UNIFIED ANALYTIC FORMS FOR CONVOLUTIONAL NEURAL NETWORKS AND WAVELET FILTER BANKS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper provides a unified analytic framework integrating expressions of several variants of convolutional neural networks and wavelet filter banks. The expressions are derived recursively, from downstream to upstream layers, for the sequences of features returned at the nodes of a general form of network architecture. The inspiring framework for the derivation of these expressions is that of the so-called M-band convolution filter banks. In addition with the inter-layer inter-node expressions, activation sequences of convolutional neural networks also are mathematically described by suitable algebraic path representations.

1 INTRODUCTION

Convolutional Neural Networks (CNNs) are powerful analysis tools in discriminant representations of images by providing various salient feature subspaces Krizhevsky et al. (2012), Jia et al. (2014), Simonyan & Zisserman (2014), Chollet et al. (2015), Szegedy et al. (2015), Abadi et al. (2016). After a decade of fine tuning and fine parameterization of CNNs with respect to a wide range of data and applications, their relevancy is now heuristically proven and we now reach the period where the scientific community is focusing on the synthesis of various theories for CNN design, analysis, validation and optimization, especially for deep networks.

Among such theories, we can mention the analysis of deep layer properties depending on the type of functionals involved. For instance, in Pal et al. (2019), dropout operators are shown to be responsible of regularization in deep learning networks whereas in Aberdam et al. (2019), this regularization is intuited through sparsity concerns. Several other examples can be provided and it is reasonable at present time to admit that there is no single universal theory for addressing deep CNN mathematics: on the one hand, if one is interested in designing invariant features, then the scattering approach of Mallat (2016) can help understanding CNN stabilization through iterations and weight propagation to a somewhat translation-rotation invariant transform. On the other hand, if one seeks classification task (finding the best separators in a suitable space), then stochastic optimization has to be pushed to its tricky recipes in order to avoid getting stuck in non-desirable local minima of the global objective function attached with the underlying CNN.

Actually, a brief literature tour of mathematical theories of deep CNNs does not allow to quickly retrieve deep equation machinery, which is substantially the most informative way of highlighting statistical properties of such networks. Literature has rather focused on analyzing activations of the networks during or after the learning stage, insightful case studies can be found in Papadopoulos et al. (2016), Tran & d'Avila Garcez (2018), Kim et al. (2018), among other references. In addition with the proven interest of such data/application-based activation analysis in understanding the salience captured by CNNs, unrolling deep equations can also help in deriving general CNN properties.

The main issue in addressing deep CNN analytical forms is that deep networks are huge in terms of the variety of architectures and the possible number of parameters. But starting with a background on the theory of filter banks, especially the one associated with the so-called M-Band Discrete Wavelet Packet Transform (M-DWPT, which is likely the closest theory to deep CNN architectures), this paper shows that deep CNN equations can fit on a short paper.

The organization of the paper is the following. Section 2 recalls basics of M-DWPT and works out its deep inspired multiserial generalization. This section is written for a sake of both highlighting similarities and bringing together, multiserial linear filter bank and CNN architectures. Section

3 provides analytic expressions of standard and expanded CNNs with respect to their congruent generalized multiserial filter banks. Section 4 derives algebraic descriptions of deep network paths for both the multiserial linear and CNN frameworks. Finally, Section 5 concludes the work by proposing a discussion and highlighting some prospective exploitation of multiserial linear and deep CNN analytical expressions.

Insights: The motivation in unifying CNNs and wavelet packet transforms in the same analytical framework can help us building interacting semi-supervised functionals involving fixed (wavelets) and learnable (CNNs) filters. It can also help in the design of more generalizable CNNs which can avoid the limitations highlighted in Atto et al. (2020): CNNs tend to get specialized by passing information that is significant in the training dataset and blocking non-relevant information with respect to the classification loss so that if a given information/frequency is absent or irrelevant for classifying the training dataset, then the corresponding CNN will be have some limitation in a transfer learning scenario where this information/frequency now become important.

Guidelines: Sections are organized in a bottom up dissertation starting from M-DWPT and linking its tree structure to that of CNNs *via* multiserial filter bank trees. For both multiserial linear filter banks and CNNs, we focus on deriving network equations without considering the final output: the latter (as well as its upstream layers) depends on the application considered and our aim is concentrating on the decomposition nodes that are common to a wide range of applications. Indeed, note that if the output is associated with regression, then there is no need of a softmax layer. In addition, if the output is classification, then there is no need to reconstruct with respect to a feature space having the same dimensionality as the input space. Furthermore, expressions will be given without any concern on the recurrences involved by the learning stage (iterative weight updates requiring unnecessary clumsy indices that are useless in operational deployment and testing purposes).

Regarding network architecture, we will focus on homogeneous ones: adaptation to group of convolutions or parallel networks is generally motivated by the sake of calculus distribution over several workstations rather than for the sake of performance. In addition, the terminology of *fully-connected layer* will not be used as such a layer can be explained by a suitable selection of convolution operators. Finally, we choose without loss of generality, a monochanel image convention for the input layer: adaptation to multichannel input images can be obtained straightforwardly by assuming that the first layer performs multichannel image coloration from a given set of color or spectral filters on a monochannel spatial field.

