Learning sparse and deep representations: new algorithms and perspectives

James Newling
I&C, EPFL

Abstract—Three papers related to representation learning are discussed. The first paper presents a broad overview of the field of unsupervised representation learning, the second paper presents a stochastic model for images, and the final paper presents a new approach to encoding images. The relationship between these papers is discussed, and a modification to the method of the third paper is proposed.

Index Terms—thesis proposal, candidacy exam write-up, EDIC, EPFL, Deconvolutional Network, POP, Unsupervised Learning

I. INTRODUCTION

In recent years multilayer or deep learning paradigms have outperformed all other approaches on several classical machine learning tasks. The rise to prominence of multilayer learning can be said to have started in 2006 with Deep Belief Networks, since when there has been a barrage of improvements on increasingly challenging tasks using ever deeper and wider architectures. Initially, the stunning improvements were attributed in part to the use of unsupervised learning, although recent groundbreaking image classification results using purely supervised Convolutional Neural Networks have brought into question such accreditation.

The importance of depth and the relevance of unsupervised learning to machine learning are the main topics of the first paper. The use of deep architectures is also closely related to the second paper reviewed, where images are modelled as hierarchies of parts. The third paper presents a new unsupervised algorithm for learning sparse, deep representations of images. In Section II each of these papers is discussed, and then in Section III possible modifications to the third paper are presented.

II. SELECTED PAPERS

A. Representation Learning: A Review and New Perspectives

In this review paper, the authors attempt to make sense of the various approaches which have been proposed for unsupervised feature learning, with a focus on deep network architectures. They broadly classify the different approaches and discuss how they are related to each other, giving their opinion on what the most pressing questions in the field may be. The paper contains few details and many claims are left unsubstantiated, but I suspect this is simply a reflection of the murky state the field is currently in.

Deep learning successes spanning speech recognition, signal processing, object recognition and natural language processing are presented, although many of the cited successes are purely supervised and thus only tenuously related to the main topics of the paper. The authors mention potential benefits of representation learning to transfer learning tasks, but it not clear whether the claim pertains only to the case where unlabelled data is available or more generally.

The authors present a list of general purpose priors which could be useful for directing unsupervised feature learning research. The list contains priors related to the distribution of $X$ (observed data), others relating $X$ and $Y$ (target), and yet others which seem more to be desirable properties that $X$ and $Y$ might have for successful learning than anything else. Some of the more clear points are,

1) smoothness: For regression problems $Y = f(X) + \epsilon$, the function $f$ is smooth.
2) hierarchical: Concepts (features) useful for explaining $X$ are best described by more ‘abstract’ concepts, which are themselves described by more abstract concepts etc.
3) semi-supervised: Abstract features describing $X$ also describe $Y$. 
4) manifolds: Probability mass for X concentrates in low dimensional manifolds.
5) natural clustering: manifolds describing different Y are distinct.
6) sparsity and invariance: Features describing X are invariant to most changes in X.

In addition to these general ideas, the authors propose two advantages of multilayer architectures,

1) feature re-use: low level features can be combined in an exponential (in number of layers) number of ways to form high level features. However there is simultaneously an exponential growth in the VC dimension of the model and so overfitting still poses a problem.
2) abstraction: as units in higher layers consist of nonlinear combinations of lower level units, the architecture seems appropriate for abstraction.

The idea of disentangling is introduced. The authors propose that the value in good feature learning lies in the disentanglement of features to form representations which are more amenable to supervised training. This is discussed in the context of shadows in machine vision, where the ability to disentangle and ultimately remove shadows is desirable. However, as acknowledged by the authors, how to translate the desire to disentangle features into a well defined minimisation problem is unclear, and remains a big challenge in unsupervised feature learning.

