Rapid Exact Signal Scanning with Deep Convolutional Neural Networks

Markus Thom and Franz Gritschneder

***Abstract*—We introduce and analyze a rigorous formulation of the dynamics of a signal processing scheme aimed at exact dense signal scanning. Related methods proposed in the recent past lack a satisfactory analysis whether they actually fulfill any exactness constraints. We improve on this through an exact characterization of the requirements for a sound sliding window approach. The tools developed in this paper are especially beneficial if Convolutional Neural Networks are employed, but can also be used as a more general framework to validate related approaches to signal scanning. The contributed theory helps to eliminate redundant computations and renders special case treatment unnecessary, resulting in a dramatic boost in efficiency particularly on massively parallel processors. This is demonstrated both theoretically in a computational complexity analysis and empirically on modern parallel processors.**

I. Introduction

Even though today's signal processing systems have achieved an unprecedented complexity, a multitude of them has a very simple basic idea in common: The application of a translation-invariant function to a large signal in a sliding fashion facilitates the dense computation of interesting output values for each possible spatial location. Consider filter-based signal denoising as an example: Here, each entry of the denoised output signal always depends on a fixed computation rule applied to only a limited number of samples within the input signal. The computation rule is completely agnostic with regard to the concrete position, it is merely important that the input samples are drawn accordingly from the input signal.

Of course, modern systems are much more sophisticated than just filtering. But one architecture essentially made up of simple filtering building blocks has proven in the recent past to improve on any other approach in a wide variety of practical applications. Due to significant advances in the design of massively parallel processors and the availability of huge annotated data sets, deep artificial neural networks which learn desired behavior by adapting their degrees of freedom to concrete sample data rather than being programmed explicitly have become the de facto state-of-the-art in the domains of signal restoration and signal classification.

The most important architecture for analyzing signals that possess a spatial structure, such as images where pixels are arranged on a two-dimensional grid, was inspired by findings on the dynamics of mammalian visual cortex [1]: *Convolutional Neural Networks (CNNs)* [2], [3] respect the weight sharing principle, hence convolution with trainable filters becomes the actual workhorse for data processing. This principle greatly reduces the network's degrees of freedom making it less susceptible to overfitting, and it incorporates a strong prior with respect to the spatial layout of the input data. In fact, this particular architecture has proven highly successful both for image restoration tasks [4], [5], [6] and pattern recognition problems [7], [8], [9].

If a CNN trained for object categorization is evaluated at each feasible image position, it is possible to assign class membership estimations to all the pixels in an image yielding a semantic segmentation of a scene [10], [11], [12]. This representation is much more powerful than just what can be gained from a conventional object detection approach which solely outputs bounding boxes of found object instances. Instead, it provides the necessary information for a complete understanding of a scene. While the computational complexity of a sophisticated classification system used in conjunction with a sliding window approach may seem excessive at first glance, the weight sharing principle of a CNN can be exploited so that intermediate computation results can be shared among adjacent image patches, resulting in a speedup of several orders of magnitude. Although this was already realized for CNNs without pooling layers more than two decades ago [13], approaches that account for pooling layers as well emerged only recently [14], [15], [16].

The approach of Giusti *et al.* [14] achieves fast scanning of entire images by the introduction of a fragmentation data structure. Here, the internal representations of a CNN are decomposed using a spatial reordering operation after each pooling layer, facilitating the evaluation of convolutions on contiguous signals at all times. The intermediate signals are however inhomogeneous with respect to their dimensionality, leaving the possibility of the usage of efficient tensor convolution routines unclear. Li *et al.* [15], on the other hand, propose to enlarge the filter banks of convolutional layers by inserting vanishing entries at regular locations. These sparse filter banks require a cumbersome re-engineering of efficient convolution implementations, which may not be able to achieve maximum throughput on modern massively parallel processors. Sermanet *et al.* [16] simply use the same processing pipeline for patches and entire images, which incurs relaxations with accuracy loss effects where the actual impact is hard to predict.

All these approaches have in common that it is not inherently clear what they actually compute and if this is the desired result at all. Instead of a rigorous mathematical proof of correctness only toy examples are available, illustrating the implementation of these approaches. This situation is especially unsatisfactory if, instead of pure convenience functions, systems subject to safety considerations should be realized where precise statements on the achieved accuracy

M. Thom and F. Gritschneder are with driveU / Institute of Measurement, Control and Microtechnology, Ulm University, 89081 Ulm, Germany (e-mail: markus.thom@uni-ulm.de, franz.gritschneder@uni-ulm.de).

are required.

