Unsupervised Representation Learning with Laplacian Pyramid Auto-encoders

Zhao Qilu
Department of Computer Science
China University of Petroleum (East China)
kg19872006@163.com

Li Zongmin
Department of Computer Science
China University of Petroleum (East China)
lizongmin@upc.edu.cn

ABSTRACT

Scale-space representation has been popular in computer vision community due to its theoretical foundation. The motivation for generating a scale-space representation of a given data set originates from the basic observation that real-world objects are composed of different structures at different scales. Hence, it’s reasonable to consider learning features with image pyramids generated by smoothing and down-sampling operations. In this paper we propose Laplacian pyramid auto-encoders, a straightforward modification of the deep convolutional auto-encoder architecture, for unsupervised representation learning. The method uses multiple encoding-decoding sub-networks within a Laplacian pyramid framework to reconstruct the original image and the low pass filtered images. The last layer of each encoding sub-network also connects to an encoding layer of the sub-network in the next level, which aims to reverse the process of Laplacian pyramid generation. Experimental results showed that Laplacian pyramid benefited the classification and reconstruction performance of deep auto-encoder approaches, and batch normalization is critical to get deep auto-encoders approaches to begin learning.

CCS CONCEPTS

• Computing methodologies → Image representations, Neural networks.

KEYWORDS

Unsupervised representation learning, Auto-encoder, Laplacian pyramid, Convolutional neural networks, Deconvolutional net, Batch normalization

ACM Reference Format:
Zhao Qilu and Li Zongmin. 2018. Unsupervised Representation Learning with Laplacian Pyramid Auto-encoders. In Proceedings of ACM/IEEE International Conference on XXX XXX (ICXX’18), ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Real world objects are meaningful only at a certain scale. You might see an apple perfectly on a table. But if looking at the earth, then it simply does not exist. This multi-scale nature of objects is quite common in nature. Scale-space theory is a framework for early visual operations with complementary motivations from physics and biological vision, which has been developed by the computer vision community to handle the multi-scale nature of image data [20]. It is a formal theory for handling visual structures at different scales, by embedding the original image into a one-parameter family of derived images, in which fine-scale structures are successively suppressed. Scale-space representation has a wide application in computer vision. For example, the scale-invariant feature transform (SIFT) [21], a successful hand-crafted feature in computer vision to detect and describe local features in images, includes an important stage of key localization, which is defined as minima and maxima of the result of difference of Gaussians (DoG) function applied in scale space to a series of resampled and smoothed images.

In consideration of the successful applications of scale-space representation in hand-crafted feature engineering, it’s reasonable to apply it in unsupervised representation learning, especially nowadays when supervised deep learning methods have achieved great success in many tasks, owing to its ability to learn features from raw pixels. Recent work (DeCAF) [7] has shown that strong generic feature representations can be extracted from the activation of pre-trained networks. DeCAF defined a new visual feature by concatenating the flattened activations of each layer in the pre-trained networks, which is learned on a set of pre-defined object recognition tasks. This feature has shown strong generalization ability when it’s applied to new tasks, which suggests that there exists a generally useful feature representation for natural visual data. However, training deep models in a supervised way needs millions of semantically-labeled images which cost lots of manual work. Collecting large labeled datasets is very difficult, and there are diminishing returns of making the dataset larger and larger. Hence, unsupervised representation learning has drawn lots of attention for
quick access to arbitrary amounts of data, despite its performance is still limited so far.

The most common method used in unsupervised representation learning is an auto-encoder which learns representations based on an encoder-decoder paradigm. An auto-encoder (AE) [3] is an artificial neural network used for unsupervised learning of efficient coding. It consists of two parts, an encoder which outputs a hidden representation and a decoder which attempts to reconstruct the input from the hidden representation. In this paper we propose Laplacian pyramid auto-encoders (LPAE), a straightforward modification of the deep convolutional auto-encoder architecture, for unsupervised representation learning. The motivation for LPAE originates from two aspects:

1. There is a basic observation that real-world objects are composed of different structures at different scales. This implies that real-world objects may appear in different ways depending on the scale of observation. Hence, learning feature representations at multiple scales can make learning system robust to the unknown scale variations that may occur.

