient, also called Persistent Contrastive Divergence (PCD) (?) when it was introduced for RBMs. The idea is simply to keep sampling negative phase $x$'s (e.g. by blocked Gibbs sampling) even though the parameters are updated once in a while, i.e., without restarting a new chain each time an update is done. It turned out that SML gives RBM with much better likelihood, whereas CD updates sometimes give rise to worsening likelihood and suffers from other issues (?). Theory suggests (?) this is a good estimator if the parameter changes are small, but practice revealed (?) that it worked even for large updates, in fact giving rise to faster mixing (?, ?). This is happening because learning actually interacts with sampling in a useful way, pushing the MCMC out of the states it just visited. This principle may also explain some of the fast mixing observed in a related approach called Herding (?, ?).

RBMs can be stacked to form a Deep Belief Network, a hybrid of directed and undirected graphical model components, which has an RBM to characterize the interactions between its top two layers, and then generates the input through a directed belief network. See ? (?) for a deeper treatment of Boltzmann Machines, RBMs, and Deep Belief Networks.

## 2.4 The Zoo: Auto-Encoders, Sparse Coding, Predictive Sparse Decomposition, Denoising Auto-Encoders, Score Matching, and More

**Auto-encoders** are neural networks which are trained to reconstruct their input (?, ?, ?). A one-hidden layer auto-encoder is very similar to an RBM and its reconstruction error gradient can be seen as an approximation of the RBM log-likelihood gradient (?). Both RBMs and auto-encoders can be used as the one-

layer unsupervised learning algorithms that gives rise to a new representation of the input or of the previous layer. In the same year that RBMs were successfully proposed for unsupervised pre-training of deep neural networks, auto-encoders were also shown to help initialize deep neural networks much better than random initialization (?). However, ordinary auto-encoders generally performed worse than RBMs, and were unsatisfying because they could potentially learn a useless identity transformation when the representation size was larger than the input (the so-called "overcomplete" case).

**Sparse coding** was introduced in computational neuroscience (?) and produced filters very similar to those observed in cortex visual area V1 (before similar filters were achieved with RBMs and sparse predictive decomposition, and denoising auto-encoders, below). They correspond to a linear directed graphical model with a continuous-valued latent variable associated with a sparsity prior (Student or Laplace, the latter corresponding to an L1 penalty on the value of the latent variable). This is like an auto-encoder, but without a parametric encoder, only a parametric decoder. The "encoding" corresponds to inference (finding the most likely hidden code associated with observed visible input) and involves solving a lengthy but convex optimization problem and much work has been devoted to speeding it up. A very interesting way to do so is with **Predictive Sparse Decomposition** (?), in which one learns a parametric encoder that approximates the result of the sparse coding inference (and in fact changes the solution so that both approximate encoding and decoding work well). Such models based on approximate inference were the first successful examples of stacking a sparse encoding (?, ?) into a deep architecture (fine-tuned for supervised classification afterwards, as per the above greedy-layerwise recipe).

**Score Matching** is an alternative statistical estimation principle (?) when the maximum likelihood framework is not tractable. It can be applied to models of continuous-valued data when the probability function can be computed tractably up to its normalization constant (which is the case for RBMs), i.e., it has a tractable energy function The *score* of the model is the partial derivative of the energy with respect to the input, and indicates in which direction the likelihood would increase the most, from a particular input $x$. Score matching is based on minimizing the squared difference between the score of the model and a target score. The latter is in general unknown but the score match can nonetheless be rewritten in terms of the expectation (under the data generating process) of first and (diagonal) second derivatives of the energy with respect to the input, which correspond to a tractable computation.

**Denoising Auto-Encoders** were first introduced (?) to bypass the frustrating

limitations of auto-encoders mentionned above: they are only meant to learn a "bottleneck", a reduced-dimension representation. The idea of Denoising Auto-Encoders (DAE) is simple: feed the encoder/decoder system with a *stochastically corrupted input*, but ask it to *reconstruct the clean input* (as one would typically do to train any denoising system). This small change turned out to systematically yield better results than those obtained with ordinary auto-encoders, and similar or better than those obtained with RBMs on a benchmark of several image classification tasks (?). Interestingly, the denoising error can be linked in several ways to the likelihood of a generative model of the distribution of the uncorrupted examples $x$ (?, ?), and in particular through the Score Matching proxy for log-likelihood (?). The link also sheds light on why a denoising auto-encoder captures the input distribution. The difference vector between the reconstruction and the input is the model's guess as to the direction of greatest increase in the likelihood, whereas the difference vector between the noisy corrupted input and the clean original is nature's hint of a direction of greatest increase in likelihood (since a noisy version of a training example is very likely to have a much lower probability under the data generating distribution than the original). The difference of these two differences is just the denoising reconstruction error residue.

