**Appears in** *arXiv:1905.01289 and NeurIPS 2019 workshop on Graph Representation Learning*

# Convolution, attention and structure embedding

Jean-Marc Andreoli*
NAVER LABS Europe, Grenoble, France
http://www.europe.naverlabs.com

April 2019

**Abstract**

Deep neural networks are composed of layers of parametrised linear operations intertwined with non linear activations. In basic models, such as the multi-layer perceptron, a linear layer operates on a simple input vector embedding of the instance being processed, and produces an output vector embedding by straight multiplication by a matrix parameter. In more complex models, the input and output are structured and their embeddings are higher order tensors. The parameter of each linear operation must then be controlled so as not to explode with the complexity of the structures involved. This is essentially the role of convolution models, which exist in many flavours dependent on the type of structure they deal with (grids, networks, time series etc.). We present here a unified framework which aims at capturing the essence of these diverse models, allowing a systematic analysis of their properties and their mutual enrichment. We also show that attention models naturally fit in the same framework: attention is convolution in which the structure itself is adaptive, and learnt, instead of being given a priori.

# 1 A generic framework for convolution on arbitrary structures

Convolution is a powerful operator, which is widely used in deep neural networks in many different flavours: [12, 11, 8, 6, 9, 17, 14]. It allows to express in a compact form operations on a structured bundle of similarly shaped data instances (embeddings of nodes in a network, of instants in a time series, of pixels in an image, etc.) taking into account some known structural dependencies between them (edges between nodes, or temporal relations between instants, or positional relations between pixels). In spite of their apparent diversity, these structures can be formalised as *families* of weighted graphs, where each graph in a family captures one aspect of the structure. We develop a generic model of convolution over such structures.

## 1.1 Some useful properties of tensors

A tensor is characterised by its shape $S{=}\langle S_1 \cdots S_{|S|}\rangle$, which is a sequence of integers, its index set which is the cartesian product $\bar{S} \triangleq \prod_{i=1:|S|}\{1 \cdots S_i\}$ of cardinality $|\bar{S}|{=}\prod_{i=1:|S|} S_i$, and its value which is a mapping from its index set into the set of scalars. By construction, the space of tensors of a given shape $S$ is of dimension $|\bar{S}|$. If $S$ and $T$ are shapes, we let $ST$ denote their concatenation. The following common operations on tensors are recalled here (the notation $a{:}S$ stands for "tensor $a$ of shape $S$"):

| | operands | result | definition |
|---|---|---|---|
| *slicing* | $a : ST \quad s \in \bar{S}$ | $a_s : T$ | $(a_s)_t \triangleq a_{st}$ |
| *flattening* | $a : ST \quad \omega : \bar{S} \mapsto \{1 \cdots K\}$ bijective, hence $K = \lvert\bar{S}\rvert$ | $a^{[\omega]} : \langle K\rangle T$ | $a^{[\omega]}_{\langle k\rangle t} \triangleq a_{(\omega^{-1}k)t}$ |
| *outer product* | $a : S \quad b : T$ | $a \otimes b : ST$ | $(a \otimes b)_{st} \triangleq a_s b_t$ |

In the case of flattening, when $T$ is of length 1 (resp. 0), then $\boldsymbol{a}^{[\omega]}$ is a matrix (resp. a vector) and flattening is then called *matricisation* (resp. *vectorisation*) [15]. A common choice for $\omega$ is the *canonical bijection* [15] $\omega_S$ defined for each $s{\in}\bar{S}$ by

$$\omega_S(s) \ \triangleq\ 1 + \sum_{i=1:|S|}(s_i - 1)\prod_{j=i+1:|S|} S_j$$

* jean-marc.andreoli@naverlabs.com

In this paper, we also make use of a less common operation on tensors, called here the *mixed* product: if $K$ is an integer and $\boldsymbol{a}, \boldsymbol{b}$ are tensors of shape $\langle K\rangle S$ and $\langle K\rangle T$, respectively, their mixed product denoted $\boldsymbol{a} \circ \boldsymbol{b}$ is a tensor of shape $ST$ defined by

$$\boldsymbol{a} \circ \boldsymbol{b} \triangleq \sum\nolimits_k \boldsymbol{a}_k \otimes \boldsymbol{b}_k$$

Operator $\circ$ combines features of both the inner and outer products. When $S$ and $T$ are both of length 1, then $\langle K\rangle S$, $\langle K\rangle T$, $ST$ are all of length 2, so $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{a} \circ \boldsymbol{b}$ are all matrices and $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{a}^\top \boldsymbol{b}$.