2 General frameworks for the design of multiserial linear filter bank architectures

2.1 Standard (uniserial) M-DWPT

Given a natural number M larger than or equal to 2, the M-DWPT achieves an orthogonal decomposition of a functional space $\mathbf{U}$ via a double-indexed sequence $\{\mathbf{W}_{j,n}\}_{j\in\mathbb{N},n=0,1,\ldots,M^j-1}$ of nested functional subspaces (see examples given in Figures 1 and 2 for $M = 2, 3$ respectively), where $\mathbb{N} = \{1, 2, \ldots\}$ stands for the set of natural numbers. Each $\mathbf{W}_{j,n}$ is the closure of a space spanned by wavelet packet functions denoted $\{W_{j,n,k} : k \in \mathbb{Z}\}$ and forming an orthonormal basis of the vector space $\mathbf{W}_{j,n}$. Index j is the decomposition level and the shift parameter n has values restricted to $\{0, 1, \ldots, M^j - 1\}$. The standard DWPT corresponds to the particular case where $M = 2$.

As illustrated in Figures 1 and 2, the M-DWPT decomposition of the function space $\mathbf{U}$ consists in first the splitting of $\mathbf{U}$ into M orthogonal subspaces: $\mathbf{U} = \bigoplus_{m=0}^{M-1} \mathbf{W}_{1,m}$, and then recursively applying the following splitting $\mathbf{W}_{j,n} = \bigoplus_{m=0}^{M-1} \mathbf{W}_{j+1,Mn+m}$, for every natural number j and every $n = 0, 1, 2, \ldots, M^j - 1$. This splitting of $\mathbf{U}$ is uniserial in the sense that in practice, it is performed by applying recursively exactly the same M wavelet[1] filters $(h_0, h_1, \ldots, h_{M-1})$ at any given node (j, n) of the M-DWPT tree. Due to this splitting scheme, layer j of an M-DWPT has exactly M^j outputs. For more details on the practical details leading to the implementation M-DWPT, the reader is asked to refer to Steffen et al. (1993).

[1] Wavelet framework is useful for constructing filters that are complementary in the spectral domain.

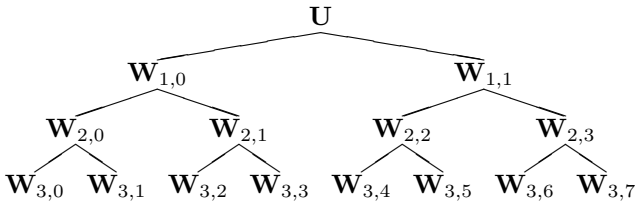

Figure 1: 2-DWPT decomposition tree of $\mathbf{U}$ down to resolution level $j = 3$. A sequence $\{h_0, h_1\}$ is used for recursive decomposition of the tree nodes.

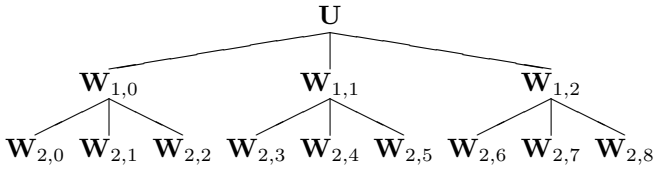

Figure 2: 3-DWPT decomposition tree of $\mathbf{U}$ down to resolution level $j = 2$. A sequence $\{h_0, h_1, h_2\}$ is used for recursive decomposition of the tree nodes.

2.2 First relaxation of M-DWPT: Multiserial splits and analogy with CNN

Starting from the standard M-DWPT and noting that most relevant deep CNN architectures have variable numbers of filters in their layers, we can consider a first extension of M-DWPT consisting of decomposing $\mathbf{U}$ by using a multiserial set $\{(h_{j,0}, h_{j,1}, \ldots, h_{j,M_j-1}) : 1 \leqslant j \leqslant J\}$ where $(h_{j,0}, h_{j,1}, \ldots, h_{j,M_j-1})$ is used at decomposition level (layer) j. The corresponding transform is called MultiSerial DWPT (MS-DWPT). An illustration of MS-DWPT is presented in Figure 3.

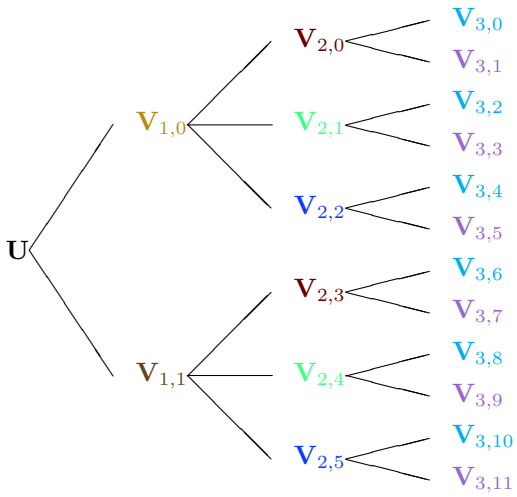

Figure 3: MultiSerial DWPT tree with $M_1 = 2$, $M_2 = 3$ and $M_3 = 2$. Three sequences $\{h_{1,0}, h_{1,1}\}$, $\{h_{2,0}, h_{2,1}, h_{2,2}\}$ and $\{h_{3,0}, h_{3,1}\}$ are used for recursive decomposition of the tree nodes. The 3 hidden layers have respectively 2, 6 and 12 nodes, but a total of $M_1 + M_2 + M_3 = 7$ different convolution filters have been used. This structure will be associated to an expanded CNNs having $j = 1, 2, 3$ hidden layers involving respectively 2, 3 and 2 convolution layers.