**Noise-Contrastive Estimation** is another estimation principle for probability models for which the energy function can be computed but not the partition function (?). It is based on training not only from samples of the target distribution but also from samples of an auxiliary "background" distribution (e.g. a flat Gaussian). The partition function is considered like a free parameter (along with the other parameters) in a kind of logistic regression trained to predict the probability that a sample belongs to the target distribution or to the background distribution.

**Semi-Supervised Embedding** is a way to use unlabeled data to learn a representation (e.g., in the hidden layers of a deep neural network), based on a hint about *pairs of examples* (?, ?). If some pairs are expected to have a similar semantic, then their representation should be encouraged to be similar, whereas otherwise their representation should be at least some distance away. This idea was used in unsupervised and semi-supervised contexts (?, ?, ?), and originates in the much older idea of *siamese networks* (?).

# 3 Convolutional Architectures for Images and Sequences

When input observations are structured according to some invariance, it is often very useful to exploit that invariance in a learning machine. We would like features that characterize the presence of objects in sequences and images to have some form of *translation equivariance*: if the object is translated (temporally or spatially), we would the associated feature detectors to also be translated.

## 3.1 weight sharing

The classical way of obtaining some form of translation equivariance is the use of weight sharing. The same feature detector is applied at different positions or time steps, thus yielding a sequence or a "map" containing the detector output for different positions or time steps. The idea of local receptive fields goes back to the Perceptron and to Hubel and Wiesel's discoveries (?) in the cat's visual cortex. It has been most influential in machine learning through so-called convolutional networks ? (?, ?).

## 3.2 feature pooling

- very popular recently (mcRBM, Coates et al. etc.)
- adds robustness to the representation and a degree of invariance.
- seems to significantly improves classification performance.
- Does it contribute to learning a more disentangled representation?

# 4 Learning Invariant Feature sets

## 4.1 What are factors of variation

For many AI-tasks, the data is derived from a complex interaction of factors that act as sources of variabilty. When combined together, these factors given rise to the rich structure characteristic of AI-related domains. For instance, In the case of natural images, the factors can include the identity of objects in a scene, the orientation and position of each object as well as the ambient illumination

conditions. In the case of speech recognition, the semantic content of the speech, the speaker identity and acoustical effects due to the environment are all sources of variability that give rise to speech data. In each case, factors that are relevant to a particular task combine with irrelevant factors to render the task much more challenging.

As a concrete example consider the task of face recognition. Two images of the same individual with different poses (eg. one image is in a full frontal orientation, while the other image is of the individual in profile) may result in images that are well separated in pixel space. On the other hand images of two distinct individuals with identical poses may well be positioned very close together in pixel space. In this example, there are two factors of variation at play: (1) the identity of the individual in the image, and (2) the person's pose with respect to the image plane. One of these factors (the pose) is irrelevant to the face recognition task and yet of the two factors it could well dominate the representation of the image in pixel space. As a result, pixel space-based face recognition systems are destined to suffer from poor performance do to sensitivity to pose.

The key to understanding the significance of the impact that the combination of factors has on the difficulty of the task is to understand that these factors typically do not combine as simple superpositions that can be easily separated by, for example, choosing the correct lower-dimensional projection of the data. Rather, as our face recognition example illustrates, these factors often appear tightly entangled in the raw data. The challenge for deep learning methods is to construct representations of the data that somehow attempt to cope with the reality of entangled factors that account for the wide variability and complexity of data in AI domains.

## 4.2 how do we deal with factors of variation: invariant features

In an effort to alleviate the problems that arrive when dealing with this sort of richly structured data, there has recently been a very broad based movement in machine learning toward building feature sets that are *invariant* to common perturbation of the datasets. The recent trend in computer vision toward representations based on large scale histograming of low-level features is one particularly effective example [CITATIONS]. [Other examples?]

