Chapter 6

Generic Object Classification

6.1 Introduction

Chapters 3 and 4 have introduced two representations that have the same invariance to rigid-motion but with very different discriminative power. As argued in Section 4.2, separable representations process the different paths of internal layers completely independently, and by doing so, they lose all the information contained in the local correlation of the different paths of these layers. On the contrary, joint representations consider an internal layer as a multidimensional function. In the proposed joint rigid-motion scattering, an internal layer was described as a set of functions, or orbits, of the rigid-motion group, but internal layers can also be interpreted as a function of some much larger group, which eventually is not hardcoded but learned.

Deep networks [HS06, LLB+98] are representation with little a priori information on the data. They can be trained to discriminate different classes of signal, and by doing so, they possibly learn some class invariant. Deep networks consist in a cascade of linear operators, whose weights are learned, and non-linear operators. For large images, deep networks, as is, would involve too many weights to be learned. Convolutional neural networks (ConvNets) [LBD+89, LKF10, KSH12, DCM+12, ZF13] take advantage of the image structure to limit the number of weights while maintaining the ability to learn generic invariance.

During an internship at Google, I have had the opportunity to experiment large scale generic object recognition with distbelief, Google's distributed implementation of deep networks [DCM+12]. Those networks involve relatively expensive convolutions, some of which whose trained weights have appeared to be highly redondant. From this finding, we have adapted the efficient rigid-motion convolution introduced in section 4.3.1 into a generic separable convolutional layer. This separable convolutional layer factorizes multidimensional convolutions into purely spatial convolutions followed by pointwise matrix multiplications along the path variable. Early experiments have shown that such separable convolutional layers require less data to train, less resources per data and result in identical to slightly

better final accuracy on ImageNet ILSVRC2012 classification dataset. Section 6.2 details these separable convolutions and briefly describes the classification experiments.

In collaboration with Edouard Oyallon, we have also experimented with pure scattering, that is without any learning of the weights of internal layers, on generic object classification datasets such as Caltech 101 and 256. Oyallon [OMS14] has shown that a slightly modified version of the joint rigid-motion scattering introduced in chapter 4 can achieve similar performances as the first layers of fully trained deep networks. These experiments are described in section 6.3. If the performances of scattering are still behind a fully trained deep network with 7 layers, these experiments open the possibility to simplify deep neural network learning by initializing the first few layers with wavelets.

6.2 Separable Convolutions for Generic Deep Networks

This sections briefly reviews the convolutional deep network for generic image classification [KSH12, DCM$^+$12, ZF13]. These networks have obtained state-of-the-art results on most large scale generic image classification datasets, such as ImageNet ILSVRC2012. Section 6.2.1 reviews the dense convolutions that implement the linear operators of the first layers of [KSH12, DCM$^+$12, ZF13]. Section 6.2.2 observes that some of the weights in dense convolutional layers are highly redundant and proposes a separable convolutional layer inspired by the rigid-motion convolution of Chapter 4. Some preliminary experiments on ImageNet have shown that such separable convolutions can achieve similar or slightly better accuracy at a lower computational cost. This is related to recents works [DSD$^+$13, LCY13] where other forms of factorizations of the first convolutional layers are proposed.

6.2.1 Dense Multidimensional Convolutions

ConvNets, as described in [KSH12, DCM$^+$12, ZF13] consist in a cascade of linear operators $\mathcal{W}_1, \dots, \mathcal{W}_m$ intertwined with non-linearties f. Given an input image x, they compute successive layers $\Phi_m x$ defined as

$$\Phi_m x = f(\mathcal{W}_m f(\mathcal{W}_{m-1} \dots f(\mathcal{W}_1 x))) \quad (6.1)$$

$$= f(\mathcal{W}_m \Phi_{m-1} x). \quad (6.2)$$

A Layer $\Phi_m x$ is indexed by a spatial position u and a depth d that plays a role similar to scattering paths p_m. Let us call N_m, D_m the size and the depth of a layer $\Phi_m x$, which are respectively the number of samples for u, d in this layer . A fully connected linear operator, that is without any convolutional structure, that connect $\Phi_m x$ to $\Phi_{m+1} x$ would require $N_m D_m N_{m+1} D_m$ weights which is impractical for first layers where both N_m and D_m are large. Convolutional networks reduce the number of weights by leveraging the structure of images. They enforce locality by connecting an output position only to the input position which lies within a window of size Q_m centered around the output position. Locality