The MultiSerial DWPT inherits several basic DWPT properties such as: sequence $(h_{j,0}, h_{j,1}, \ldots, h_{j,M_j-1})$ is associated with scaling and wavelet convolution filters for any $j \in \{1, 2, \ldots, J\}$. Standard bivariate wavelet-based filter construction also assumes 2D separability and identical 1D filter on both variables, that is:

$$h_{j,m_j}[p, q] = f_{j,m_j}[p] f_{j,m_j}[q] \tag{1}$$

Moreover, $F_{j,m_j}[\bullet]$ satisfy the following paraunitary conditions: matrix $\left(F_{j,m_j}\left(\omega+\frac{n\pi}{M_j}\right)\right)_{0\leqslant n,m_j\leqslant M_j-1}$ is unitary for every real number ω, where

$$F_{k,m}(\omega)=\frac{1}{\sqrt{M_1}}\sum_{\ell\in\mathbb{Z}}f_{k,m}[\ell]e^{i\ell\omega} \tag{2}$$

is a normalized Fourier transform of the mirror of $f_{k,m}$. Finally, it is assumed that $F_0(0)=1$ to ensure that approximation node is located at the top of any layer.

These conditions follow from the DWPT framework and are motivated by orthogonality constraints: the different outputs are exactly complementary. There is no information redundancy in an MS-DWPT because of its tight frame property.

2.3 Second relaxation: removing wavelet and orthogonality constraints

Wavelet properties (fast decay conditions for both f_0 and F_0 introduced in equation 1 and equation 2), 2D separability constraints and orthogonal splits will constrain significantly the design of convolution filters from MS-DWPT. For bringing closer MS-DWPT and CNN, we remove these constraints from now on. Instead, we suggest a constraint guaranteeing only bounded over-completeness of the representation. This variant is no longer a wavelet framework (vanishing moments are not guaranteed for instance). The corresponding multiserial frame obtained will be called Bounded Overcomplete MS-Discrete Linear Transform (BOMS-DLT) involves a single constraint:

$$1\leqslant\sum_{m=0}^{M_j-1}|H_{j,m}[\xi_1,\xi_2]|\leqslant M_j \tag{3}$$

for every $1\leqslant j\leqslant J$ and $(\xi_1,\xi_2)\in\mathbb{R}^2$ where $H_m(\xi_1,\xi_2)=\sum_{p,q\in\mathbb{Z}}h_m[p,q]\exp\left(-i(p\xi_1+q\xi_2)\right)$. It consists in splitting $\mathbf{U}$ as: $\mathbf{U}\subseteq\bigcup_{m=0}^{M_1-1}\mathbf{V}_{1,m}$ where coefficients of $\mathfrak{I}\in\mathbf{U}$ on space $\mathbf{V}_{1,m}$ with $1\leqslant m\leqslant M_1$ are (activation image):

$$\mathfrak{C}_{1,m}[k,\ell]=\sum_{p,q\in\mathbb{Z}}h_{1,m}[p,q]\mathfrak{I}[k-p,\ell-q] \tag{4}$$

where $\mathfrak{I}$ is the input image and $\mathfrak{C}_{1,m}$ is the convolutional output map (layer 1 and m-th convolution filter). From this first layer, we then iteratively perform the multiserial decomposition: $\mathbf{V}_{j,n}\subseteq\bigcup_{m=0}^{M_{j+1}-1}\mathbf{V}_{j+1,M_{j+1}n+m}$ for every natural number j and every $n=0,1,2,\ldots,M_1M_2\cdots M_j-1$ where

$$\mathfrak{C}_{j+1,M_{j+1}n+m}[k,\ell]=\sum_{p,q\in\mathbb{Z}}h_{j+1,m}[p,q]\mathfrak{C}_{j,n}[k-p,\ell-q] \tag{5}$$

for $1\leqslant m\leqslant M_{j+1}$, where we have assumed that M_{j+1} convolution filters $\{h_{j+1,0},h_{j+1,1},\ldots,h_{j+1,M_{j+1}-1}\}$ are selected at level $j+1$. Note that the corresponding tree architecture is the same as in Figure 3.

Note also that equation 5 can be rewritten by taking $(j+1)$-th strides (s_{j+1},t_{j+1}) into account:

$$\mathfrak{C}_{j+1,M_{j+1}n+m}[k,\ell]=\sum_{p,q\in\mathbb{Z}}h_{j+1,m}[p,q]\mathfrak{C}_{j,n}[s_{j+1}k-p,t_{j+1}\ell-q] \tag{6}$$

equation 6 provides a general framework for multiserial multilayer transforms which include a standard M-DWPT when assuming that: $M_1=M_2=\ldots=M_j=M$, afterwards $s_j=t_j=M_j=M$ and finally, that all M filters considered for the decomposition are wavelet-based and paraunitary.