The approaches to deep representation learning can be broadly divided into two groups. There are the approaches based on probabilistic graphical models, and those based on neural networks. Ultimately the difference lies in whether hidden units are considered to be latent random variables or computational units. The authors highlight the similarities between the two approaches. As an example sparse coding, which superficially belongs to the computational group, can equally well be viewed as obtaining the MAP estimate of a graphical model.

The idea of explaining away is presented in the context of directed graphical models. The idea that latent variables which are independent (earthquake/burged) can become highly dependent when conditioned on lower level variables (alarm) and thus compete in explaining lower level variables. This idea is related to a problem faced in the third paper presented [1] and will be discussed later. The authors claim that the phenomenon of explaining away is not observed in undirected graphical models such as RBMs, although they do not provide an explanation for this. Explaining away is claimed to be an extremely useful characteristic as it provides parsimony.

A thorough review of undirected graphical models and their training is presented in the paper. This is followed by a clear exposition of various non-stochastic models, in particular autoencoders (sparse, denoising, contractive) and predictive sparse decompositions. Finally, an overview of manifold learning and its connection to autoencoders is given. I will not present details of these topics here.

The authors talk about recent groundbreaking classification results (2011, 2012) where very large labelled training sets are used without any unsupervised pre-training. The successes suggests that unsupervised pre-training acts as a type of prior, which in the presence of large amounts of data becomes overwhelmed. The authors pose the question of why no previous results (2006 to 2011) trained on large datasets revealed this. Certain elements of the design and training of deep neural networks are highlighted as areas potentially important for representation learning. An example is the pooling operation, generalisations of which may be fertile regions for future research.

In summary, the paper takes us on a tour of the current state of the art in unsupervised feature learning, attempts to define what good feature learning is about and proposes elements of current approaches which seem most promising for future research.

B. POP:Patchwork of Parts Models for Object Recognition

1) Background: In this paper, a generative model for image edges is presented and used for detection and classification. The paper is self-contained and the presentation is very clear, with each additional layer of complexity being added only when appropriate. I will attempt to summarise the model for images containing a single object, with the extension to multi-object images is straightforward.

2) The model: The POP model generates binary edge feature maps for E edge orientations on a grid L. The random variable being modelled is $X = \{X_e(x)|x \in L, e = 1 \ldots E\}$, where $X_e(x) = 1$ designates the presence of an edge of orientation $e$ at location $x$. The authors first introduce a rigid model, where a single probability array $(p_e(y))_{y \in \mathbb{Z}^2}$ is defined. Conditional on an object being detected at position $r$, the marginal probability is given by

$$P(X_e(x) = 1|r) = p_e(x - r).$$

For non-rigid objects the model fails as small deformations misalign object parts from the rigid template. To overcome this weakness, an object location is defined by several points instead of just a mean displacement $r$. There is still a single probability array, but now $n$ reference points are chosen $(y_i)_{i=1 \ldots n}$ within the probability array (template) to which an object within an image instantiation needs to be aligned. The position of an object within an instantiation is defined by

$$\theta = (r, v_1 \ldots v_n)$$

where the reference points in the instantiation are located at $z_i = y + r_i + v_i$. Only points within a small distance of reference points are expected to be well aligned with the template, and so the authors choose to exclude points outside a window $W$ of reference points in modelling the marginal probability, now given by

$$P(X_e(x) = 1|\theta) = \begin{cases} \frac{1}{|\mathcal{I}(x)|} \sum_{i \in \mathcal{I}(x)} (p_e(x - z_i + y_i)), & \text{if } \mathcal{I}(x) \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases}$$

where $\mathcal{I}(x) = \{i : x \in z_i + W\}$, and points in the image lying within multiple windows, $(|\mathcal{I}(x)| \geq 2)$ have edge probabilities being an average across occupied windows.
A support for $\theta$ is defined,

$$S(\theta) = \{ x \in L : \max p_c(x|\theta) > \rho \}.$$

To account for the appearance of edges in the background, the model stipulates that

$$P(X_r(x) = 1|x \notin S(\theta)) = p_{e,bgd}.$$

The marginal probability for $X$ is now completely specified,

$$P(X|\theta) = \prod_{x \in S(\theta)} \prod_{c} [p(x|\theta)]^{X_c(x)} [1 - p(x|\theta)]^{1 - X_c(x)} \times \prod_{x \notin S(\theta)} \prod_{c} [p_{e,bgd}]^{X_c(x)} [1 - p_{e,bgd}]^{1 - X_c(x)}.$$