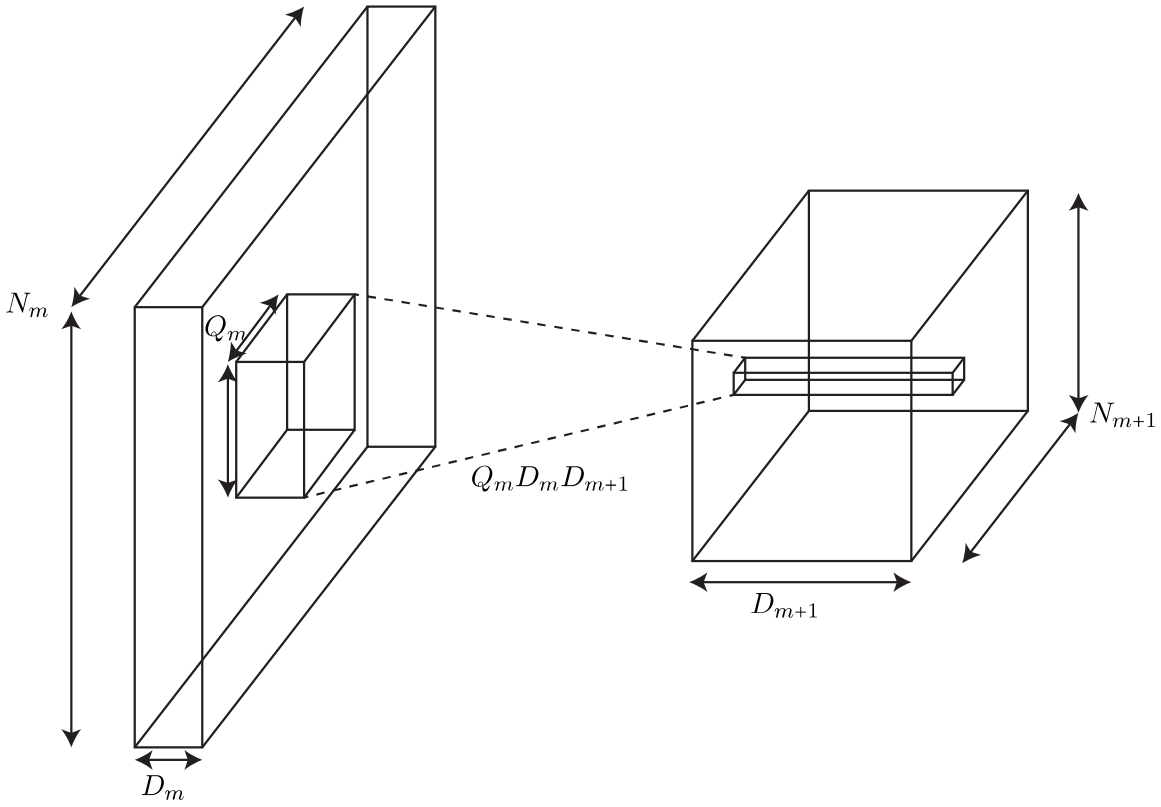

Figure 6.1: Dense multidimensional convolutions in deep networks for image classification [KSH12, DCM+12, ZF13]. A block of $Q_m \times D_m$ values in the input layer is considered as a vector, which is multiplied by a a matrix with $Q_m \times D_m \times D_{m+1}$ weights to generate a vector of dimension D_{m+1} in the output layer. This linear operator is local in the sense that the input block is much smaller than the input image, and convolutional in the sense that the same matrix is applied to every block, independently of their location.

reduces the number of weights to $Q_m D_m N_{m+1} D_{m+1}$, that is by a factor of N_m/Q_m. Q_m is typically equal to 11×11 for the first convolutional layers and 3×3 for the last convolutional layers, whereas D_m typically increases from 3 to 2048. The number of weights is further reduced by convolution, that is by forcing the weights corresponding to windows centered around different output position to be equal. Convolution further lowers the number of weights to $Q_m D_m D_{m+1}$ which then does not depend upon the size of the image. Locality and convolutions are essential to the practicability of back-propagation algorithm, which would otherwise require an enormous amount of processing time and memory to learn the weights. The dense convolution is computed as

$$\mathcal{W}_m \Phi_m x(u, d_{m+1}) = \sum_{\substack{0 \le d_m < D_m \\ v \in Q_m}} w_{v,d_m,d_{m+1}} \Phi_m x(u+v, d_m). \tag{6.3}$$

This dense convolution is illustrated in figure 6.1.