3 Analytic expressions of CNNs

BOMS-DLT is the natural way of deriving *expanded* CNN analytical forms: the term expanded is used here because standard CNNs already encompass several fusion stages which make them less

writable. First, we will present analytical expressions of expanded versions of the CNNs sharing similar architecture than BOMS-DLT in Section 3.1. Then we will derive in Section 3.2, the analytical expressions of standard CNNs from the expanded ones.

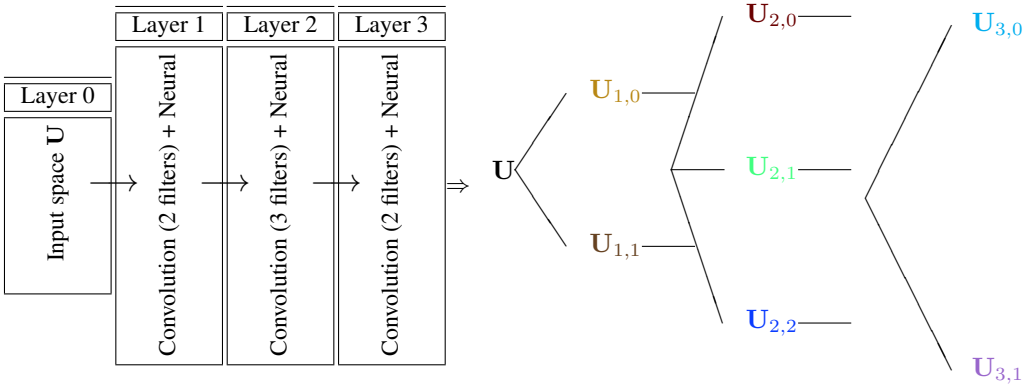

Figure 4: Standard CNN structure composed by 3 hidden layers that have respectively 2, 3 and 2 convolution filters. It involves fusion (sum in general) of convolutional outputs associated with the same filter over the previous layer channels and prior to applying the transfer function: compare the above graph with the expanded one given by Figure 3.

3.1 Analytic expressions of expanded CNNs

It is worth noticing that applying a neural transfer function on every $\mathfrak{C}_{j,n}$ prior to convolution in equation 5 suffices for obtaining a first version of an expanded CNN[2]. Such a CNN is called *expanded* because outputs of layer j are $M_j \times n_{j-1}$ where n_{j-1} is the number of outputs coming from layer $j-1$ and M_j is the number of 2D convolution filters pertaining to layer j.

Let us construct a more intricate expanded CNN step by step, by adding sophistication and heuristics relatively to state-of-the-art standard CNNs. First, let us focus on layer 1: for an expanded CNN, it is composed by M_1 convolution filters as in equation 4, but with the following M_1 neural outputs

$$\mathfrak{D}_{1,m}[k,\ell] = \Upsilon_{1,m}\left(\sum_{p,q\in\mathbb{Z}} h_{1,m}[p,q]\mathfrak{I}[s_1 k - p, t_1 \ell - q]\right) \tag{7}$$

for $m = 0, 1, 2, \ldots M_1 - 1$ where (s_1, t_1) is a couple of *stride* parameters and $\Upsilon_{1,m}$ is a neural transfer function[3]. Iterating from equation 7, a lightweight feedforward deep CNN simply follows as:

$$\mathfrak{D}_{j+1,M_{j+1}n+m}[k,\ell] = \Upsilon_{j+1,M_{j+1}n+m}\left(\sum_{p,q\in\mathbb{Z}} h_{j+1,m}[p,q]\mathfrak{D}_{j,n}[s_{j+1}k - p, t_{j+1}\ell - q]\right) \tag{8}$$

for $j \geqslant 1$, $0 \leqslant m \leqslant M_{j+1} - 1$ and $0 \leqslant n \leqslant M_1 M_2 \cdots M_j - 1$, where $\mathfrak{D}_{1,n}$ has been defined by equation 7. The non-linear transfer functions $\Upsilon_{j+1,M_{j+1}n+m}$ are not necessarily the same (one can combine, from layer to layer and/or in the same layer, sigmoids, hyperbolic tangent, rectified linear unit, *etc.*).

One can add bias/correction terms $\theta_{j+1,M_{j+1}n+m}$ to equation 8 for a heavier model:

$$\begin{aligned}&\mathfrak{D}_{j+1,M_{j+1}n+m}[k,\ell] \\ &\quad = \Upsilon_{j+1,M_{j+1}n+m}\left(-\theta_{j+1,M_{j+1}n+m} + \sum_{p,q\in\mathbb{Z}} h_{j+1,m}[p,q]\mathfrak{D}_{j,n}[s_{j+1}k - p, t_{j+1}\ell - q]\right)\end{aligned} \tag{9}$$

[2]Note that equation 3 is just a useful constraint on BOMS-DLT: it has a consequence on the multiserial filter selection, not on the tree architecture. In practice, such a constraint can be integrated in CNN filter selection if one wishes the latter to not definitely erase frequency information that is absent in the training database for instance.