2. Auto-encoder uses a bottle-neck mechanism for forcing model abstraction, which can prevent a trivial identity mapping from being learned. The bottle-neck mechanism leads to an inherent tension: the greater the forced abstraction, the smaller the information content that can be expressed. Laplacian pyramid provides a framework for multi-path deep auto-encoders where each path can focus on a difference image which preserves part of the original image information. By mimicking the recovering process of the original image in Laplacian pyramid, a hierarchical encoding strategy is used to aggregate the information content expressed by each encoding path. This strategy would improve the learning efficiency of the whole model, enabling it to preserve information content as much as possible while avoiding a trivial identity mapping.

A typical architecture of LPAE is shown in Figure 1. LPAE is different with the traditional auto-encoder that tries to reconstruct its own inputs. LPAE uses multi-path auto-encoders to reconstruct the Gaussian pyramid from the Laplacian pyramid. Each path has connections with next level, which enables a hierarchical encoding strategy mentioned above.

2 RELATED WORK

Unsupervised representation learning, aiming to use data without any annotation, is a fairly well studied problem in machine learning community. Examples include dictionary learning [19], independent component analysis [13], auto-encoders [3], matrix factorization [26], and various forms of clustering [11]. We can use K-means algorithm to group an unlabeled data set into k clusters, whose centroids can be used to produce features [6]. Unsupervised dictionary learning exploits the underlying structure of the unlabeled data to optimize dictionary elements. An example of unsupervised dictionary learning is sparse coding, which aims to learn sets of over-complete bases to represent data efficiently [19].

Recently deep learning methods trained in a supervised way have dramatically improved the state of the art performance on a variety of computer vision tasks. Since supervised deep learning model is capable of learning high-performance visual representations, what about unsupervised deep learning model? Exemplar CNN [8] proposes a method for training CNNs [18] through a surrogate task automatically generated from unlabeled images. DCAGN [21] identified a family of CNN architectures suitable for the adversarial learning framework (GAN) [9] which has a wide application in image generation.

Another popular method is to train auto-encoders that learns representations based on an encoder-decoder paradigm. Denoising auto-encoders [28] tries to reconstruct the input from a corrupted version of it, which make the hidden layer discover more robust features. Sparse auto-encoders can learn useful structures in the input data by imposing sparsity on the hidden units during training. Sparsity may be achieved by regularization terms in the loss function [22]. Contractive auto-encoder [25] adds a regularization term in their loss function that makes the model robust to slight variations of input values. By making strong assumptions concerning the distribution of latent variables, variational auto-encoders [16] inherit auto-encoder architecture for learning latent representations. Stacked what-where auto-encoder [30] attempts to learn a factorized representation that encodes invariance and equivariance, and leverage both labeled and unlabeled data to learn this representation in a unified framework. The ladder network [24] contains several lateral shortcut connections from the encoder to decoder at each level of the hierarchy, and the lateral shortcut connections allow the higher levels of the hierarchy to focus on abstract invariant features.

3 APPROACH

The scale-space representation we use is the Laplacian pyramid [4]. After reviewing this, we introduce our LPAE model which integrates multiple deep CAEs into the framework of a Laplacian pyramid.

3.1 Laplacian Pyramid

The Laplacian pyramid is a linear invertible image representation consisting of a set of band-pass images, spaced an octave apart, plus a low-frequency residual. The first step in Laplacian pyramid coding is to low-pass filter the original image \(g_0\) to obtain image \(g_1\), which is considered a “reduce” version of \(g_0\) since both resolution and sample density are decreased. In a similar way we form \(g_2\) as a reduced version of \(g_1\), and so on. Filtering is performed by a procedure equivalent to convolution with one of a family of local, symmetric weighting functions. An important member of this family resembles the Gaussian probability distribution, so the sequence of images \([g_0, g_1, \ldots, g_n]\) is called the Gaussian pyramid. Suppose we have selected the 5-by-5 generating kernel \(w\), the level-to-level averaging process is performed by the function REDUCE as below:

\[
g_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) \ast g_{l-1}(2l + m, 2j + n)
\]

where \(i, j\) denote the coordinate of the pixel.