This paper improves upon previous work by establishing a comprehensive theory of *subsignal compatible transformations*: these form a family of functions that exactly fulfill the invariants required for a sound sliding window approach. We provide a characterization of such transformations and show that their compositions fulfill the same invariants. Further, we demonstrate how CNNs interconnect with the developed theory, and we show how they can be transformed from a subsignal-based application to a signal-based application using a sliding window approach without any accuracy loss while yielding significant speedups. We provide mathematical statements that rigorously prove the exactness, that is that both subsignal-based and signal-based application lead to the very same results. The developed theory can be used directly for an efficient implementation on massively parallel processing systems, for validation if a given implementation is correct, or as a theoretical framework for analyzing new notions of signal processing based on translation-invariant functions applied in a sliding fashion.

The remainder of this paper is structured as follows. In Sect. II, we give an introduction to the CNN structure, fix the notation and introduce what we mean by subsignals. Section III establishes the basics of our theory on subsignal compatible transformations and shows how the building blocks of CNNs fit into the theory. In the following Sect. IV we extend the theory to functions applied in a strided fashion, which is particularly important for pooling operators evaluated on non-overlapping blocks. Section V provides a theoretical computational complexity analysis. Practical considerations and the results of an experimental evaluation on real parallel processors are discussed in Sect. VI. The paper is concluded with a discussion of our results in Sect. VII.

II. Prerequisites

In this section, we start by introducing the building blocks of a CNN. Next, we fix the notation used throughout the paper. The section is concluded by the formal definition of the subsignal extraction operator and statements on its properties.

A. Convolutional Neural Networks

CNNs are organized in a number of specialized layers [17]. Each layer receives input data from its predecessor, processes it, and sends the result to the next layer. The network's output is then the output of the final layer. The training process consists of tuning the network's degrees of freedom until the network produces the desired output given concrete input sample data [18]. After a network has been trained, it can be used as a predictor on previously unseen data in regression or classification tasks.

The different specialized layer types are given as follows. *Convolutional layers* respect the weight sharing principle: they convolve their input with a trainable filter bank and add a trainable scalar bias to form the layer output. These layers fall into the class of subsignal compatible transformations detailed in Sect. III, a mathematical analysis of the involved computations is given in Sect. III-C.

Fully-connected layers are just a special case of convolutional layers in that they carry out a convolution with unit spatial filter size. Mathematical treatment of these layers is hence superseded by the analysis of convolutional layers.

Non-linearity layers independently send each sample of a signal through a scalar transfer function. This prevents the entire network to form a purely linear system and hence enhances the network's representational capacity. Since these operations are agnostic with respect to any spatial structure, an analysis is straightforward and handled in Sect. III-C.

Eventually, *pooling layers* strengthen a network's invariance to small translations of the input data by evaluation of a fixed pooling kernel followed by a downsampling operation. For brevity of the presentation, we here consider only functions which are applied to non-overlapping blocks. Pooling requires an extension of the plain theory of subsignal compatible transformations, provided in Sect. IV.

B. Notation

For the sake of simplicity, we restrict our mathematical analysis to vector-shaped signals. The generalization of our results to more complex signals like images is straightforward by application of the theory to the two independent spatial dimensions of images. This is briefly discussed in Sect. VI.

We write $\mathbb{N}_1 := \mathbb{N} \setminus \{ 0 \}$ for the positive natural numbers. If M is a set and $q \in \mathbb{N}_1$, then M^q denotes the set of all q-tuples with entries from M. The elements of M^q are called *signals*, their q entries are called *samples*. If $\xi = (\xi_1, \dots, \xi_q) \in M^q$ is a signal and $I \in \{1, \dots, q\}^r$ is an index list with r entries, we use the *formal sum* $\omega := \sum_{\nu=1}^{r} \xi_{I_\nu} \cdot e_\nu^r$ for the element $\omega \in M^r$ with $\omega_\nu = \xi_{I_\nu}$ for all $\nu \in \{1, \dots, r\}$. For example, when M equals the set of real numbers $\mathbb{R}$ and hence M^r is the r-dimensional Euclidean space, then the formal sum ω corresponds to the linear combination of canonical basis vectors e_ν^r weighted with selected coordinates of the signal ξ.

For $\xi \in M^q$ we write $\dim_M(\xi) = q$ for the *dimensionality* of ξ. This does not need to correspond exactly with the concept of dimensionality in the sense of linear algebra. If for example $M = \mathbb{N}^c$ for categorical data with $c \in \mathbb{N}_1$ features, then M^q is not a vector space over M. The theory presented in this paper requires algebraic structures such as vector spaces or analytic structures such as the real numbers only for certain examples. The bulk of our results hold for signals with samples from arbitrary sets.