**Proposition 1** (Decomposition)**.** *Let $S, T$ be arbitrary shapes and $K$ an integer. Let $\boldsymbol{a}$ be a tensor of shape $\langle K\rangle S$ such that the family $(\boldsymbol{a}_k)_{k=1:K}$ be a basis of the space of tensors of shape $S$ (hence $K{=}|\bar{S}|$). Then for any tensor $\boldsymbol{\Phi}$ of shape $ST$ there exists a unique tensor $\boldsymbol{\Theta}$ of shape $\langle K\rangle T$ such that $\boldsymbol{\Phi} = \boldsymbol{a} \circ \boldsymbol{\Theta}$.*

*Proof.* Observe that $\boldsymbol{\Theta} \mapsto \boldsymbol{a} \circ \boldsymbol{\Theta}$ is a linear mapping from the space of tensors of shape $\langle K\rangle T$ into the space of tensors of shape $ST$. The assumption ($\boldsymbol{a}$ is a basis) implies that it is injective, and since the two spaces have the same dimension, the mapping is an isomorphism. □

The expression $\sum_k \boldsymbol{a}_k \otimes \boldsymbol{\Theta}_k$ is formally similar to a linear combination of the basis tensors $(\boldsymbol{a}_k)_{k=1:K}$, except that the coefficients $\boldsymbol{\Theta}_k$ are tensors and scalar multiplication is replaced by tensor product. When $S$ and $T$ are both of length 1, then $\boldsymbol{\Phi}$ is a matrix, and the decomposition simply becomes $\boldsymbol{\Phi}{=}\boldsymbol{a}^\top \boldsymbol{\Theta}$, which is uniquely realised by $\boldsymbol{\Theta}{=}\boldsymbol{a}^{-1\top} \boldsymbol{\Phi}$ ($\boldsymbol{a}$ is invertible by assumption). Proposition 1 is therefore a "tensorised" form of matrix inversion.

## 1.2 A generic convolution model

In a convolution layer, the input does not consist of a simple embedding vector, as in a standard linear layer. Instead, it is a matrix $\boldsymbol{x}$ of shape $\langle M, P\rangle$, representing a bundle of $M$ entries encoded as vectors of shape $\langle P\rangle$. Similarly, the output $\boldsymbol{y}$ is a matrix of shape $\langle N, Q\rangle$ ($N$ entries encoded with shape $\langle Q\rangle$). For example, in image convolutions, $M, P$ are the number of pixels and channels, respectively, of the input image, while $N, Q$ are those of the output image. More generally $\boldsymbol{x}$ and $\boldsymbol{y}$ could be tensors — e.g. images are ternary tensors — but a tensor can always be flattened into a matrix, or even a vector (see Section 1.1). Matricisation, rather than full vectorisation, is used here in order to keep separate the uncontrolled, structural dimensions (width and height in images, of size $M$ in input and $N$ in output) from the controlled ones (channels, of size $P$ in input and $Q$ in output). By analogy with a simple linear layer, the most general form of a convolution layer is an arbitrary linear transform, given by

$$\boldsymbol{y}_{nq} = \sum\nolimits_{mp} \boldsymbol{x}_{mp} \boldsymbol{\Phi}_{mnpq} \tag{1}$$

Tensor $\boldsymbol{\Phi}$, of shape $\langle M, N, P, Q\rangle$, induces (linear) dependencies between each component of each input entry in $\boldsymbol{x}$ and each component of each output entry in $\boldsymbol{y}$. Using an arbitrary $\boldsymbol{\Phi}$ directly as parameter of the convolution is not satisfactory. First, its shape depends on the numbers $M, N$ of input and output entries: $M, N$ may vary for different instances of the data, or may be too large to be involved in the size of a parameter[1]. Furthermore, in Equation (1), the structural dependencies between the $M$ input and $N$ output entries are not captured.
We propose to capture this structure as a tensor $\boldsymbol{A}$ of shape $\langle K, M, N\rangle$, for some integer $K$, and to constrain $\boldsymbol{\Phi}$ to be of the form:

$$\boldsymbol{\Phi} = \boldsymbol{A} \circ \boldsymbol{\Theta} \quad \left(= \sum\nolimits_k \boldsymbol{A}_k \otimes \boldsymbol{\Theta}_k\right) \tag{2}$$

where $\boldsymbol{\Theta}$ is a tensor of shape $\langle K, P, Q\rangle$. Integer $K$ is assumed to be a hyper-parameter controlled by the model, so $\boldsymbol{\Theta}$ has a fully controlled shape and is chosen as parameter of the convolution. Tensor $\boldsymbol{A}$ on the other hand characterises the structure underlying the convolution, and can be viewed as a family $(\boldsymbol{A}_k)_{k=1:K}$ of matrices (weighted graphs between input and output entries). The variety of existing convolution mechanisms derives from various choices for $K$ and $\boldsymbol{A}$ (called resp. the *size* and *basis* of the convolution), which obey different intuitions in different domains. Examples are given below. But in general, combining Equations (1) and (2) together, we obtain a formula for convolution over arbitrary structures:

**Proposition 2.** *A* convolution *of basis $\boldsymbol{A}$, a tensor of shape $\langle K, M, N\rangle$, and parameter $\boldsymbol{\Theta}$, a tensor of shape $\langle K, P, Q\rangle$, is a linear transform which maps a bundle of inputs $\boldsymbol{x}$ represented as a matrix of shape $\langle M, P\rangle$, into a bundle of outputs $\boldsymbol{y}$ represented as a matrix of shape $\langle N, Q\rangle$, according to the rule*

$$\boldsymbol{y} = \sum\nolimits_k \boldsymbol{A}_k^\top \boldsymbol{x} \boldsymbol{\Theta}_k \tag{3}$$

[1]The dependence on $P, Q$, on the other hand, is not problematic, since these are hyper-parameters controlled by the model (embedding sizes).

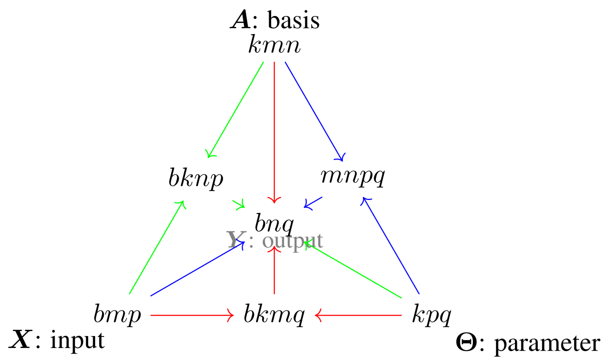


Figure 1: A representation of three alternatives (red-green-blue, each starting from one side of the triangle) to compute a convolution $\boldsymbol{Y}$ (order-3 tensor at the centre). The vertices of the triangle are the order-3 tensors involved ($\boldsymbol{X}$: input, $\boldsymbol{\Theta}$: parameter, $\boldsymbol{A}$: basis) with their respective indices ($b$: batch index, $m/n$: input/output entry, $p/q$: input/output channel, $k$: basis index). The arrows represent sum-product operations in the so called Einstein's notation. For example, the two arrows $bmp, kpq \rightarrow bkmq$ (bottom) represent an operation yielding the order-4 tensor $R_{bkmq} = \sum_p \boldsymbol{X}_{bmp}\boldsymbol{\Theta}_{kpq}$.

Note that our model of structural dependencies is flexible. If $(\boldsymbol{A}_k)_{k=1:K}$ is taken to be a basis of the whole space of matrices of shape $\langle M, N\rangle$, then by Proposition 1 any $\boldsymbol{\Phi}$ can be written as $\boldsymbol{A} \circ \boldsymbol{\Theta}$, and the resulting class of convolutions is the class of arbitrary linear transforms. But of course, this assumes $K{=}MN$, which is uncontrolled. At the other end of the spectrum, if $K{=}1$ and $\boldsymbol{A}_1$ is the identity matrix, the input entries are processed identically and fully independently, leading to a degenerate class of convolutions also known in the image domain as $1{\times}1$ convolutions. In fact, Equation (2) can be viewed as a truncated version of the factorisation of $\boldsymbol{\Phi}$ defined by Proposition 1 where family $(\boldsymbol{A}_k)_{k=1:K}$ is seen as a subset of a basis (of the whole space of matrices of shape $\langle M, N\rangle$), of which the other members are ignored. $(\boldsymbol{A}_k)_{k=1:K}$ act as "principal components".

**Proposition 3.** *Given two convolutions of size $K', K''$, basis $\boldsymbol{A}', \boldsymbol{A}''$, parameter $\boldsymbol{\Theta}', \boldsymbol{\Theta}''$, respectively, their composition, when the dimensions match (i.e. $\langle N', Q'\rangle{=}\langle M'', P''\rangle$), is a convolution of size $K$, basis $\boldsymbol{A}$, parameter $\boldsymbol{\Theta}$ where*

$$K = K'K'' \qquad \boldsymbol{A}_{\omega(k',k'')} = \boldsymbol{A}'_{k'}\boldsymbol{A}''_{k''} \qquad \boldsymbol{\Theta}_{\omega(k',k'')} = \boldsymbol{\Theta}'_{k'}\boldsymbol{\Theta}''_{k''}$$

*and $\omega$ is a bijective mapping $\{1\cdots K'\}{\times}\{1\cdots K''\}{\mapsto}\{1\cdots K\}$, e.g. the canonical bijection $\omega_{\langle K',K''\rangle}$.*