To a certain degree, simply training a deep model – whether it be by stacking a seriers of RBMs as in the DBN (in section **??**) or by the joint training of the layers of a Deep Boltzmann Machine (discussed in section **??** – should engender an increasing amount of invariance to increasingly higher-level representations.

However with our current set of models, it appears as though depth alone is insufficient to foster a sufficient degree of invariance at all levels of the representation and that an explicit modification of the inductive bias of the models is warranted.

Within the context of deep learning, the problem of learning invariant feature sets has long been considered an important goal. As discussed in some detail in section **??**, one of the key innovations of the convolutional network architecture is the inclusion of max-pooling layers. These layers pool together locally shifted versions of the filters represented in the layer below. The result is a set features that are invariant to local translations of objects and object parts within the image.

More generally, invariant features are designed to be insensitive to variations in the data that are uninformative to the target task while remaining selective to relevant aspects of the data. The result is a more stable representation that is well suited to be used as an input to a classifier. With irrelevant sources of variance removed, the resulting feature space has the property that distances between data points represented in this space are a more meaningful indicator of their true similarity. In classification tasks, this property naturally simplifies the separation of training data associated with different class labels.

Thus far, we have considered only invariant features, such as those found in the convolutional network, whose invariant properties were hand-engineered by specifying the filters to be pooled. This approach, while clearly effective in constructing features invariant to factors of variation such as translation, are fundamentally limited to expressing types of invariance that can be imposed upon them by human intervention. Ideally we would like our learning algorithms to automatically discover appropriate sets of invariant features. Features sets that *learn* to be invariant to certain factors of variation have the potential advantage of discovering patterns of invariance that are either difficult to hand-engineer (e.g. in-plane object rotations) or simply *a priori* not known to be useful. In the remainder of this section we review some of the recent progress in techniques to learning invariant features and invariant feature hierarchies.

[talk about invariance engendered by localized filters. i.e. they are trivially invariant to variations outside their receptive field.]

## 4.3 Invariance via Sparsity

Learning sparse feature sets has long been popular both in the context of learning feature hierarchies as a deep learning strategy and as a means of learning effective shallow representations of the data. In the context of an object recognition task, ? (?) established that using a sparse representations learned from image data as

input to an SVM classifyer lead to better classification performance than using either to the raw pixel data or a non-sparse PCA representation of the data.

Recently, ? (?) showed that sparsity can also lead to more invariant feature representations. In the context of auto-encoder networks trained on natural images [movies?], they showed that adding a sparsity penalty to the hidden unit activations that the resulting features are more invariant to specific transformations such as translations, rotations normal to the image plane as well as in-plane rotations of the objects.

At this point it is not clear by what mechanism sparsity promotes the learning of invariant features. It is certainly true that sparsity tends to cause the learned feature detectors to be more localized (in training on natural images, the learned features form Gabor-like edge detectors[CITE OLshausen]; in training on natural sounds, the learned features are wavelet-like in that they tend to be localized in both time and frequency [CITE Lewicki]. It is also true that localized filters are naturally invariant to variations outside their local region of interest or *receptive field*. It is not clear that feature locality is sufficient to entirely account for the invariance observed by Goodfellow et al. Nor is it clear that the superior performance classification performance of sparse representations observed by ? (?) and others [CITE] may be attributed to this property of generating more invariant features. What is clear is that these issues merit further study.

## 4.4 Teasing Apart Explanatory Factors Via Slow Features Analysis

We perceive the world around us through a temporally structured stream of perceptions (e.g. a video). Even as one moves their eyes and head, the identity of the surrounding objects generally does not change. More generally, a plausible hypothesis is that many of the most interesting high-level explanatory factors for individual perceptions have some form *temporal stability*. The principle of identifying slowly moving/changing factors in temporal/spatial data has been investigated by many (?, ?, ?, ?, ?) as a principle for finding useful representations of images, and as an explanation for why V1 simple and complex cells behave the way they do. This kind of analysis is often called *slow feature analysis*. A good overview can be found in (?). Note that it is easy to obtain features that change slowly because they are obtained through a recurrence (e.g. a moving average of current and past observation). Instead, in these works, one learns features of an instantaneous perception (e.g. a single image) such that consecutive values of

each feature change slowly. For this to be of any use, it is also required that these features capture as much as possible of the input variations (e.g. constant features would be very stable but quite useless). A very interesting recent theoretical contribution (?) shows that *if there exist categories and these categories are temporally stable, than slow feature analysis can discover them even in the complete absence of labeled examples.*