We define a function EXPAND as the reverse of the function REDUCE. Its effect is to expand an \((M+1)\)-by-\((N+1)\) image into a \((2M+1)\)-by-\((2N+1)\) image by interpolating new node values.
Thus, for $0 < k < l$, which can be used to reconstruct the corresponding Gaussian pyramid. A typical architecture of LPAE is shown in Figure 1. We to learn a family of hidden representations for the Laplacian pyramid, this sum.

The Laplacian pyramid is a sequence of difference images

$$l_k = g_k - \text{EXPAND}(g_{k+1})$$

since there is no image $g_{n+1}$ to serve as the prediction image for $g_n$, we say $l_n = g_n$.

### 3.2 Laplacian Pyramid Auto-encoders

Suppose we have a Laplacian pyramid $\{l_0, l_1, \ldots, l_n\}$ and the corresponding Gaussian pyramid $\{g_0, g_1, \ldots, g_n\}$, the aim of our model is to learn a family of hidden representations for the Laplacian pyramid, which can be used to reconstruct the corresponding Gaussian pyramid. A typical architecture of LPAE is shown in Figure 1. We use $E_k(\cdot)$ and $D_k(\cdot)$ to denote the encoding network and decoding network at level $k$, separately. The hidden representation $h_k$ is the output of $E_k(\cdot)$.

$$h_k = \begin{cases} E_k(l_k, h_{k+1}), & k \neq n \\ E_k(l_k), & k = n \end{cases}$$

For each sub-network, the loss function is as below.

$$\text{loss}_k = \|g_k - D(h_k)\|_2$$

And the total loss is the sum of losses at all levels.

$$\text{loss} = \sum_{k=0}^{n} \text{loss}_k$$

### 3.3 Details of the Network Architecture

As shown in Figure 2, we use a CNN to encode the input, and employ a deconvolutional net (Deconvnet) [29] to produce the reconstruction at each level. The numbers in each cell denote the size of receptive field, number of feature maps and stride. For example, “3*3*64, 1” at the top left means a convolutional layer with 3-by-3 receptive field, 64 feature maps and a stride of 1 pixel for...
Table 3: Classification Performance on STL10 and CIFAR10

| Algorithm       | STL10   | CIFAR10  |
|-----------------|---------|----------|
| 2-scales LPAE   | 71.9%   | 79.4%    |
| 3-scales LPAE   | 73.3%   | -        |
| 4-scales LPAE   | 72.3%   | -        |
| Deep CAE I      | 70.9%   | 76.5%    |
| Deep CAE II     | 67.7%   | 73.1%    |
| Conv. K-means Network [5] | 60.1%   | 82.0%    |
| HMP [2]         | 64.5%   | -        |
| View-Invariant K-means [12] | 63.7%   | 81.9%    |
| Exemplar CNN [8] | 74.2%   | 84.3%    |
| SWW Auto-encoder [30] | 74.3%   | -        |
| Supervised state of the art | 70.1%[27] | 96.5%[10] |

Each dimension of input. All convolutional layers and deconvolutional layers use ReLU nonlinearity, which is omitted in the notation. No fully connected layer has been used, which helps handle input data of different size. Each layer is followed by a batch normalization layer. Batch normalization (BN) layer [14] is important for the training of deep models based on the CAE, and we give practical proof in the experimental results. We up-sample the outputs of each CNN, and concatenate them with feature maps of a convolutional layer in the next level. This data flow aims to reverse the process of Laplacian pyramid generation.

4 EXPERIMENTS

To compare our approach to deep CAEs and other unsupervised feature learning methods, we report classification results on the STL-10 [6] and CIFAR-10 [17].

4.1 Datasets

STL-10 contains 96x96 pixel images and relatively less labeled data (5,000 training samples, 100,000 unlabeled samples and 8,000 test samples). It is especially well suited for unsupervised learning as it contains a large set of 100,000 unlabeled samples. In all experiments, we trained our model and Deep CAEs from the unlabeled subset of STL-10, and used the encoding parts as generic feature extractors. The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. Since the resolution of CIFAR-10 images is low, we only evaluated 2-scales LPAE on CIFAR-10.