*Proof.* Simple application of Equation (3). □

### 1.3 Separable convolutions

Parameter $\boldsymbol{\Theta}$, of shape $\langle K, P, Q\rangle$, although controlled, may still be too large and it may be useful to constrain it further, e.g. by imposing it to be of the factorised form

$$\boldsymbol{\Theta} \;=\; \boldsymbol{\Theta}^{(\text{basis})} \circ \boldsymbol{\Theta}^{(\text{channel})} \tag{4}$$

where $\boldsymbol{\Theta}^{(\text{basis})}$ is a matrix of shape $\langle H, K\rangle$ (for some integer $H$) and $\boldsymbol{\Theta}^{(\text{channel})}$ a tensor of shape $\langle H, P, Q\rangle$. The resulting convolutions are said to be *separable*. This generalises the so called depth-wise separable convolutions, common in the image domain [1], which are the special case $H{=}1$. The parameter size of a separable convolution is $H(K{+}PQ)$ instead of $KPQ$ for an arbitrary one. An alternative form of dimension reduction is discussed in Section 3.3.

### 1.4 A note on the computation of convolutions

In practice, the input and output entries are usually batched. Batched input $\boldsymbol{X}$ and output $\boldsymbol{Y}$ are given by tensors of shape $\langle B, M, P\rangle$ and $\langle B, N, Q\rangle$, respectively, where $B$ is the batch size. Figure 1 shows the three alternatives to compute $\boldsymbol{Y}$ (at the centre of the triangle) as a function of $\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{\Theta}$ (on the vertices of the triangle), according to the convolution formula of Equation (3) extended to batches:

$$\boldsymbol{Y}_b = \sum\nolimits_k \boldsymbol{A}_k^\top \boldsymbol{X}_b \boldsymbol{\Theta}_k$$

Which of these alternatives should be used essentially depends on the respective dimensions $M, P, N, Q, B, K$. In any case, operations involving the basis tensor $\boldsymbol{A}$ may require a specific treatment, since it is usually very sparse, and sometimes possesses a regularity which can be exploited for optimal computation, as in the case of the "shift matrices" of grid convolution (Figure 2).

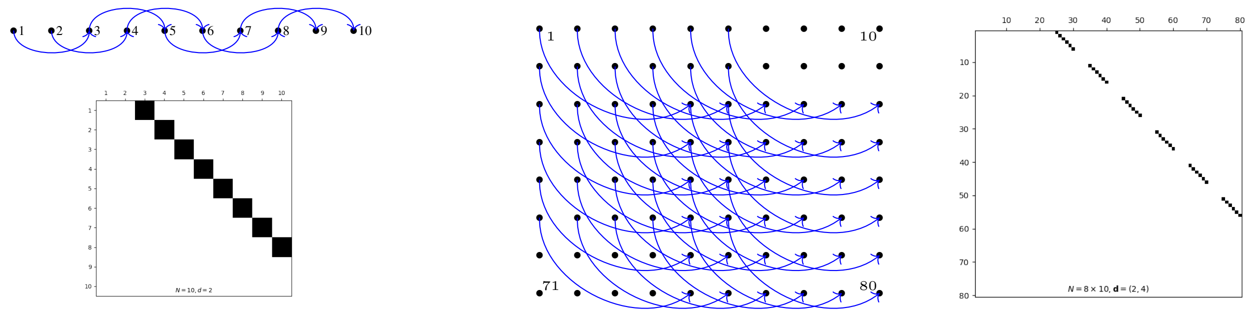


Figure 2: Shift matrices constitute the basis of grid convolutions. Left: shift by 2 in a 1-D grid of dimension 10; Right: shift by $(2, 4)$ in a 2-D grid of dimensions $8{\times}10$ (flattened by the canonical mapping into indices 1:80).

# 2 Some examples

## 2.1 Grid convolutions

A grid is the index set $\bar{S}$ associated with a given sequence of integers $S$. In the case of images, the archetypal grids, $S$ is the sequence $\langle \text{width}, \text{height} \rangle$ of length $|S|{=}2$. We assume given some bijective mapping $\omega{:}\bar{S}{\mapsto}\{1 \cdots N\}$ where $N{=}|\bar{S}|$, for example the canonical mapping $\omega_S$ (see Section 1.1). In this way, an embedding of the whole grid, which would naturally be represented by a tensor of shape $S\langle L \rangle$ where each node in the grid is encoded as a vector of shape $\langle L \rangle$, can be matricised (see Section 1.1) into a matrix of shape $\langle N, L \rangle$ as used in our model. Let's first consider convolutions which preserve the grid, hence $M{=}N$.