Temporal coherence is therefore a prior that could be used by learning systems to discover categories. Going further in that direction, we hypothesize that more structure about the underlying explanatory factors could be extracted from temporal coherence, still without using any labeled examples. First, different explanatory factors will tend to operate at *different time scales*. With such a prior, we can not only separate the stable features from the instantaneous variations, but we could disentangle different concepts that belong to different time scales. Second, instead of the usual squared error penalty (over the time change of each feature), one could use priors that favor the kind of time course we actually find around us. Typically, a factor is either present or not (there is a notion of sparsity there), and when it is present, it would tend to change slowly. This corresponds to a form of sparsity not only of the feature, but also of its change (either no change, or small change). Third, an explanatory factor is rarely represented by a single scalar. For example, camera geometry in 3 dimensions is characterized by 6 degrees of freedom. Typically, these factors would either not change, or change together. This could be characterized by a group sparsity prior (?).

## 4.5 Learning to Pool Features

While there is evidence that sparsity contributes to learning invariant feature sets, the most reliable and effective way to construct invariant representations remains the pooling of features.

In section **??**, we encountered feature pooling (max-pooling) as a way to ensure that the representation in the max-pooling layers of the convolutional network were invariant to small translations of the image.

The principle that invariance to various factors of the data can be induced through the use of pooling together of a set of simple filter responses is a powerful one. In

The principle that invariant features can emerge from the organization of features into pools was established by ? (?). -feature subspaces was

**ASSOM: ? (?)**

- First instance of invariant feature learning via feature pooling?
- Synthesis of *self-organizing map* (SOM) architechture and the *learning subspace method*.

subspace ICA: Hyvarinen & Hoyer 2000 topographic ICA: Hyvarinen, Hoyer & Inki 2001

[IPSD] [mcrbm] [tiled-convolutional ICA]

# 5 Beyond Invariant Features: Disentangling Factors of Variation

Complex data arise from the rich interaction of many sources. An image is composed of the interaction between one or more light sources, the orbject shapes and the material properties of the various surfaces. An audio scene is composed of the ...

These interactions

The Deep learning approach to dealing with these issues is to attempt to leverage the data itself, ideally in vast quantities, to learn representations that *disentangle* the various explanatory sources. Doing so naturally renders a representation more invariance to small perturbations of the data and enables more robust classification performance.

Consider learning a represetation which will ultimately be used as input to a classifier.

It is important to distinguish between the related but distinct goals of learning invariant features and learning disentangled features. As discussed above, the goal of building invariant features sets is to learn a set of features that are invariant to common but irrelevant sources of variance in the data.

In building a disentangling feature set, the goal is to separate a number of relevant factors of variation in the dataset, while retaining some or all of them. While invariant features can be thought of as keeping the relevant signal

Obviously, what we really would like is for a particular feature set to be invariant to the irrelevant features and disentangle the relevant features. Unfortunately, it is often difficult to determine *a priori* which set of features will ultimiately be relevant to the task at hand. Further, as is often the case in the context of deep learning methods (cite Collobert and Westin and Kohler vision paper NIPS 2009),

the feature set being trained may be destined to be used in multiple tasks that may have distinct subsets of relevant features. Considerations such as these lead us to the conclusion that the most robust approach to feature learning is to disentangle as many factors as possible, discarding as little information about the data as is practical.

This perspective represents a small paradigm shift in the currently predominant deep learning methodology. However we believe that the potential benefits are immeasurable.

## 5.1 Neuroscientific Basis for Disentangling

Dynamic Routing Olshausen et al. (1993) Understanding V1 Olshausen and Field (2005)

## 5.2 Early work in Disentangling Factors

"Separating Style and Content with Bilinear Models" Tenenbaum and Freeman (2000). "Bilinear Sparse Coding for Invariant Vision" Grimes and Rao (2005). "Multilinear ICA" Vasilescu and Terzopoulos (2005).

While there are presently few examples of work dedicated to the task of disentangling One promising direction for...

It remains to integrate this work into the wider context of deep learning methods. One of the main motivations for seeking a representation that disentangles the factors of variation – as oppose to features being selectively invariant with respect to these factors – is to be able to learn the statitical relationships between the various factors. This learning of the various

# 6 On the importance of top-down connections

- Ruslan's DBM