If M is a set and $c \in \mathbb{N}_1$ is a positive natural number, we write $\cup_c(M) := \cup_{q=c}^{\infty} M^q$ for the set that contains all the signals of dimensionality greater than or equal to c with samples from M. For example, if $\xi \in \cup_c(M)$ then there is a natural number $q \geq c$ so that $\xi = (\xi_1, \dots, \xi_q)$ with $\xi_\nu \in M$ for all $\nu \in \{1, \dots, q\}$. Note that $\cup_1(M)$ contains all non-empty signals with samples from M.

C. Division of a Signal into Subsignals

A *subsignal* is a contiguous list of samples contained in a larger signal. Let us first formalize the concept of extracting subsignals with a fixed number of samples from a given signal:

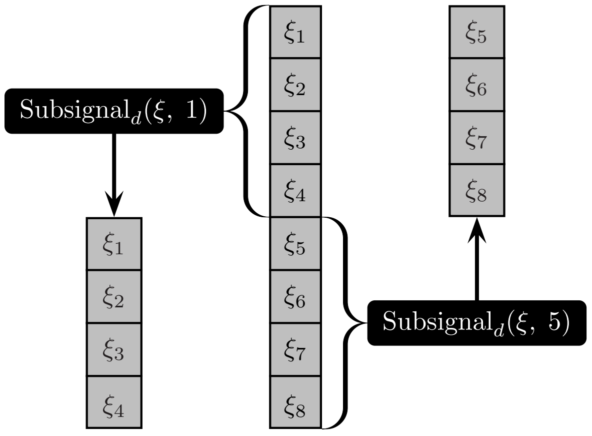

$\mathrm{Subsignal}_d(\xi,\ 1)$
$\mathrm{Subsignal}_d(\xi,\ 5)$
ξ_1 ξ_2 ξ_3 ξ_4 ξ_5 ξ_6 ξ_7 ξ_8

Fig. 1. Illustration of the subsignal extraction operator applied to a signal ξ with $D = 8$ samples for extraction of subsignals with $d = 4$ samples. The left-hand side shows the first subsignal and the right-hand side shows the final subsignal with the maximum subsignal index of $D - d + 1 = 5$.

Definition 1. Let M be a set and let $d \in \mathbb{N}_1$ denote a fixed subsignal dimensionality. Then

$$\mathrm{Subsignal}_d\colon \bigcup_{D=d}^{\infty} \left(M^D \times \{1, \ldots, D-d+1\}\right) \to M^d,$$
$$(\xi,\ i) \mapsto \sum_{\nu=1}^{d} \xi_{i+\nu-1} \cdot e_\nu^d,$$

is called the *subsignal extraction operator*. Here, ξ is the input signal and i denotes the *subsignal index.*

It is straightforward to verify that $\mathrm{Subsignal}_d$ is well-defined and actually returns all possible $D-d+1$ contiguous subsignals of length d from a given signal with D samples (see Fig. 1). Note that for application of this operator it has always to be ensured that the requested subsignal index i is within bounds, that is $i \in \{1, \ldots, D-d+1\}$ must hold to address a valid subsignal.

Iterated extraction of subsignals of different length can be collapsed into one operator evaluation:

Lemma 2. Let M be a set and $c, d \in \mathbb{N}_1$, $c \leq d$, be two subsignal dimensionalities. Then

$$\mathrm{Subsignal}_c(\mathrm{Subsignal}_d(\xi,\ i),\ j) = \mathrm{Subsignal}_c(\xi,\ i+j-1)$$

for all $\xi \in \cup_c(M)$, for all $i \in \{1, \ldots, \dim_M(\xi) - d + 1\}$ and for all $j \in \{1, \ldots, d - c + 1\}$.

Proof. The subsignal indices of the left-hand side are well within bounds. Since $i + j - 1 \in \{1, \ldots, \dim_M(\xi) - c + 1\}$ this also holds for the right-hand side. We find that

$$\begin{aligned}
&\mathrm{Subsignal}_c(\mathrm{Subsignal}_d(\xi,\ i),\ j)\\
\overset{\text{D. 1}}{=}\ & \sum_{\lambda=1}^{c} \mathrm{Subsignal}_d(\xi,\ i)_{j+\lambda-1} \cdot e_\lambda^c\\
\overset{\text{D. 1}}{=}\ & \sum_{\lambda=1}^{c} \left(\sum_{\nu=1}^{d} \xi_{i+\nu-1} \cdot e_\nu^d\right)_{j+\lambda-1} \cdot e_\lambda^c\\
\overset{(\Diamond)}{=}\ & \sum_{\lambda=1}^{c} \xi_{(i+j-1)+\lambda-1} \cdot e_\lambda^c\\
\overset{\text{D. 1}}{=}\ & \mathrm{Subsignal}_c(\xi,\ i+j-1),
\end{aligned}$$