A prior on the location parameter $\theta$, denoted $f(\theta)$ is specified as being independent in $r$ and an $n$-dimensional Gaussian in $v$: $v_1 \ldots v_n \sim N(0, \hat{\Sigma}).$ The covariance matrix $\hat{\Sigma}$ is obtained in the training stage. Finally, for a class of objects $c$ there are $M_c$ separate models, with a prior on model $m$ of $P_c(m)$. We thus have,

$$P(X|C = c) = \sum_{m=1}^{M_c} P_c(m) \int P_{c,m}(X|\theta)f_{c,m}(\theta) d\theta.$$

The idea is for gross variability in realisations of class $c$ to be captured by the different models, with smaller variations being captured by the location parameter.

3) Object classification, detection, and model fitting: Given an image containing an object of unknown class, with prior class probabilities $P(C = c)$ for $c = 1 \ldots N_C$, the Bayes classifier is

$$\hat{c} = \arg \max_c P(X|C = c)P(C = c).$$

The authors prefer rather to compare probabilities of MAP estimates of each model, as computing (1) is infeasible when $n$ is large. They also assume a flat class prior, thus using

$$\hat{c} = \arg \max_c \max_{1 \leq m \leq M_c} \max_{\theta} P_{c,m}(X|\theta = \hat{\theta})f_{c,m}(\hat{\theta}).$$

The inner maximisation over $\hat{\theta}$ is a high dimensional and non-convex problem. The authors suggest two greedy approaches to approximating the solution and discuss the merits of each, these will not be discuss here.

The problem of detecting instances of class $c$ is solved by localizing all $r$ such that $J(r) > \tau_c$, where

$$J(r) = \max_{1 \leq m \leq M_c} \max_v P_{c,m}(X|r, v)f_{c,m}(v)$$

and $\tau_c$ is some threshold value.

Finally, I outline the slightly trickier problem of fitting the model. First consider the case where each class contains only one model. The parameters to learn are then $\Sigma$, $(p_e(y))_{y \in \mathbb{Z}^2}$ and $(y_i)_{i=1 \ldots n}$. These parameters are related to observed edge maps via the (latent) displacement variables. The approach taken by the authors is to obtain each $y_i$ along with values of $(p_e(y))_{y \in \mathbb{Z}^2}$ in it’s vicinity separately, and then weave the $n$ patches of probabilities together at the end. The number of reference points $n$ is not specified up front; reference points are initially proposed on a grid, in general there are more than $n$ such initial reference points. Only the reference points with final edge probabilities significantly different from the background level are retained in the final model. Learning is done using the EM algorithm, alternating between updating edge probabilities and displacement probabilities. The parameter $\Sigma$ is estimated only once $(p_e(y))_{y \in \mathbb{Z}^2}$ and $(y_i)_{i=1 \ldots n}$ have been obtained, combining the covariance of optimal displacement parameters in the training set with a prior estimate. The extension to multiple models per class is done in a dynamic and greedy way such that the decision to create a new class model is taken when all models have been sufficiently trained and the new training data is sufficiently distinct from existing models to warrant a new model.