[3]We can use the same neural function by default and in this case, $\Upsilon_{1,m} \triangleq \Upsilon_1$ which can further be denoted Υ when exactly the same function is used whatever the layer of the CNN.

and further integrate connection dropout *via* binary sequences ($\epsilon_{j,n} = 0$ for removal of node (j, n)):

$$\mathfrak{D}_{j+1,M_{j+1}n+m}[k,\ell] = \Upsilon_{j+1,M_{j+1}n+m}\Bigg(- \theta_{j+1,M_{j+1}n+m} + \sum_{p,q\in\mathbb{Z}} h_{j+1,m}[p,q]\epsilon_{j,n}\mathfrak{D}_{j,n}[s_{j+1}k-p, t_{j+1}\ell-q]\Bigg) \tag{10}$$

3.2 Analytic expressions of standard CNNs

In the expanded version of a CNN given by equation 10 (see also Figure 3), the decomposition space is very huge and saving all features will require a huge latent memory[4]. In practice, standard CNNs integrate a fusion stage for reducing this feature space. The standard CNNs have graphs similar to that of Figure 4 which mask a fusion stage that may sow confusion[5]. The following provides analytical forms of the equations involved by standard CNNs.

Assuming a mono-channel image $\mathfrak{I}$ as in equation 4, the first neuro-convolutional layer of a standard CNN has the same M_1 outputs as the expanded one, that is equation 7 including the possible bias correction terms:

$$\mathfrak{D}_{1,m_1}[k,\ell] = \Upsilon_{1,m_1}\Bigg(- \theta_{1,m_1} + \sum_{p,q\in\mathbb{Z}} h_{1,m_1}[p,q]\mathfrak{I}[s_1k-p, t_1\ell-q]\Bigg) \tag{11}$$

where $m_1 \in \{0, 1, 2, \ldots, M_1 - 1\}$. Note that if we assume a multichannel image (M_0 channels that can be associated with RGB, multispectral, polarimetric SAR, ...), then the expanded CNN will return $M_0 \times M_1$ outputs whereas a standard CNN will replace equation 7 by the following one:

$$\mathfrak{D}_{1,m_1}[k,\ell] = \Upsilon_{1,m_1}\Bigg(- \theta_{1,m_1} + \bigoplus_{m_0=0}^{M_0-1} \sum_{p,q\in\mathbb{Z}} h_{1,m_1}[p,q]\mathfrak{I}[s_1k-p, t_1\ell-q, m_0]\Bigg) \tag{12}$$

where m_0 is current channel and $\oplus$ is a fusion applied over all channels filtered by h_{1,m_1} ($\oplus$ is a standard *sum* in almost all standard CNN architectures available from scientific repositories). Thus, instead of getting $M_0 \times M_1$ outputs, the number of outputs remains equal to the number of filters given in the layer 1: this is the main difference with respect to the expanded CNNs framework presented in Section 3.1.

Let us go back to the monochannel image case and highlight all the differences between expanded and standard CNNs through a layer trip: the second layer of a standard CNN with respect to the monochannel image derives from equation 11 and is subject to

$$\mathfrak{D}_{2,m_2}[k,\ell] = \Upsilon_{2,m_2}\Bigg(- \theta_{2,m_2} + \bigoplus_{m_1=0}^{M_1-1} \sum_{p,q\in\mathbb{Z}} h_{2,m_2}[p,q]\epsilon_{1,m_1}\mathfrak{D}_{1,m_1}[s_2k-p, t_2\ell-q]\Bigg) \tag{13}$$

where $\oplus$ denotes a fusion operator that can be different with that involved in equation 12. As it can be seen from this equation, layer 1 outputs $(\mathfrak{D}_{1,m_1})_{m_1}$ are convolved by using the specific filter h_{2,m_2} and the convolution results are fused, as in equation 12: the layer 1 outputs have been considered as channels of an M_1-variate image.

We finally derive the layer $j + 1$ outputs, given the j-th ones by using the same procedure:

$$\begin{aligned}&\mathfrak{D}_{j+1,m_{j+1}}[k,\ell] \\ &\quad = \Upsilon_{j+1,m_{j+1}}\Bigg(- \theta_{j+1,m_{j+1}} + \bigoplus_{m_j=0}^{M_j-1} \sum_{p,q\in\mathbb{Z}} h_{m_{j+1}}[p,q]\epsilon_{j,m_j}\mathfrak{D}_{j,m_j}[s_{j+1}k-p, t_{j+1}\ell-q]\Bigg)\end{aligned} \tag{14}$$

where $m_{j+1} \in \{0, 1, 2, \ldots, M_{j+1} - 1\}$. Adaptation of the latter equation to scattering transforms as described in Sifre & Mallat (2014) consists in replacing any $\Upsilon_{j+1,m_{j+1}}$ by the modulus operator.

[4]For a network such as AlexNet Krizhevsky et al. (2012) involving $M_1 = 96$, $M_2 = 128$, $M_3 = 384$, $M_4 = 192$ and $M_5 = 128$ convolution filters, then $M_1M_2M_3M_4M_5$ yields more than 100 billions of active convolution nodes in an expanded CNN, in contrast with the thousand active convolution nodes actually available in this standard CNN due to the fusion stage described by equation 12 and equation 14.

[5]The convolution outputs computed on a given node (j, n) by the series of layer filter are considered as image channels and are summed up.