where in the $(\Diamond)$ step we have substituted $\nu = j + \lambda - 1$. □

III. Subsignal Compatible Transformations

This section introduces the concept of subsignal compatible transformations. These are functions that can be applied to an entire signal at once and then yield the same result as if they were applied to each subsignal independently. We show that functions applied in a sliding fashion can be characterized as subsignal compatible transformations, and that the composition of subsignal compatible transformations is again a subsignal compatible transformation.

In the end of this section, we consider CNNs without pooling layers and demonstrate that these satisfy the requirements of subsignal compatible transformations. As a consequence, such networks can be applied to the whole input signal at once without having to handle individual subsignals. CNNs that *do* contain pooling layers require more theoretical preparations and are discussed verbosely in Sect. IV.

We begin with the major definition of this section:

Definition 3. Let M and N be sets, let $c \in \mathbb{N}_1$ be a positive natural number, and let $T\colon \cup_c(M) \to \cup_1(N)$ be a function. We then call T a *subsignal compatible transformation with dimensionality reduction constant* c if and only if these two properties hold:

(i) *Dimensionality reduction property (DRP):*
$\dim_N(T(\xi)) = \dim_M(\xi) - c + 1$ for all $\xi \in \cup_c(M)$.

(ii) *Exchange property (XP):*
For all subsignal dimensionalities $d \in \mathbb{N}_1$, $d \geq c$, it holds that $T(\mathrm{Subsignal}_d(\xi,\ i)) = \mathrm{Subsignal}_{d-c+1}(T(\xi),\ i)$ for all $\xi \in \cup_d(M)$ and all $i \in \{1, \ldots, \dim_M(\xi) - d + 1\}$.

The first property guarantees that T reduces the dimensionality of its argument always by the same amount regardless of the concrete input. The second property states that if T is applied to an individual subsignal, then this is the same as applying T to the entire signal and afterwards extracting the appropriate samples from the resulting signal. Therefore, if with subsignal-based application of T the outcome for *all* feasible subsignals should be determined, it suffices to carry out signal-based application of T on the entire input signal once, preventing redundant computations. Figure 2 illustrates these concepts.

We note that the exchange property is well-defined: The dimensionality reduction property guarantees that the dimensionalities on both sides of the equation match. Further, the subsignal index i is within bounds on both sides. This is trivial for the left-hand side, and can be seen for the right-hand side since $\dim_M(\xi) - d + 1 = (\dim_M(\xi) - c + 1) - (d - c + 1) + 1$.

We immediately have an identity theorem for subsignal compatible transformations:

Theorem 4. Let M, N be sets and $T_1, T_2\colon \cup_c(M) \to \cup_1(N)$ two subsignal compatible transformations with dimensionality reduction constant $c \in \mathbb{N}_1$. If $T_1(\rho) = T_2(\rho)$ holds for all $\rho \in M^c$, then already $T_1 = T_2$.

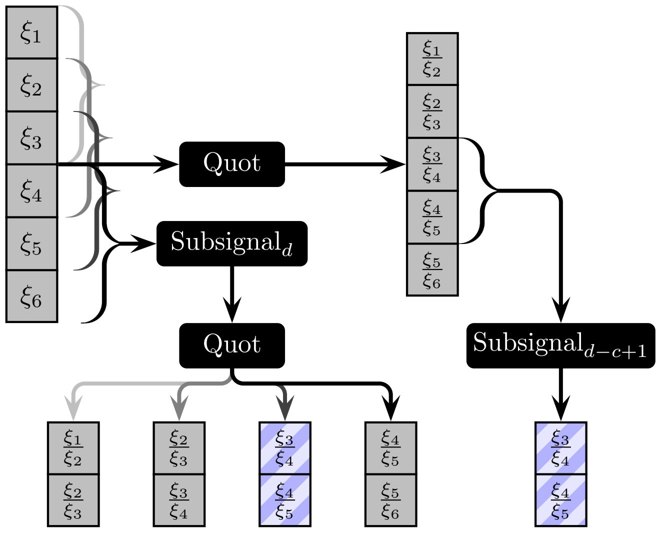


Fig. 2. Example of a subsignal compatible transformation. Here, Quot is a non-linear operator that computes the quotient of $c = 2$ adjacent samples and always reduces the dimensionality of its input by one sample, satisfying the dimensionality reduction property. The lower part shows the result of first extracting subsignals with $d = 3$ samples from the input signal ξ and then evaluating Quot. This yields processed subsignals $\text{Quot}\,(\text{Subsignal}_d(\xi,\ i))$ with $d - c + 1 = 2$ samples each. The exchange property guarantees that these processed subsignals can also be found in $\text{Quot}(\xi)$, exemplarily shown at the right-hand side of the graphics for subsignal index $i = 3$.