4) Results and Comments: For object classification and detection, POP is tested on the MNIST digit dataset and a CMU-MIT face dataset. For multiobject detection, POP is tested on a zipcode dataset, this will not be discussed here. Comparison is made between POP and an SVM, with POP performing significantly better than the SVM when the training set is small (10 - 100 instances per class), but no better when the training set is large. The success with small training sets may be due to the well engineered priors that POP has. The authors experiment with various parameters and subroutines, the most significant of which seems to be how training data is split between class models. We previously mentioned the greedy algorithm proposed for model splitting, but the authors find that a more careful splitting via an EM type algorithm on all the training data simultaneously results in models with lower classification error rates. The best result they obtain using all of the data is a very impressive 0.68% classification error rate on MNIST, although the detail of what they train on to obtain this value is not clear.

The most evident is perhaps superficial connection that POP has to deep neural networks is its compositional nature, corresponding to the ‘abstraction’ highlighted in [2] as a likely advantage of deep architectures. The second advantage postulated by [2] is feature re-use which the POP model does not exhibit. The POP model allows for deformation invariances to be explicitly encoded in the covariance matrix of the location parameters, whereas the underlying mechanism by which deformation is handled in deep neural networks is unknown.

C. Deconvolutional Networks

Initially proposed as a method for unsupervised learning of low- and mid-level features for image recognition [1], [3], Deconvolutional Networks (DNs) have proved useful as a supplementary tool in the visualisation of neural networks [4].

Deconvolutional Networks can be viewed as a form of convolutional sparse coding, and as such act as decoders and not encoders. Thus it is easy to infer low level features from high level features but difficult to infer high level features from low level features, where by low level features we mean features near to the input layer. This is in contrast to unsupervised feature learning algorithms such as Deep Belief Networks [5] and Sparse Autoencoders [6] which consist of both an encoder and decoder part, and others such as
Convolutional Networks [7] which consist strictly of encoding. The stated hope of forgoing an explicit encoder with DNs is that more parsimonious features may be learned if ease of encoding is sacrificed. This is related to the idea of explaining away mentioned earlier, where a parsimonious explanation (encoding) is found by allowing units to compete with each another. Before further discussion, I will briefly introduce the model and algorithms.

Using the notation of of the paper, an image $y_i$, composed of $K_0$ colour channels $y_i^1 \ldots y_i^{K_0}$, has each channel $c$ represented as a sum of $K_1$ latent feature maps $z_k^c$ convolved with filters $f_{k.c}$:

$$y_i^c = \sum_{k=1}^{K_1} z_k^c \odot f_{k.c}. \quad (3)$$

Ultimately we would like filters $f_{k.c}$ which will provide optimal results on some unknown task and so, as with all unsupervised learning problems, a proxy objective function needs to be defined. The formulation (3) is underdetermined (given filters $f_{k.c}$, there is no unique solution $z_k^c$). The authors thus introduce a regularisation term on $z_k^c$ resulting in the loss function,

$$C_1(y^c) = \frac{\lambda}{2} \sum_{c=1}^{K_0} \sum_{k=1}^{K_1} \|z_k^c \odot f_{k.c} - y_i^c\|^2 + \sum_{k=1}^{K_1} |z_k^c|^p. \quad (4)$$

When $p \geq 1$, the loss function (4) is convex in $z$ and $f$ separately but not simultaneously, and thus one can iteratively alternate between minimising and updating $z$ and $f$ separately to obtain an approximate solution. However, even though it is a convex problem when $p \geq 1$, obtaining the $z$ which minimises (4) for fixed filters $f$ is a difficult and poorly conditioned problem according to the authors. Indeed the authors claim that direct minimisation does not work as well as a continuation method which minimises an auxiliary loss function given by,

$$C_1(y^c) = \frac{\beta}{2} \sum_{c=1}^{K_0} \sum_{k=1}^{K_1} ||z_k^c \odot f_{k.c} - y_i^c||^2 + \frac{\beta}{2} \sum_{k=1}^{K_1} ||z_k^c - x_k^c||^2 + \sum_{k=1}^{K_1} |x_k^c|^p. \quad (5)$$

One sees that when $\beta$ is large, the minimizer of Eqn. (5) provides a close approximation to the minimizer of Eqn. (4). Thus, by alternatively minimising Eqn. (5) in $x$ and in $z$ for successively greater values of $\beta$ and using solutions from the preceding $\beta$ iteration as initial guesses, one converges to the true $z$ minimizing (4). The exact algorithm for obtaining feature maps in reproduced as Alg. (1).