6.2.2 Separable Convolutions

The dense convolutions (6.3) can be interpreted as a sum of two dimensional convolutions with filters $w_{\cdot,d_m,d_{m+1}}$

$$\mathcal{W}_m \Phi_m x(u, d_{m+1}) = \sum_{0 \le d_m < D_m} \Phi_m x(\cdot, d_m) \star w_{\cdot,d_m,d_{m+1}}(u). \tag{6.4}$$

Figure 6.2 shows the filters that are learned in the first two layers of a network similar to [KSH12]. One can denote that for a given output depth, these filters are highly redundant along the input depth. This suggest to factorize the matrix of weights $w_{v,d_m,d_{m+1}}$ into purely spatial convolutions with weights w followed by a pointwise matrix multiplication with weights $\mathring{w}$ along the depth variable. A separable multidimensional convolution, as opposed to the dense multidimensional convolution (6.3), is defined as

$$\widetilde{\mathcal{W}}_m \Phi_m x(u, d_{m+1}) = \sum_{\substack{0 \le k_m \le K_m \\ 0 \le d_m}} \mathring{w}_{k_m,d_m,d_{m+1}} \left(w_{\cdot,k_m,d_m} \star \Phi_m x(\cdot, d_m)(u) \right) \tag{6.5}$$

The separable multidimensional convolution (6.5) is very similar to the separable group convolution (4.30) introduced in Chapter 4. Each two dimensional signal of $\Phi_m x(\cdot, d_m)$ is filtered with K_M two dimensional filters $w_{\cdot,k_m,d_m}$ which play the role of warped filters $y(A'^{-1}\cdot)$ in the separable group convolution (4.30). The output of those first convolutions is an intermediate layer of depth $D_m \times K_m$, therefore we call K_m the depth-multiplier. In the affine wavelet transform (4.40-4.43) of Chapter 4, the depth-multiplier was typically LJ, which was the cardinality of the first wavelet family $\{\phi_J, \psi_{l,j}\}_{0 \le l < L,\ j < J}$. The intermediate layer is then retransformed by multiplying its pointwise vector of dimension $1 \times K_m \times D_m$ with a matrix $\mathring{w}$ of size $D_{m+1} \times (K_m \times D_m)$, which results in a output layer of depth D_{m+1}. This matrix multiplication with $\mathring{w}$ is similar to the filtering with $\mathring{y}$ along A in the