4 ALGEBRAIC DESCRIPTIONS OF DEEP NETWORK PATHS

It follows by comparing equation 10 and equation 14 that the j-th layer of the expanded CNN involves $M_1 M_2 \cdots M_j$ output channels whereas it has only M_j channels in a standard CNN, where M_j denotes the number of convolution filters given in layer j for $1 \leqslant j \leqslant J$. The algebraic descriptions of their corresponding graphs is thus different.

From equation 10, the expanded CNN defines a tree structure that can be described by BOMS-DLT (see graph of Figure 3): a root node $\mathbf{U}$ and an activation sequence that forms a path of the form $\mathcal{P} = \big(\mathbf{U}, \{\mathbf{V}_{j,n(j)}\}_{1 \leqslant j \leqslant J}\big)$. By construction, each $\mathbf{V}_{j,n(j)}$ is obtained by decomposing $\mathbf{U}$ by means of a particular sequence of convolution filters $(h_{\ell,m_\ell})_{\ell=1,2,\cdots,j}$ where each m_j belongs to $\{0, 1, \ldots, M_j - 1\}$. Therefore, the position parameter in layer j is

$$n(j) = m_j + \sum_{\ell=1}^{j-1} m_\ell \prod_{k=1}^{j-\ell} M_k \triangleq \sum_{\ell=1}^{j} m_\ell \prod_{k=1}^{j-\ell} M_k \tag{15}$$

satisfying $0 \leqslant n(j) \leqslant M_1 M_2, \cdots M_k - 1$.

Thus, path $\mathcal{P}$ can be assigned to the ordered sequence $(m_\ell)_{\ell \geqslant 1}$ where any $m_\ell \in \{0, 1, \ldots, M_\ell - 1\}$. This sequence characterizes the unique set of active nodes leading to a specific node $\mathbf{V}_{j,n(j)}$. An alternative to equation 15 is the recursive equation, which requires the convention $n(0) = 0$: $n(j) = M_j n(j-1) + m_j$.

From equation 14, the fusion stage involved in standard CNNs reduces considerably their graph structuring (see Figure 4) and all downhill nodes contribute to a given uphill node: providing the pair (j, m_j) of layer index and convolution filter position in that layer suffices for the retrieval of the aggregated nodes leading to (j, m_j). Alternative descriptions corresponding to group invariant scattering transforms can be found in Mallat (2012).

5 DISCUSSION AND PROSPECTS

Addressing further mathematical developments of CNNs by using standard conventions through equation 14 is probably less traceable (due to fusion of operators) than when using the fully multiserial expansion of equation 10. We expect that with the increase of storage and computing capabilities, CNN developers will prefer the expanded (and more explicable) CNN of equation 10. Moreover, the latter can lead to generalized hyperserial convolutions in the sense described by Figure 5. In such a hyperserial CNN, the series of filters selected depend no more on the layer: they depend on any node under consideration in the layer. Thus, a triple indexed sequence of convolution filters $\{h_{j,n,0}, h_{j,n,1}, \ldots, h_{j,n,M_{j,n}-1}\}$ will be attached to node (j, n) of layer j for the hyperserial CNNs.

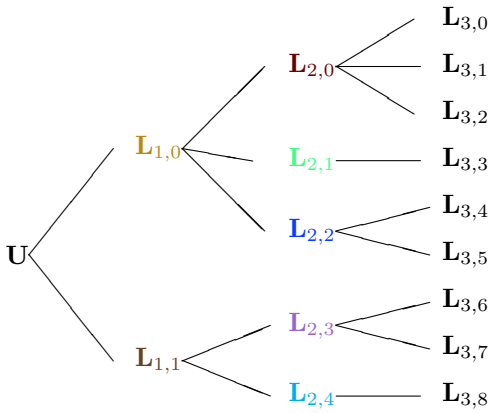

Figure 5: Hyperserial CNN structure where the node $\mathbf{L}_{j,n}$ of layer j involves $M_{j,n}$ convolution filters and the sequence $(M_{j,n})_n$ is not necessarily constant. Compare this graph with the multiserial one given by Figure 3: on the latter, every node of layer j has the same number M_j of convolution filters (it is multiserial because $M_j \neq M_\ell$ in general from layer to layer). The multiserial graph is described by 2 parameters whereas the hyperserial one requires 3 parameters.