For learning the filters, several epochs of alternating between minimising $f$ in (5) and minimising $z$ for all images via Alg. (1) are run. The full algorithm for optimising the filters thus contains an EM-like algorithm within an EM-like algorithm, with several parameters needing to be set. In our experiments, the fine tuning of parameters to obtain sensible results is difficult and time consuming, and obtaining the optimal solution is slow.

**Algorithm 1 Solve Z**

Input includes:
1. small initial $\beta$ value, $\beta_0$
2. $\beta$ growth factor, $\beta_{inc}$
3. Initial $z$ estimate

while $\beta < \beta_{max}$ do

Given $z$ minimize and update $x$ in (5)

Given $x$ minimize and update $z$ in (5).

$\beta \leftarrow \beta \cdot \beta_{inc}$

return $z$

The authors use $p = 1$ throughout, which they claim encourages sparsity. In our experiments, the feature maps obtained with $p = 1$ are not sparse. While the authors do not claim that sparsity is their primary objective, we believe it to be desirable. Indeed, the authors acknowledge that taking $0 < p < 1$ in (5) does not require any modifications to the continuation method just described. They do not mention that $p = 0$ is as simple to implement, and they do not perform experiments with any value other than $p = 1$. In Section III-A this will be further discussed in the context of my PhD proposal.

To obtain higher level features the process is repeated on the resulting feature maps. I have two concerns about this approach which are unfortunately not addressed by the authors. Firstly, the resulting decoder is linear, and it seems implausible that images can be expressed as linear combinations of abstract concepts. Secondly, while it seems plausible that smooth images can be approximated (in the $\ell_2$-norm sense) as sparse linear combinations of patches, when images are themselves sparse and spiky it seems significantly more challenging, unless the spike positions have strong dependencies. Both of these concerns could be overcome with the use of maxpooling, as introduced in the 2011 paper [3].

**1) Results and Further Discussion:** Multilayer features are learned and extracted on the Caltech-101 dataset, concatenated using Spatial Pyramid Matching, and a final SVM classifier is learned. Comparison is made to the case where features are SIFT-based, where a marginal improvement is observed using features from a DN. However, there does not appear to be a significant improvement in classification with the addition of a second Deconvolution layer, begging the question of whether anything useful has been learned in the second layer. Further concerns with the approach as presented in this paper are the following:

- sparsity: The authors champion sparse representations, but in reality the feature maps are not sparse, at least in the experiments we have performed and outlined in Section III-A. Figure 6 in the paper suggests sparsity, but this is in the $\ell_1$ sense and thus not true sparsity. While an $\ell_1$ penalty often equates to sparse features, such as is the case in sparse coding, it is not always so.
- computational burden of encoding: The algorithm used for encoding as described is sensitive to hyperparameter choice and computationally heavy. Moreover the solution
returned is not a global minimizer of (5).
- decoding: Section 3.3 of [1], which describes image reconstruction, is not clear, but it may be that my previous comment about the decoder being linear are incorrect.

III. RESEARCH PROPOSAL

A. Stamping algorithm

As mentioned there are two properties of DNs which we believe can be improved on: simplicity of implementation and computational speed. A third property is sparsity: although true sparsity could be obtained within the DN framework by setting \( p = 0 \) in (5), the authors do not do this. We propose a new low-tech algorithm, called Stamping, which makes improvements in these domains.