4.2 Experimental Setup

To make a thorough evaluation of our model, we worked with three network architectures of different scales. We have shown the network architecture of 4-scales LAPE in Figure 2. By removing the level 3 of 4-scales LAPE, we get 3-scales LAPE. Likewise, we can get 2-scales LAPE. We use deep CAEs as baselines, and the architectures are shown in Table 1 and Table 2. Each layer is followed by a batch normalization layer.

No pre-processing was applied to training images besides ZCA whitening. All models mentioned above were trained with mini-batch Adaptive Moment Estimation (Adam) [15] with a mini-batch size of 50. All weights were initialized from a zero-centered Normal distribution with standard deviation 0.02. Learning rate was set to 0.001 in all models. All models were implemented in TensorFlow 1.3 [1].

At test time we applied the encoding network of each model as a generic feature extractor. To the feature maps of each layer we applied the max-pooling method that is commonly used for STL-10 and CIFAR-10 dataset. The pooled features were then flattened into vectors, and we trained a softmax classifier on these feature vectors. For all models, max-pooling results in 16 or 9 values per feature map.

4.3 Classification Performance

In Table 3 we compare LAPE to several unsupervised feature learning methods, including the current state of the art on each dataset. We also list the state of the art for methods involving supervised feature learning (which is not directly comparable). In Table 3 we reported the best performance of LPAEs and Deep CAEs we have achieved in the experiments. Figure 3 plots the performance of LPAEs and Deep CAEs (with or without BN layers) against the number of epochs.

Observations are as follows. First, LPAE methods outperformed deep CAEs which didn’t consider the scale-space representation. Second, LPAE methods didn’t achieve the state of the art, but it still outperformed several baselines on STL10. LPAE methods performed poor on CIFAR10, which is likely due to the low resolution of CIFAR10 images. Apparently, low resolution fails to provide significant scale-space information. Third, the performance of LPAE didn’t increase with the number of scales. In our opinion, this result indicates that determining the number of scales in LPAE needs the empirical knowledge. Fourth, the performances of LPAEs and deep CAEs achieved the best very fast and stayed stable after 10 epochs. Fifth, BN has a very important influence on the performance of LPAEs and deep CAEs. The result at epoch 0 in Figure 3 is the performance of random filters. It’s clear that LPAEs and deep CAEs perform worse than the random filters without BN layers, and using BN layers can lead to a drastic difference of performance.

4.4 Reconstruction Loss

It’s clear that the Laplacian pyramid and BN layers have important influences on the classification performance. Thus, it’s reasonable to study their influences on the reconstruction. Figure 5 and Figure 6 plot the reconstruction loss of LPAEs at the bottom level. In Figure 6 we compared LPAEs with deep CAEs. Apparently, LPAEs performed much better than deep CAEs on image reconstruction, indicating that LPAEs can express more information content than deep CAEs. Meanwhile, LPAE methods performed better than deep CAEs on the classification tasks, indicating that LPAEs achieved better representation abstraction. These experimental results support the viewpoint we mentioned in the introduction section. These results also suggest that LPAE is much more suitable than deep CAE for image generation tasks. Figure 4 and Figure 5 show the influence of BN layers on reconstruction. Removing BN
Figure 3: Top left: LPAE with BN layers VS LPAE without BN layers on STL10; Top right: LPAE VS Deep CAEs on CIFAR10; Bottom left: Deep CAEs with BN layers VS Deep CAEs without BN layers; Bottom right: LPAEs VS Deep CAEs on STL10.