Proof. Let $\xi \in \cup_c(M)$. For $\mu \in \{1, \dots, \dim_M(\xi)-c+1\}$ we yield with the precondition (PC) and the exchange property where the subsignal dimensionality d is set to c:

$$\begin{aligned} T_1(\xi)_\mu &\overset{\text{XP}}{=} T_1\,(\text{Subsignal}_c(\xi,\ \mu)) \\ &\overset{\text{PC}}{=} T_2\,(\text{Subsignal}_c(\xi,\ \mu)) \overset{\text{XP}}{=} T_2(\xi)_\mu. \end{aligned}$$

Hence all the samples of the transformed signals are equal, thus $T_1(\xi) = T_2(\xi)$ for all ξ in the domain of T_1 and T_2. □

A. Relationship between Functions Applied in a Sliding Fashion and Subsignal Compatible Transformations

We now investigate functions which are applied to a signal in a *sliding* fashion. Let us first define what is meant hereby:

Definition 5. Let M and N be sets, let $c \in \mathbb{N}_1$ be a positive natural number and let $f\colon M^c \to N$ be a function. Then

$$\text{Slide}_f\colon\ \cup_c\,(M) \to \cup_1(N),$$
$$\xi \mapsto \sum_{i=1}^{\dim_M(\xi)-c+1} f\,(\text{Subsignal}_c(\xi,\ i)) \cdot e_i^{\dim_M(\xi)-c+1},$$

is the operator that applies f in a *sliding fashion* to all the subsignals of length c of the input signal and stores the result in a contiguous signal. The sliding window is here always advanced by exactly one entry after each evaluation of f.

The next result states that functions applied in a sliding fashion are essentially the same as subsignal compatible transformations, and that the exchange property could be weakened to hold only for the case where the dimensionality reduction constant equals the subsignal dimensionality:

Theorem 6. Let M and N be sets, let $c \in \mathbb{N}_1$ and let $T\colon \cup_c\,(M) \to \cup_1(N)$ be a function. Then the following are equivalent:

(a) T is a subsignal compatible transformation with dimensionality reduction constant c.
(b) T fulfills the dimensionality reduction property, and for all $\xi \in \cup_c(M)$ and all $i \in \{1, \dots, \dim_M(\xi) - c + 1\}$ it holds that $T\,(\text{Subsignal}_c(\xi,\ i)) = T(\xi)_i$.
(c) There is a unique function $f\colon M^c \to N$ with $T = \text{Slide}_f$.

Proof. (a) ⇒ (b): Trivial, since the dimensionality reduction property is fulfilled by definition, and the claimed condition is only the special case of the exchange property where $d = c$.

(b) ⇒ (c): For showing existence, define $f\colon M^c \to N$, $\xi \mapsto T(\xi)$. For $\xi \in M^c$ we have $\dim_N(T(\xi)) = 1$ due to the dimensionality reduction property, therefore f is well-defined. Now let $\xi \in \cup_c(M)$ and define $D := \dim_M(\xi)$. It is clear that $\dim_N(T(\xi)) = \dim_N(\text{Slide}_f(\xi)) = D - c + 1$. Now let $i \in \{1, \dots, D - c + 1\}$, then the precondition (PC) implies

$$\begin{aligned} \text{Slide}_f(\xi)_i &\overset{\text{D. 5}}{=} f\,(\text{Subsignal}_c(\xi,\ i)) \\ &= T(\text{Subsignal}_c(\xi,\ i)) \overset{\text{PC}}{=} T(\xi)_i, \end{aligned}$$

hence $T = \text{Slide}_f$.

Considering uniqueness, suppose that there exist functions $f_1, f_2\colon M^c \to N$ with $T = \text{Slide}_{f_1} = \text{Slide}_{f_2}$. Let $\rho \in M^c$ be arbitrary, then with Definition 5 we have $f_1(\rho) = \text{Slide}_{f_1}(\rho) = \text{Slide}_{f_2}(\rho) = f_2(\rho)$, therefore $f_1 = f_2$ on M^c.