- Stamping is simple to implement and has zero parameters to tune.
- Stamping is fast, with a running\(^1\) time \( O\left(kWF\left(\log_k(WHF) + whi\right)\right)\), where the image is of size \( W \times H \) and the \( F \) stamps are of size \( w \times h \). This is essentially the time to convolve the images with all of the filters \( k \) times (see Section III-A3 for details), where \( k \) is small (less than 10).
- Stamping produces truly sparse feature maps. Stamping is similar to DNs in that the two main steps are 1) given filters, obtain feature maps of images so as to (approximately) minimise a loss function and 2) given feature maps, update filters so as to (approximately) minimise the same loss function.

How (1) and (2) are performed differs between DNs and Stamping. With DNs, an attempt is made to find exact solutions to (1) and (2) while in Stamping the low-tech approximation described below is used.

1) Obtaining feature maps: A greedy approach to obtaining feature maps is taken. The idea is to iteratively stamp the residual image with the filter (the stamp henceforth) and at the position which results in the largest possible decrease in residual image with the filter (the stamp feature maps is taken. The idea is to iteratively approximations described below is used.

2) Updating stamps: The stamps are updated so as to

\[
\begin{align*}
s_{\text{new}} & = \arg \min_{s \neq s_{\text{old}}} \sum_{i=1}^{n_r} (\delta_i s - r_i)^2, \tag{6}
\end{align*}
\]

where \( \delta_i \) is the amount that stamp \( s_{\text{old}} \) was used to remove patch \( r_i \) of the residual image, and \( n_r \) is the number of times stamp \( s \) was used. This results in the simple formula for stamp \( s_{\text{new}} \):

\[
s_{\text{new}} \leftarrow \text{normalise} \left( \frac{n_r}{\sum_{i=1}^{n_r} \delta_i r_i} \right)
\]

Pseudocode for the two steps is presented in Algs. (2) and (4)

\(^{1}\)LazyStamp, see III-A3

\(^{2}\)with Stamping the convolution is performed with transposed stamps

Algorithm 2 StampImage

Input: Stamps, Image
//Initialise the feature maps \( Z[s] \) for \( 0 \leq s < n \text{Stamps} \) to zero:
for \( 0 \leq s < n \text{Stamps} \) do
for \( 0 \leq i < \text{ImageHeight} - \text{StampHeight} \) do
for \( 0 \leq j < \text{ImageWidth} - \text{StampWidth} \) do
\( Z[s,i,j] \leftarrow 0 \)
//Initialise the cumulative weighted patches array to be zero:
for \( 0 \leq s < n \text{Stamps} \) do
for \( 0 \leq i < \text{StampHeight} \) do
for \( 0 \leq j < \text{StampWidth} \) do
\( \text{cPatches}[s,i,j] \leftarrow 0 \)
Gains, BestZs \leftarrow \text{getGains}(\text{Residual}, \text{Stamps})
BestS, BestI, BestJ \leftarrow \text{argmax}(\text{Gains})
BestGain \leftarrow \text{Gains}[\text{BestS}, \text{BestI}, \text{BestJ}]
//While there is a gain above small threshold \( \epsilon \) to be had by stamping, stamp and update
while \( \text{BestGain} > \epsilon \) do
\( \text{BestZ} \leftarrow \text{BestZs}[\text{BestS}, \text{BestI}, \text{BestJ}] \)
\( \text{Delta} \leftarrow \text{BestZ} - Z[\text{BestS}, \text{BestI}, \text{BestJ}] \)
\( Z[\text{BestS}, \text{BestI}, \text{BestJ}] \leftarrow \text{BestZ} \)
\( \text{Patch} = \text{Residual}[\text{BestI}:\text{BestI}+\text{StampHeight}, \text{BestJ}:\text{BestJ}+\text{StampWidth}] \)
\( \text{cPatches}[\text{BestS}] \leftarrow \text{cPatches}[\text{BestS}] + \text{Delta}*\text{Patch} \)
\( \text{Residual} \leftarrow \text{Residual} - \text{Delta}\ast\text{Stamps}[\text{BestS}] \)
Gains, BestZs \leftarrow \text{getGains}(\text{Residual}, \text{Stamps})
BestS, BestI, BestJ \leftarrow \text{argmax}(\text{Gains})
BestGain \leftarrow \text{Gains}[\text{BestS}, \text{BestI}, \text{BestJ}]
Return \( Z \), \( \text{cPatches} \)