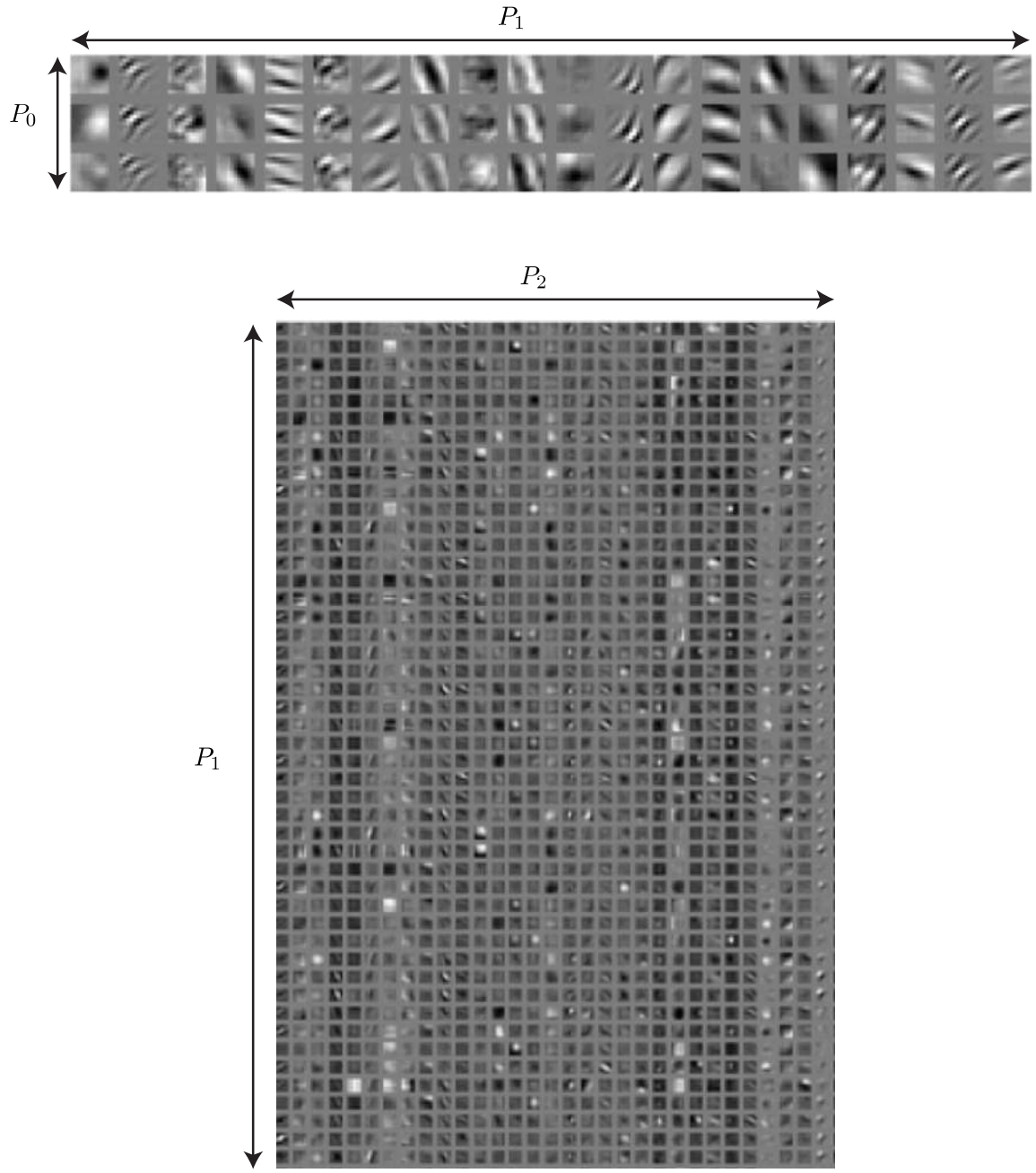

Figure 6.2: A visualization of the learned weights of the first two layers of deep networks similar to [KSH12, DCM+12, ZF13]. Rows corresponds to input depth and columns corresponds to output depth. Computing one output depth of a dense convolution, 6.3 is equivalent to compute a two dimensional convolution of each input depth $\Phi_m(., d_m)$ with the two dimensional filter $w_{.,d_m,d_{m+1}}$ located at rows d_m and column d_{m+1} in these arrays of filters.