(c) ⇒ (a): Let us suppose that $T = \text{Slide}_f$ for a function $f\colon M^c \to N$. Slide_f inherently fulfills the dimensionality reduction property. Let $d \in \mathbb{N}_1$, $d \geq c$, be an arbitrary subsignal dimensionality and let $\xi \in \cup_d(M)$ be a signal. Further, let $i \in \{1, \dots, \dim_M(\xi)-d+1\}$ be an arbitrary subsignal index. Remembering that $\dim_M\,(\text{Subsignal}_d(\xi,\ i)) = d$ and using Lemma 2 we have

$$\begin{aligned} &\text{Slide}_f\,(\text{Subsignal}_d(\xi,\ i)) \\ \overset{\text{D. 5}}{=} &\sum_{j=1}^{d-c+1} f\,(\text{Subsignal}_c(\text{Subsignal}_d(\xi,\ i),\ j)) \cdot e_j^{d-c+1} \\ \overset{\text{L. 2}}{=} &\sum_{j=1}^{d-c+1} f\,(\text{Subsignal}_c(\xi,\ i+j-1)) \cdot e_j^{d-c+1} \\ \overset{\text{D. 5}}{=} &\sum_{j=1}^{d-c+1} \text{Slide}_f(\xi)_{i+j-1} \cdot e_j^{d-c+1} \\ \overset{\text{D. 1}}{=} &\text{Subsignal}_{d-c+1}(\text{Slide}_f(\xi),\ i), \end{aligned}$$

thus the exchange property is satisfied as well. □

Therefore, for each subsignal compatible transformation there is a unique function that *generates* the transformation. We have hence yielded a succinct characterization which helps in deciding whether a given transformation fulfills the dimensionality reduction property and the exchange property. It is further clear that subsignal compatible transformation evaluations themselves can be parallelized since there is no data dependency between individual samples of the outcome.

Reconsidering Fig. 2 it is now obvious that the Quot operator introduced there is nothing but the quotient of two samples evaluated in a sliding fashion. It seems plausible from this example that convolution is also a subsignal compatible transformation. We will prove this rigorously in Sect. III-C.

Before we discuss more theoretical properties, let us first consider an example of a transformation that is *not* subsignal compatible:

Example 7. Let $\mathbb{Z}$ denote the integers and consider the function $T\colon \cup_1(\mathbb{Z}) \to \cup_1(\mathbb{Z})$, $\xi \mapsto (-1)^{\dim_{\mathbb{Z}}(\xi)} \cdot \xi$, which fulfills the dimensionality reduction property with dimensionality reduction constant $c := 1$. The exchange property is, however, not satisfied: Let $d := 1$ and $\xi \in \mathbb{Z}^2$, then for $i := 1$ we have $T(\text{Subsignal}_d(\xi,\ i)) = T(\xi_1) = -\xi_1$, but it is $\text{Subsignal}_{d-c+1}(T(\xi),\ i) = \text{Subsignal}_{d-c+1}(\xi) = \xi_1$. Since $\xi_1 \neq -\xi_1$ unless ξ_1 vanishes, T cannot be a subsignal compatible transformation.

B. *Composition of Subsignal Compatible Transformations*

The composition of subsignal compatible transformations is again a subsignal compatible transformation, where the dimensionality reduction constant has to be adjusted:

Theorem 8. Let M, N and P be sets and let $c_1, c_2 \in \mathbb{N}_1$. Suppose $T_1\colon \cup_{c_1}(M) \to \cup_1(N)$ is a subsignal compatible transformation with dimensionality reduction constant c_1, and $T_2\colon \cup_{c_2}(N) \to \cup_1(P)$ is a subsignal compatible transformation with dimensionality reduction constant c_2.

Define $c := c_1 + c_2 - 1 \in \mathbb{N}_1$. Then $T\colon \cup_c(M) \to \cup_1(P)$, $\xi \mapsto T_2(T_1(\xi))$, is a subsignal compatible transformation with dimensionality reduction constant c.

Proof. We first note that $c \geq 1$ since $c_1 \geq 1$ and $c_2 \geq 1$, hence indeed $c \in \mathbb{N}_1$. Let $\xi \in \cup_c(M)$ be arbitrary for demonstrating that T is well-defined. As $c \geq c_1$ because of $c_2 \geq 1$, we yield $\cup_c(M) \subseteq \cup_{c_1}(M)$ and hence $T_1(\xi)$ is well-defined. Further, we yield $\dim_N(T_1(\xi)) = \dim_M(\xi) - c_1 + 1 \geq c - c_1 + 1 = c_2$ using the dimensionality reduction property of T_1, therefore $T_1(\xi) \in \cup_{c_2}(N)$. Thus $T_2(T_1(\xi))$ is well-defined, and so is T.