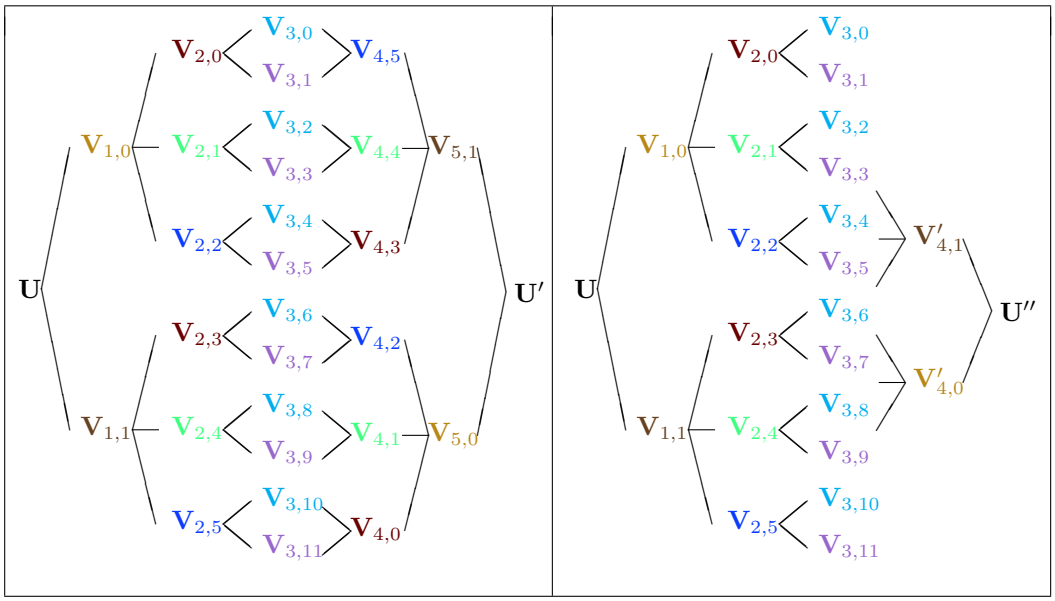

Figure 6: Graphs illustrating several possible processing: (i) analysis at nodes $\{\mathbf{V}_{3,n} : n = 0, 1, \ldots, 11\}$ of hidden layer 3 and reconstruction at output node $\mathbf{U}'$; (ii) joint analysis from a late fusion stage at nodes $\{\mathbf{V}_{4,n} : n = 0, 1, \ldots, 5\}$ of hidden layer 4 (in contrast with early fusion stage for standard CNNs) (iii) selective analysis at nodes $\{\mathbf{V}_{4,n} : n = 0, 1\}$ and instance reconstruction at output node $\mathbf{U}''$.

Among the prospective developments that can be based on the analytical expressions provided by the paper, one can cite sparsity, stationarization, decorrelation or central-limit like properties as in Atto & Pastor (2010) when considering monoserial M-DWPT. Their adaptation to monoserial DWPT requires only the use of theorems providing transforms properties of random variables once the analytical form of the non-linear function Υ is selected. For proving such properties for CNNs, the main difficulties are related to the absence of mathematical frame constraints such as orthogonality and tight frame properties. But once the training of a CNN has ended, then the analysis can be performed by using the learned convolution filters. Several other open issues can be reported here: for instance,

- exploring the best decomposition scheme for expanded CNNs depending on the application, see for instance two possible configurations illustrated by Figure 6;
- using a more informative fusion operator than the sum for standard CNNs in equation 14;
- integrating a constraint per layer such as equation 3 that can help obtaining more generalization properties for very deep CNNs;
- defining constraint programming models for selecting the set of filters to be applied to a given layer with respect to node footprint distance or mutual information criteria, *etc*.

The answers require a huge amount of theories, developments and applications to a wide range of datasets: this makes CNNs a captivating, mysterious and puzzling domain, at large.

When focusing on statistics of machine learning, the proposed unification framework can help in dtermining generalization bounds. This requires however fixing the non-linear activation forms or imposing some structural constraints as in Lee & Raginsky (2019) and Long & Sedghi (2020).

Another statistical prospect concerns building invariants (translation, rotation, scaling, shearing, perspective, *etc*, see Sifre & Mallat (2013) for instance) to complex group of transformations that are necessary for an unambiguous image content description. Expanded CNNs of Section 3.1 removes the recombination along the channel axis. In this respect, seeking different forms of invariance will require specific series of operators. In a deep learning framework, it is reasonable to expect that using different forms of non-linear activation functions can helps in obtaining the required invariances. This is why expansions presented in the paper does not all consider a same activation function Υ.