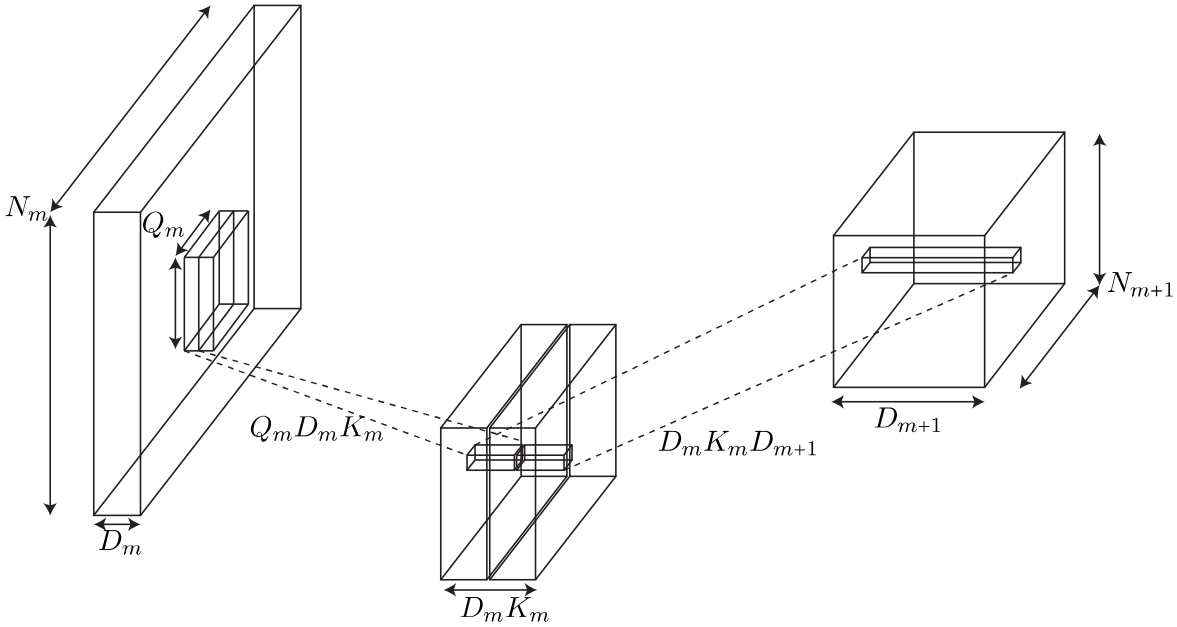

Figure 6.3: The proposed separable multidimensional convolution. First, each input depth is independantly filtered with K_m filters of compact support Q_m, which results in $Q_m D_m K_m$ weights. The result is an intermediate layer of depth $D_m K_m$. At each position u, the vector consisting of all the depths of the intermediate layer is retransformed with a matrix multiplication, which results in $D_m K_m D_{m+1}$ weights and an output layer of depth D_{m+1}.

separable affine convolution (4.30), except that in the separable convolutional layer, the underlying group is unknown and therefore may not have a convolutional structure. Figure 6.3 illustrates the architecture of a separable convolutional layer.

A separable convolutional layer can have the same input and output depth and can therefore act as a replacement of a dense convolutional layer, with less redundancy. The number of free parameters can be adjusted by the depth multiplier K_m, which can be chosen arbitrarily without changing the input and output depth. A separable convolutional layer has $Q_m D_m K_m$weights in its first spatial component and $K_m D_m D_{m+1}$ weights in its second depth component and has therefore a total of $K_m D_m (Q_m + D_{m+1})$ weights. This has to be compared to a dense convolutional layer with $Q_m D_m D_{m+1}$ free parameters. In particular, one can note that the dependency in Q_m, D_{m+1} goes from $Q_m D_{m+1}$ for a dense layer to $Q_m + D_{m+1}$ for a separable layer.

m	D_m	D_{m+1}	Q_m	K_m	#dense param	#separable param	ratio
1	3	96	7×7	4	14112	1740	87%
				8		3480	75%
2	96	256	5×5	4	614k	107k	82%
				8		215k	64%

Table 6.1: Comparison of the number of parameters for the first layers in the network of [ZF13] with dense versus separable convolutions. m is the index of the layer, D_m is the input depth, D_{m+1} is the output depth, Q_m is the spatial window sizes, K_m is the depth multiplier for separable layers. A separable architecture has fewer parameters and therefore requires less computational resources per data. Experiments on ImageNet have shown that a network with separable layers also requires less data to achieve similar or slightly better performance than a network with dense layers with the exact same architecture.

6.2.3 ImageNet Classification with Separable Convolutional Layers

Table 6.1 gives the number of parameters for the network used in [ZF13] which has obtained state-of-the-art results on Imagenet, where the first two layers are implemented with dense or separable convolutional layers. During the second part of my internship at Google, I have replaced convolutional dense layers with separable layers in the distbelief [DCM+12] implementation of the deep network described in [ZF13]. We have trained it on ImageNet ILSVRC2012, which contains 1000 classes, 1.2M training images and 150k testing images of generic objects with clutter. On this dataset, we have recorded that the separable versions required 20% fewer steps to converge to an identical or slightly better final accuracy, compared to the dense version. This suggests that separable convolutions are an appropriate way to factorize dense convolutions without losing expressive power, and that the architecture benefits from this factorization since it then needs less data to achieve the same accuracy. The training and inference time per step was also smaller.

6.3 Object Classification with Joint Scattering

Convolutional networks (ConvNets) such as the one described in [KSH12, DCM+12, ZF13] have obtained state-of-the-art results on all large scale image classification datasets such as ImageNet. It appears that the first layers of a deep network trained on ImageNet also perform very well on smaller datasets such as CalTech101 or CalTech256 [GDDM13, ZF13, DJV+13]. This suggests that these first layers capture some intrinsic properties of real world images.

Other experiments, mainly conducted by Oyallon [OMS14], have shown that a slightly modified version of the joint rigid-motion scattering introduced in Chapter 4 can also obtain results that are competitive with the first layers of a ConvNet trained on ImageNet. This suggest that the first layers of ConvNets capture some basic geometric properties of the