Figure 4: Reconstruction loss of deep CAEs.

layers leads to failure of training, except for deep CAE II. Apparently the backpropagation couldn’t work in the training of LPAEs and deep CAE I without BN layers, which also explained their poor classification performances after removing BN layers. LPAE model has multiple input and output paths, which increases the difficulty of training. Using BN layers led to a drastic drop in reconstruction loss and helped stabilize training. Deep CAE II showed a different behavior from other models after removing BN layers. Using BN or not has no influence on its reconstruction performance. Let’s go back and look at its classification performance. From Figure 3 we can see that its accuracy curve first went up and then went down after removing BN layers, which was different from other models. In our opinion this observation suggested that the deeper
architecture helped training and resulted in a relatively good reconstruction performance, but it’s unstable. One possible explanation is that BN helps gradient flow in deeper models. Overall, BN is critical to get deep networks to begin learning.

5 CONCLUSIONS
In this paper we embed deep auto-encoders into the framework of Laplacian pyramid, and apply the model to unsupervised representation learning. Experiments have shown some interesting results which benefit the research and practical applications of deep auto-encoders approaches.

ACKNOWLEDGMENTS
Thanks for all the supports we have.

REFERENCES
[1] Martin Abadi, Paul Barham, and et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016. 265–283.
[2] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. 2012. Unsupervised Feature Learning for RGB-D Based Object Recognition. In Experimental Robotics - The 13th International Symposium on Experimental Robotics, ISER 2012, June 18-21, 2012, Quebec City, Canada. 387–402.
[3] H. Bourlard and Y. Kamp. 1988. Auto-association by multilayer perceptrons and singular value decomposition. Biological Cybernetics 59, 4 (01 Sep 1988), 291–294.
[4] Peter J Bart and Edward H Adelson. 1983. The Laplacian Pyramid as a Compact Image Code. IEEE Transactions on Communications 31, 4 (1983), 532–540.
[5] Adam Coates and Andrew Y. Ng. 2011. Selecting Receptive Fields in Deep Networks. In Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain. 2528–2536.
[6] Adam Coates, Andrew Y. Ng, and Honglak Lee. 2011. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. 215–223.
[7] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014 647–655.
[8] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin A. Riedmiller, and Thomas Brox. 2016. Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks. IEEE Trans. Pattern Anal. Mach. Intell. 38, 9 (2016), 1734–1747.
[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. CoRR abs/1406.2661 (2014).
[10] Benjamin Graham. 2014. Fractional Max-Pooling. CoRR abs/1412.6071 (2014). 
http://arxiv.org/abs/1412.6071
[11] J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society 28, 1 (1979), 100–108.
[12] Ka Yu Hui. 2013. Direct Modeling of Complex Invariances for Visual Object Features. In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013 352–360.
[13] Aapo Hyvärinen and Erkki Oja. 2000. Independent component analysis: algorithms and applications. Neural Networks 13, 4-5 (2000), 411–430. https://doi.org/10.1016/S0893-6080(00)00026-5
[14] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. 448–456.

[15] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. CoRR abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[16] Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. CoRR abs/1312.6114 (2013). arXiv:1312.6114 http://arxiv.org/abs/1312.6114

[17] A Krizhevsky and G Hinton. 2009. Learning multiple layers of features from tiny images. 1 (01 2009).

[18] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Handwritten Digit Recognition with a Back-Propagation Network. In Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]. 396–404.

[19] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. 2006. Efficient sparse coding algorithms. In Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006. 896–900.

[20] Tony Lindeberg. 1994. Scale-space theory: a basic tool for analyzing structures at different scales. Journal of Applied Statistics 21, 1 (1994), 225–270.

[21] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. CoRR abs/1511.06434 (2015). arXiv:1511.06434 http://arxiv.org/abs/1511.06434

[22] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 – July 2, 2011. 833–840.

[23] Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakkola. 2004. Maximum-Margin Matrix Factorization. In Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]. 1529–1536.

[24] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. 2013. Multi-Task Bayesian Optimization. In Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. 2004–2012.

[25] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. Journal of Machine Learning Research 11 (2010), 3371–3408.

[26] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Robert Fergus. 2010. Deconvolutional networks. In The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010. 2528–2535.

[27] Junho Jake Zhao, Michael Mathieu, Ross Goroshin, and Yann LeCun. 2015. Stacked What-Where Auto-encoders. CoRR abs/1506.02351 (2015). arXiv:1506.02351 http://arxiv.org/abs/1506.02351