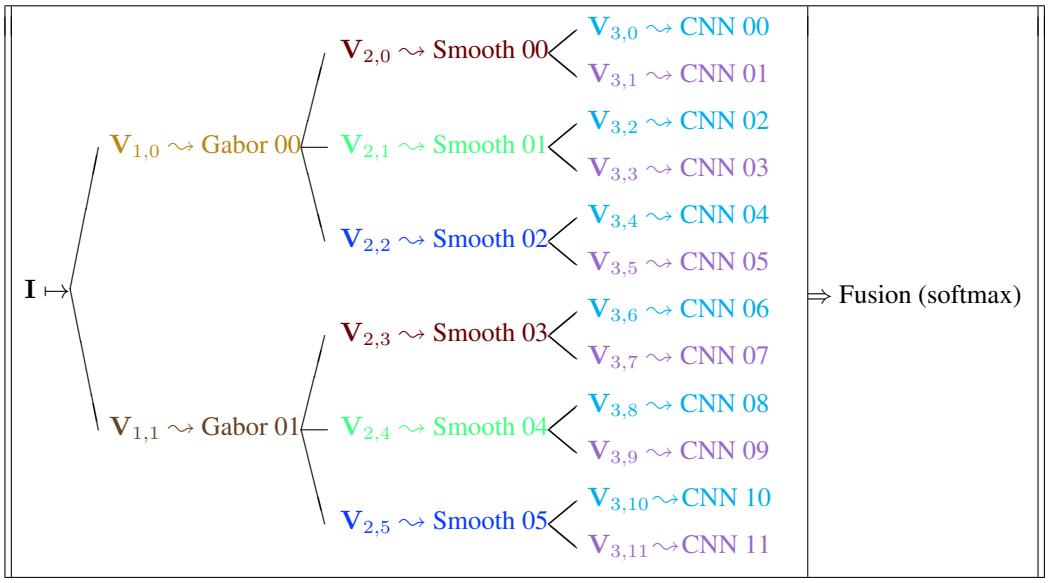

Figure 7: Hybrid representation space. Gabor 00 and 01: Gabor magnitude coefficients corresponding to two different wavelengths. Smooth 00 to 05: post-processing associated with different scaling filters (for instance Gaussian filters with different scaling parameters). CNNs 00 to 11: CNN architectures, where even numbers are associated with Xception CNN and odd numbers correspond to DenseNet CNN. The fusion stage is associated with a geometric mean on the softmax probabilities computed by the different CNNs.

An illustrative example of a hybrid framework integrating standard filter banks and CNNs is given in Figure 7. From this architecture and when using transfer learning from Xception Chollet (2017) and DenseNet Huang et al. (2017) CNNs (cloned 6 times each, $\mathbf{V}_{3,2k}$ for Xception and $\mathbf{V}_{3,2k+1}$ for DenseNet when $k = 0, 1, \ldots, 5$), we have been able to obtain up to 4% of performance gain in comparison with a transfer learning[6] by using one single Xception or DenseNet instance (see Table 1), when the issue is the classification of the Describable Textures Dataset (DTD[7]). This means that using different Gabor filters (thus different frequency selectivities) make Xception more consistent in terms of learning texture features. At this stage, the Gabor phase has not been integrated in the framework of Figure 7 because learning is inhibited by complex mechanisms associated with the phase. This is experimental observation is probably due to the classification constraints: we are not seeking for the best representation of the input image, but the most relevant class separators.

Table 1: Performance of the hybrid framework presented in Figure 7 for a standard classification issue of DTD. Comparison is provided with respect to transfer learning frameworks associated with a single CNN being either Xception or DenseNet.

Standard CNN		**Hybrib filter bank & CNN methods**					
Xception	DenseNet						Xception
		CNN 00	CNN 02	CNN 04	CNN 06	CNN 08	CNN 10
		72	72	69	73	71	70
							DenseNet
		CNN 01	CNN 03	CNN 05	CNN 07	CNN 09	CNN 11
		71	69	67	70	67	66
		Fusion of CNNs 00-to-11 categorical probabilities + decision					
72	69						76

[6]The transfer involves learning only the optimal fully connected layer preceding the output layer.

[7]DTD Cimpoi et al. (2015): collection of 5640 wild textural images associated perceptual description characteristic, including 47 texture categories and divided in 10 experimental splits.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pp. 265–283, Berkeley, CA, USA, 2016. USENIX Association. ISBN 978-1-931971-33-1. URL http://dl.acm.org/citation.cfm?id=3026877.3026899.

Aviad. Aberdam, Jeremias. Sulam, and Michael. Elad. Multi-layer sparse coding: The holistic way. *SIAM Journal on Mathematics of Data Science*, 1(1):46–77, 2019. doi: 10.1137/18M1183352. URL https://doi.org/10.1137/18M1183352.

A. M. Atto and D. Pastor. Central limit theorems for wavelet packet decompositions of stationary random processes. *IEEE Transactions on Signal Processing*, 58(2):896 – 901, Feb. 2010.

A. M. Atto, R. R. Bisset, and E. Trouvé. Frames learned by prime convolution layers in a deep learning framework. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. URL https://doi.org/10.1109/TNNLS.2020.3009059.

F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pp. 675–678, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3063-3. doi: 10.1145/2647868.2654889. URL http://doi.acm.org/10.1145/2647868.2654889.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2668–2677, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

Jaeho Lee and Maxim Raginsky. Learning finite-dimensional coding schemes with nonlinear reconstruction maps. *SIAM Journal on Mathematics of Data Science*, 1(3):617–642, 2019. doi: 10.1137/18M1234461. URL https://doi.org/10.1137/18M1234461.

Philip M. Long and Hanie Sedghi. Generalization bounds for deep convolutional neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1e_FpNFDr.

Stéphane Mallat. Group invariant scattering, 2012.

Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016. doi: 10.1098/rsta.2015.0203. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2015.0203.

Ambar Pal, Connor Lane, René Vidal, and Benjamin D. Haeffele. On the regularization properties of structured dropout, 2019.

G. T. Papadopoulos, E. Machairidou, and P. Daras. Deep cross-layer activation features for visual recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 923–927, Sep. 2016. doi: 10.1109/ICIP.2016.7532492.

L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1233–1240, 2013. doi: 10.1109/CVPR.2013.163.

Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for texture classification, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

P. Steffen, P. N. Heller, R. A. Gopinath, and C. S. Burrus. Theory of regular m-band wavelet bases. *IEEE Transactions on Signal Processing*, 41(12):3497 – 3511, Dec. 1993.

C. Szegedy, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015. doi: 10.1109/CVPR.2015.7298594.

S. N. Tran and A. S. d'Avila Garcez. Deep logic networks: Inserting and extracting knowledge from deep belief networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(2): 246–258, Feb 2018. ISSN 2162-237X. doi: 10.1109/TNNLS.2016.2603784.