**Definition 1.** *For each integer valued vector $\boldsymbol{d}{\in}\mathbb{Z}^{|S|}$, we define the* shift matrix $\mathcal{A}_{\boldsymbol{d}}$ *of shape $\langle N, N \rangle$ by*

$$(\mathcal{A}_{\boldsymbol{d}})_{mn} \triangleq \mathbb{I}[\omega^{-1}n - \omega^{-1}m = \boldsymbol{d}]$$

*A* grid convolution *of size $K$ and basis $\boldsymbol{A}$ is one such that for each $k{\in}1{:}K$, $\boldsymbol{A}_k{=}\mathcal{A}_{\boldsymbol{\Delta}_k}$ for some $\boldsymbol{\Delta}_k{\in}\mathbb{Z}^{|S|}$.*

Thus, $\mathcal{A}_{\boldsymbol{d}}$ is the adjacency matrix of the relation: "node $n$ is obtained from node $m$ by a shift of $\boldsymbol{d}$ in the grid". It is illustrated in Figure 2 in the case of grids of order $|S|{=}1, 2$ (typically, sentences and images). With some padding conventions, Equation (3) for a grid convolution becomes, for any node $s{\in}\bar{S}$ in the grid,

$$\boldsymbol{y}_{(\omega(s))} = \sum\nolimits_k \boldsymbol{x}_{(\omega(s-\boldsymbol{\Delta}_k))} \boldsymbol{\Theta}_k$$

The traditional grid (image) convolutions of "Convolutional Neural Networks" (CNNs) [12] are exactly obtained by choosing $\boldsymbol{\Delta}$ to be a regular right cuboid with possibly different strides $\boldsymbol{\delta}_i$ and offsets $\boldsymbol{\epsilon}_i$ in the different grid dimensions $i{=}1{:}|S|$. In that case, we have $K = \prod_{i=1:|S|} K_i$ and for each $k{\in}1{:}K$

$$\boldsymbol{\Delta}_{ki} = \boldsymbol{\epsilon}_i + (\omega^{-1}_{\langle K_1 \cdots K_{|S|} \rangle} k)_i \boldsymbol{\delta}_i$$

Parameter $\boldsymbol{\Theta}$, of shape $\langle K, P, Q \rangle$, then appears as the flattened version of a tensor of shape $\langle K_1, \cdots, K_{|S|}, P, Q \rangle$, which is the familiar shape of grid convolution kernels.
Variants of grid convolutions which do not necessarily preserve the grid can also be captured in our framework using different choices of $\boldsymbol{\Delta}$, and variants of the shift matrices. This includes average pooling and dilated convolutions, where the output grid is a sub-sample of the input one ($N$ is a divisor of $M$ rather than $M{=}N$). However, our framework covers only convolutions which are linear transforms, which rules out such things as max pooling.

## 2.2 Graph convolutions

Let $\mathcal{G}$ be a graph over $\{1 \cdots N\}$ given a priori. We assume $M{=}N$ (graph convolutions usually preserve the graph).

**Definition 2.** *A* graph convolution *of size $K$ and basis $\boldsymbol{A}$ is one such that for each $k{\in}1{:}K$, matrix $\boldsymbol{A}_k$ is constructed from $\mathcal{G}$ by some procedure dependent on $k$.*

The traditional "Graph Convolution Networks" (GCNs) [9] are exactly obtained by choosing $K{=}1$ and $\boldsymbol{A}_1$ to be a normalised form of the adjacency matrix defined by $\mathcal{G}$. Constraining the size to 1 yields a very simple, efficient architecture, at the price of some expressiveness. For example, although grids can be represented as graphs, grid convolutions cannot be expressed as graph convolutions with a size restricted to 1.

In alternative definitions of graph convolution, the size is possibly greater than 1, and each $\boldsymbol{A}_k$ is computed from $\mathcal{G}$ in a different way. For example, in the full spectral analysis of graph convolution [4], each $\boldsymbol{A}_k$ is a Chebyshev polynomial of the scaled Laplacian matrix of $\mathcal{G}$, up to order $K$. In a simpler version [13], Chebyshev polynomials are replaced by elementary monomials, and $\boldsymbol{A}_k$ is simply the adjacency matrix of $\mathcal{G}$ raised to the power of $k$, capturing the random walks of length $k$ through the graph. A similar approach can be applied to knowledge graphs [16], by introducing one basis matrix $\boldsymbol{A}_k$ for each random walk sort (instead of length) from a given set of sorts (instead of up to a given length), where a sort is a sequence of relations. For example, in a film knowledge base, a sort could be "`played.characterIn.genre`", and a typical instance of random walk of that sort could be "`LeonardNimoy-Spock-StarTreck-SciFi`".

# 3 Attention as content-based convolution

## 3.1 Content-based vs index-based convolution

In the previous examples of convolution, the basis tensor captures prior knowledge about the structural relationships between input and output entries through their indices. This is not the only option. Instead of relying solely on indices, the basis tensor of a convolution can also be computed from any content associated with the input and output entries. We propose a generic model to achieve this, and claim that it captures the essence of many attention mechanisms: [18, 7, 19, 2, 3, 10].