For all $\xi \in \cup_c(M)$, the dimensionality reduction property of T_1 and T_2 now implies

$$\begin{aligned}\dim_P(T(\xi)) &= \dim_P(T_2(T_1(\xi)))\\ &\overset{\text{DRP}}{=} \dim_N(T_1(\xi)) - c_2 + 1\\ &\overset{\text{DRP}}{=} \dim_M(\xi) - c_1 + 1 - c_2 + 1\\ &= \dim_M(\xi) - c + 1,\end{aligned}$$

therefore T fulfills the dimensionality reduction property.

Let $d \in \mathbb{N}_1$, $d \geq c$, be arbitrary, and let $\xi \in \cup_d(M)$ and $i \in \{1, \ldots, \dim_M(\xi) - d + 1\}$. Since T_1 and T_2 both satisfy the exchange property we find

$$\begin{aligned}&T(\text{Subsignal}_d(\xi,\ i))\\ =\ &T_2(T_1(\text{Subsignal}_d(\xi,\ i)))\\ \overset{\text{XP}}{=}\ &T_2(\text{Subsignal}_{d-c_1+1}(T_1(\xi,\ i)))\\ \overset{\text{XP}}{=}\ &\text{Subsignal}_{d-c_1+1-c_2+1}(T_2(T_1(\xi)),\ i)\\ =\ &\text{Subsignal}_{d-c+1}(T(\xi),\ i),\end{aligned}$$

where $d \geq c_1$ and $d - c_1 + 1 \geq c_2$ hold during the two respective applications of the exchange property. Therefore, T also fulfills the exchange property. $\square$

This result can be generalized immediately to compositions of more than two subsignal compatible transformations:

Corollary 9. Let $n \in \mathbb{N}$, $n \geq 2$, and let $M_1, \ldots, M_{n+1}$ be sets. For each $\lambda \in \{1, \ldots, n\}$ let $T_\lambda\colon \cup_{c_\lambda}(M_\lambda) \to \cup_1(M_{\lambda+1})$ be a subsignal compatible transformation with dimensionality reduction constant $c_\lambda \in \mathbb{N}_1$. Then the composed function $T\colon \cup_c(M_1) \to \cup_1(M_{n+1})$, $\xi \mapsto \left(\circ_1^{\lambda=n} T_\lambda\right)(\xi)$, is a subsignal compatible transformation with dimensionality reduction constant $c := \sum_{\mu=1}^{n} c_\mu - n + 1 \in \mathbb{N}_1$.

Proof. Define $S_1 := T_1$, and for each $\lambda \in \{2, \ldots, n\}$ let $S_\lambda\colon \cup_{\sum_{\mu=1}^{\lambda} c_\mu - \lambda + 1}(M_1) \to \cup_1(M_{\lambda+1})$, $\xi \mapsto T_\lambda(S_{\lambda-1}(\xi))$, be a function. Since $T = S_n$, the claim follows when it is shown with induction for λ that S_λ is a subsignal compatible transformation with dimensionality reduction constant $\sum_{\mu=1}^{\lambda} c_\mu - \lambda + 1$. While the situation $\lambda = 1$ is trivial, the induction step follows with Theorem 8. $\square$

C. *CNNs without Pooling Layers*

To conclude this section, we demonstrate how CNNs without any pooling layers fit in the theory developed so far. Since pooling layers require a non-trivial extension of the theory, they are detailed in Sect. IV.

Convolutional layers are the most substantial ingredient of CNNs, here the trainable degrees of freedom are located which facilitate adaptation of the network to a specific task. In these layers, multi-channel input feature maps are convolved channel-wise with adjustable filter banks, the result is accumulated and an adjustable bias is added to yield the output feature map.

We begin with introducing indexing rules for iterated structures to account for the multi-channel nature of the occurring signals. Let M be a set, $a, b \in \mathbb{N}_1$ positive natural numbers and $\xi \in (M^a)^b$ a multi-channel signal. It is then $\xi_j \in M^a$ for indices $j \in \{1, \ldots, b\}$, and moreover $(\xi_j)_i \in M$ for indices $j \in \{1, \ldots, b\}$ and $i \in \{1, \ldots, a\}$. This rule is extended in the natural way for sets written explicitly as products with more than two factors. Therefore, if $\xi \in ((M^a)^b)^c$ for another number $c \in \mathbb{N}_1$, then for example $(\xi_k)_j \in M^a$ for indices $k \in \{1, \ldots, c\}$ and $j \in \{1, \ldots, b\}$.