Algorithm 3 getGains

Input: Residual, Stamps
Returns two arrays:
(1) Gains, which is an array of the same size as \( Z \), containing the maximum values by which the loss function can be reduced with Stamping, for each (stamp, i, j) conditional on all other positions remaining unchanged.
(2) BestZs, the total amount stamped (may be negative, zero or positive) at each each position to obtain minimum loss (maximum gain). The details depend on the loss function.

Algorithm 4 UpdateStamps

Input: Stamps, Images
//Initialise the cumulative (for all images) weighted patches array to be zero:
for \( 0 \leq s < n \text{Stamps} \) do
for \( 0 \leq i < \text{StampHeight} \) do
for \( 0 \leq j < \text{StampWidth} \) do
\( \text{ccPatches}[s,i,j] \leftarrow 0 \)
for Image in Images do
\( Z, \text{cPatches} \leftarrow \text{StampImage}(\text{Stamps}, \text{Image}) \)
\( \text{ccPatches} \leftarrow \text{ccPatches} + \text{cPatches} \)
Stamps \leftarrow \text{normalised}(\text{ccPatches}) \)
3) LazyStamp and Results on MNIST: In Figure 1, results comparing DNs and Stamping are presented. On the left is plotted the $l_1$ norm of feature maps vs the reconstruction error. On the right is plotted $l_0$ norm of feature maps vs Pearson correlation between reconstruction and image. Minimising the $l_0$ norm of the feature maps is more desirable than minimising the $l_1$ norm of the feature map in terms of obtaining true sparsity, and we believe that the Pearson correlation between two images is a better indicator of similarity between the reconstruction and the original image than residual error as there is no semantic change to an image when scaled, so be believe the right figure is more useful for performing comparisons.

In the left figure DNs with $l_1$ penalty perform best, particularly for large $l_1$ norms. On the other hand, the figure on the right suggests the advantage of Stamping with $l_0$ penalty for obtaining truly sparse feature maps. We also have an implementation of Stamping which differs slightly from that presented in Algs. (2) and (4). LazyStamp, which is faster and appears to perform just as well. LazyStamp only recomputes Residual and Gain (see (2)) when all positions with potential gains are exhausted, and does not stamp in positions where the Residual has changed due to a previous nearby stamp within each round. In general, the number of such exhaustive rounds before completion of the algorithm (all gains less than $\epsilon$ after an update of Residual and Gain), which we call $k$, is small.

4) Future Work with Stamping: Future work with Stamping will be to perform classification and compare results with DNs and other methods such as [8]. We will then to consider approaches for transforming into a multilayer approach, we suspect that maxpooling or a related operation will be necessary between layers, as described in [3].

Stamping is essentially a matching pursuit algorithm, and thus well studied and compared to lasso like approaches in other contexts. I plan to explore these contexts and see if there are good ideas to adopted from them.

B. Other Area of Research

I am also working on unsupervised learning of Convolutional Networks (CNs). In [8] a single layer of filters learned by performing kmeans on image patches is used in a simple architecture to produce (in 2011) state of the art classification results on the CIFAR-10 dataset. The paper was followed by another paper by the same authors [9] in which additional layers were added, with improved classification results. The novelty of the second paper lies in how features between layers are sparsely connected, breaking the quadratic dependency of the number of parameters on network width. Currently I have successfully reproduced results for the single layer setup, and would like to reproduce the multilayer results before investigating new architectures and interlayer connection techniques. In brief, I am currently most interested in understanding and improving deep architectures with a focus on the unsupervised learning paradigm.

References