**Definition 3.** *An* attention mechanism *is a parametrised mapping which takes as input two matrices, of shape $\langle M, P' \rangle$ and $\langle N, Q' \rangle$, respectively, and returns an output matrix of shape $\langle M, N \rangle$. The input matrices represent $M$ and $N$ entries encoded as vectors of shape $\langle P' \rangle$ and $\langle Q' \rangle$, respectively, and the output matrix represents an influence graph of the former on the latter, based on their encodings.*

Attention mechanisms can be added or multiplied term-wise, or transformed by term-wise, row-wise or column-wise normalisation. Two particularly useful transformations are *masking* and (column-wise or row-wise) *softmax* normalisation, often used in conjunction. Masking is described here in log domain: given a mask as an a priori matrix $\boldsymbol{H}$ of shape $\langle M, N \rangle$ with values in $\{-\infty, 0\}$ (the log of a binary matrix), if $a$ is an attention mechanism, then one can straightforwardly form the mechanism $a{+}\boldsymbol{H}$: it masks (sets to $-\infty$) the output of $a$ wherever $\boldsymbol{H}$ is $-\infty$ leaving the other values unchanged. In particular, if $\boldsymbol{H}$ is sparse, i.e. the density of $-\infty$ is high, then $a{+}\boldsymbol{H}$ is also sparse, i.e. the density of masked values is high, whatever the sparseness of $a$. This is useful when the dimensions $M, N$ are large and the size $MN$ of the output of $a$ becomes unmanageable. Masking allows to limit a priori which entries from the first input can influence entries from the second input. Observe that when softmax normalisation is applied to a masked attention, the masked values become 0, cancelling the influence of the corresponding inputs in the linear domain[2].

**Definition 4.** *An* attention convolution *of size $K$ and basis $\boldsymbol{A}$ is one such that for each $k{=}1{:}K$, $\boldsymbol{A}_k{=}a(\boldsymbol{x}', \boldsymbol{y}'; \Xi_k)$ for some attention mechanism $a$ and some $\Xi_k$ in the parameter space of $a$. The convolution now has three input matrices, the main input $\boldsymbol{x}$ of shape $\langle M, P \rangle$, and two auxiliary inputs $\boldsymbol{x}', \boldsymbol{y}'$ of shape $\langle M, P' \rangle$, $\langle N, Q' \rangle$ respectively, and returns an output matrix $\boldsymbol{y}$ of shape $\langle N, Q \rangle$ according to Equation* (3)*:*

$$\boldsymbol{y} = \sum\nolimits_k a(\boldsymbol{x}', \boldsymbol{y}'; \Xi_k)^\top \boldsymbol{x} \boldsymbol{\Theta}_k \tag{5}$$

*In* cross-attention *(resp.* self-attention*) convolutions, the main input $\boldsymbol{x}$ is also used as the auxiliary input $\boldsymbol{x}'$ (resp. as both $\boldsymbol{x}', \boldsymbol{y}'$), which, to be shape-consistent, requires $P'{=}P$ (resp. $N{=}M$ and $P'{=}Q'{=}P$).*

Graph and grid convolutions described in the previous sections can be seen as degenerate cases of attention convolutions, in which the output of the mechanism does not depend on its input, but solely on its parameter, given a priori (not learnt). As a result, they are independent of their auxiliary input (ignored), and linear in their main input. On the other hand, in the non degenerate case, the mechanism uses both its inputs, and the resulting convolutions may be non linear in either of them. Furthermore, self- or cross-attention convolutions are not even linear in their main input, since it also occurs as input to the mechanism. A commonly used attention mechanism is bi-affine attention:

**Definition 5.** *Let $\xi$ be a scalar, $\boldsymbol{\mu}, \boldsymbol{\nu}$ be vectors of shape, respectively, $\langle P' \rangle, \langle Q' \rangle$, and $\boldsymbol{\Lambda}$ be a matrix of shape $\langle P', Q' \rangle$. The* bi-affine *attention mechanism $\mathcal{A}$ parametrised by $\Xi{=}\langle \xi, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\Lambda} \rangle$ is defined, for matrices $\boldsymbol{x}', \boldsymbol{y}'$ of shape, respectively, $\langle M, P' \rangle$ and $\langle N, Q' \rangle$, by*

$$\mathcal{A}(\boldsymbol{x}', \boldsymbol{y}'; \Xi) \triangleq \boldsymbol{x}' \boldsymbol{\Lambda} \boldsymbol{y}'^T + (\boldsymbol{x}' \boldsymbol{\mu}) \otimes \mathbf{1}_N + \mathbf{1}_M \otimes (\boldsymbol{y}' \boldsymbol{\nu}) + \xi \mathbf{1}_M \otimes \mathbf{1}_N \tag{6}$$

$$\text{equivalently,} \qquad \mathcal{A}(\boldsymbol{x}', \boldsymbol{y}'; \Xi)_{mn} = \sum\nolimits_{p'q'} \boldsymbol{\Lambda}_{p'q'} \boldsymbol{x}'_{mp'} \boldsymbol{y}'_{nq'} + \sum\nolimits_{p'} \boldsymbol{\mu}_{p'} \boldsymbol{x}'_{mp'} + \sum\nolimits_{q'} \boldsymbol{\nu}_{q'} \boldsymbol{y}'_{nq'} + \xi$$

[2]As a general rule, softmax takes input in log domain (scores) and produces output in linear domain (probabilities).

Observe that parameter $\Xi$ has a fully controlled shape, independent of $M, N$. Bi-affine attention is used in the specific context of parsing in [5]. It is also used, with some restrictions on parameter $\Xi$, as a generic attention mechanism in the Transformer model for sequences [18], and in Graph attention networks [19], as shown below.

### 3.2 Attention in Graph Attention Networks

Graph attention networks [19] are based on self-attention convolutions as defined above, except they use a variant of Equation (5): the attention mechanism $a$ is passed the term $\boldsymbol{x\Theta_k}$ instead of $\boldsymbol{x}$ as both auxiliary inputs. In fact, that term is already available before entering the attention mechanism, when computation is started from the bottom of the triangle in Figure 1.
The attention mechanism proposed in [19] starts, in each head $k$, with the bi-affine mechanism of Equation (6) without its bi-linear term, i.e. $\boldsymbol{\Lambda}_k{=}0$. The output is then masked by a graph given a priori, limiting the zone of influence on each node to a neighbourhood of that node. When such prior graph is available, this makes sense, esp. to deal with large structures such as publication networks (up to 50,000 nodes in their experiments, hence, without mask, the output of the mechanism would be of size of magnitude $10^9$). The masked output is then normalised by a term-wise "leaky ReLU" followed by a column-wise softmax.
These choices can be motivated to some extent by properties of the simplified bi-affine mechanism at work. Indeed, observe that the masked values are still masked after the leaky ReLU (which would not be the case with a plain ReLU), and, after the softmax, the masked values become 0, cancelling the influence of the corresponding inputs. Skipping ReLU altogether before the softmax would make the term involving $\boldsymbol{\nu}$ in the right-hand side of Equation (6) redundant: it is constant along each column, and softmax is invariant to an additive constant.

### 3.3 Attention in Transformer

We now show how the attention model described by Equations (5) and (6) encompasses the scaled dot product attention used in the Transformer model of [18]. In Transformer attention convolutions, the auxiliary inputs are called "key" and "query", respectively, while the main input is called "value". Attention is used in three distinct layers of the Transformer architecture. Two of them are instances of self-attention (on the source sequence and on the target sequence, respectively) while the third one is a cross-attention (the main input is the source sequence and the remaining auxiliary input is the target sequence). Masking is used in the target sequence self-attention, to ensure that tokens in that sequence do not have influence on their predecessors. This is because, in Transformer, the ultimate output of all the attention layers is used to model the next token from each position in the target sequence, so should not rely on the availability of that token.
In all cases, the scaled dot product attention used in Transformer essentially relies on the bi-affine attention mechanism of Equation (6), followed by a column-wise softmax. Actually, only the bi-linear part of Equation (6) is kept, i.e. all the parameters except $\boldsymbol{\Lambda}_k$ are null. Furthermore, parameter $\boldsymbol{\Lambda}_k$ is constrained to be of the form

$$\boldsymbol{\Lambda}_k \;=\; \boldsymbol{\Lambda}_k^{(\text{key})}\boldsymbol{\Lambda}_k^{(\text{query})\top} \qquad \left(=\boldsymbol{\Lambda}_k^{(\text{key})\top}\circ\boldsymbol{\Lambda}_k^{(\text{query})\top}\right) \tag{7}$$

where matrices $\boldsymbol{\Lambda}_k^{(\text{key})}, \boldsymbol{\Lambda}_k^{(\text{query})}$ are of shape $\langle P', D\rangle, \langle Q', D\rangle$, respectively. This can be viewed as a simple dimension reduction technique, since only $(P'{+}Q')D$ parameters are required instead of $P'Q'$ for an arbitrary $\boldsymbol{\Lambda}_k$.
Now, Transformer attention introduces a seemingly richer mechanism to combine the different heads. Instead of simply summing them together as in Equation (5), it combines them with yet another linear layer:

$$\boldsymbol{y} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_K]\boldsymbol{\Theta}^{(\text{O})\top} \qquad \text{where} \qquad \boldsymbol{h}_k \triangleq \boldsymbol{A}_k^\top \boldsymbol{x}\boldsymbol{\Theta}_k^{(\text{value})}$$

where $\boldsymbol{\Theta}^{(\text{O})}$ is a matrix of shape $\langle Q, KD\rangle$. In fact, this expression can be rewritten, splitting $\boldsymbol{\Theta}^{(\text{O})}$ into $K$ blocks $(\boldsymbol{\Theta}_k^{(\text{O})})_{k=1:K}$ each of shape $\langle Q, D\rangle$, as

$$\boldsymbol{y} = \sum\nolimits_k \boldsymbol{h}_k\boldsymbol{\Theta}_k^{(\text{O})\top} = \sum\nolimits_k \boldsymbol{A}_k^\top\boldsymbol{x}\boldsymbol{\Theta}_k^{(\text{value})}\boldsymbol{\Theta}_k^{(\text{O})\top}$$

In other words, it is strictly equivalent to the sum model of Equation (5), only with the constraint

$$\boldsymbol{\Theta}_k \;=\; \boldsymbol{\Theta}_k^{(\text{value})}\boldsymbol{\Theta}_k^{(\text{O})\top} \qquad \left(=\boldsymbol{\Theta}_k^{(\text{value})\top}\circ\boldsymbol{\Theta}_k^{(\text{O})\top}\right) \tag{8}$$

This constraint is not even specific to an attention model and could apply to any convolution. In fact, Equations (8) and (7) are meant to reduce the dimensionality of the parameters ($\boldsymbol{\Theta}$, and, in the case of attention, $\boldsymbol{\Lambda}$) by factorisation. Formally, they apply the exact same recipe as applied to $\boldsymbol{\Phi}$ in Equation (2) or $\boldsymbol{\Theta}$ in Equation (4), with the same purpose.
Finally, Transformers take the extreme approach of relying exclusively on content-based convolution ("attention is all you need"), so that any index-based information such as the relative position of the tokens must be incorporated into the content. They propose a smart but not completely intuitive scheme to achieve that, called "positional encoding". Alternatively, one or several additional heads with purely index-based basis matrices (e.g. shift matrices as in grid convolutions) could also be used, in complement to the attention heads.

# References


[1] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *arXiv:1610.02357 [cs]* (Oct. 7, 2016). arXiv: `1610.02357`.

[2] Yagmur Gizem Cinar et al. "Period-aware content attention RNNs for time series forecasting with missing values". In: *Neurocomputing* 312 (Oct. 27, 2018), pp. 177–186.

[3] Yagmur G. Cinar et al. "Position-based Content Attention for Time Series Forecasting with Sequence-to-sequence RNNs". In: *arXiv:1703.10089 [cs]* (Mar. 29, 2017). arXiv: `1703.10089`.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: NIPS. Barcelona, Spain, 2016, p. 9.

[5] Timothy Dozat and Christopher D. Manning. "Deep Biaffine Attention for Neural Dependency Parsing." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

[6] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". In: *arXiv:1603.07285 [cs, stat]* (Mar. 23, 2016). arXiv: `1603.07285`.

[7] Maha Elbayad, Laurent Besacier, and Jakob Verbeek. "Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction". In: *arXiv:1808.03867 [cs]* (Aug. 11, 2018). arXiv: `1808.03867`.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. 785 pp.

[9] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *arXiv:1609.02907 [cs, stat]* (Sept. 9, 2016). arXiv: `1609.02907`.

[10] Wouter Kool, Herke van Hoof, and Max Welling. "Attention Solves Your TSP, Approximately". In: *arXiv:1803.08475 [cs, stat]* (Mar. 22, 2018). arXiv: `1803.08475`.

[11] Y. LeCun, K. Kavukcuoglu, and C. Farabet. "Convolutional networks and applications in vision". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. Proceedings of 2010 IEEE International Symposium on Circuits and Systems. May 2010, pp. 253–256.

[12] Yann LeCun and Yoshua Bengio. "Convolutional networks for images, speech, and time series". In: *The Handbook of Brain Theory and Neural Networks*. Ed. by Michael A. Arbib. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.

[13] Yaguang Li et al. "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting". In: *arXiv:1707.01926 [cs, stat]* (July 6, 2017). arXiv: `1707.01926`.

[14] Stéphane Mallat. "Understanding Deep Convolutional Networks". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 13, 2016), p. 20150203. arXiv: `1601.04920`.

[15] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. "Introduction to Tensor Decompositions and their Applications in Machine Learning". In: *arXiv:1711.10781 [cs, stat]* (Nov. 29, 2017). arXiv: `1711.10781`.

[16] Michael Schlichtkrull et al. "Modeling Relational Data with Graph Convolutional Networks". In: *arXiv:1703.06103 [cs, stat]* (Mar. 17, 2017). arXiv: `1703.06103`.

[17] Xingjian Shi et al. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *arXiv:1506.04214 [cs]* (June 12, 2015). arXiv: `1506.04214`.

[18] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv:1706.03762 [cs]* (June 12, 2017). arXiv: `1706.03762`.

[19] Petar Veličković et al. "Graph Attention Networks". In: *arXiv:1710.10903 [cs, stat]* (Oct. 30, 2017). arXiv: `1710.10903`.