These rules will become more clear if we consider the multi-channel convolution operation $*$. Suppose the samples are members of a ring R, $m \in \mathbb{N}_1$ denotes the number of input channels, $n \in \mathbb{N}_1$ is the number of output channels, and $c \in \mathbb{N}_1$ equals the number of samples considered at any one time during convolution with the filter bank, or in other words the receptive field size of the convolutional layer. Then input signals or feature maps with $D \in \mathbb{N}_1$ samples are of the form $\xi \in (R^m)^D$, and filter banks can be represented by a tensor $w \in ((R^n)^m)^c$. We must have that $D \geq c$, that is the filter kernel should be smaller than the input signal.

The output feature map $(\xi * w) \in (R^n)^{D-c+1}$ is then

$$(\xi * w)_i := \sum_{\lambda=1}^{m} \sum_{\mu=1}^{c} (w_\mu)_\lambda \cdot (\xi_{c+i-\mu})_\lambda \in R^n$$

for indices $i \in \{1, \ldots, D-c+1\}$. Note that $(w_\mu)_\lambda \in R^n$ and $(\xi_{c+i-\mu})_\lambda \in R$, so that the result of their product is understood here as scalar product. The operation is well-defined since $c+i-\mu \in \{1, \ldots, D\}$, which follows immediately through substitution of the extreme values of i and μ.

This multi-channel convolution operation is indeed a subsignal compatible transformation:

Example 10. Define $M := R^m$ and $N := R^n$ and consider

$$f_{\text{conv}} \colon M^c \to N,$$
$$\xi \mapsto \sum_{\lambda=1}^{m} \sum_{\mu=1}^{c} (w_\mu)_\lambda \cdot (\xi_{c-\mu+1})_\lambda.$$

Since $\mu \in \{1, \ldots, c\}$ it is $c-\mu+1 \in \{1, \ldots, c\}$, hence f_{conv} is well-defined. For all $\xi \in M^D$ and any $i \in \{1, \ldots, D-c+1\}$ follows

$$\begin{aligned}
&\text{Slide}_{f_{\text{conv}}}(\xi)_i \\
\overset{\text{D. 5}}{=}\ & f_{\text{conv}}(\text{Subsignal}_c(\xi,\ i)) \\
\overset{\text{D. 1}}{=}\ & f_{\text{conv}}\left(\sum_{\nu=1}^{c} \xi_{i+\nu-1} \cdot e_\nu^c\right) \\
=\ & \sum_{\lambda=1}^{m} \sum_{\mu=1}^{c} (w_\mu)_\lambda \cdot \left(\left(\sum_{\nu=1}^{c} \xi_{i+\nu-1} \cdot e_\nu^c\right)_{c-\mu+1}\right)_\lambda \\
\overset{(\Diamond)}{=}\ & \sum_{\lambda=1}^{m} \sum_{\mu=1}^{c} (w_\mu)_\lambda \cdot (\xi_{i+c-\mu+1-1})_\lambda \\
=\ & (\xi * w)_i,
\end{aligned}$$

where $\nu = c - \mu + 1$ was substituted in the $(\Diamond)$ step. The multi-channel convolution operation as defined above is hence in fact the application of f_{conv} in a sliding fashion. Therefore, Theorem 6 guarantees that $*$ is a subsignal compatible transformation with dimensionality reduction constant c.

Since fully-connected layers are merely a special case of convolutional layers, these do not need any special treatment here. Addition of biases does not require any knowledge on the spatial structure of the convolution's result and is therefore a trivial subsignal compatible transformation with dimensionality reduction constant 1. Non-linearity layers are nothing but the application of a scalar-valued function to all the samples of an input signal. Hence these layers form also subsignal compatible transformations with dimensionality reduction constant 1 due to Theorem 6.

Furthermore, compositions of these operations can also be understood as subsignal compatible transformations with Corollary 9. As a consequence, the exchange property facilitates application of CNNs without pooling layers to an entire signal at once instead of each subsignal independently without incurring any accuracy loss. The next section will extend this result to CNNs that may also feature pooling layers.

IV. Pooling Layers and Functions Applied in a Strided Fashion

So far we have shown how convolutional layers and non-linearity layers of a CNN fit in the theoretical framework of subsignal compatible transformations. In this section, we analyze pooling layers which apply a pooling kernel to non-overlapping blocks of the input signal. This is equivalent to a function applied in a sliding fashion followed by a downsampling operation, and here consequently called application of a function in a *strided* fashion.

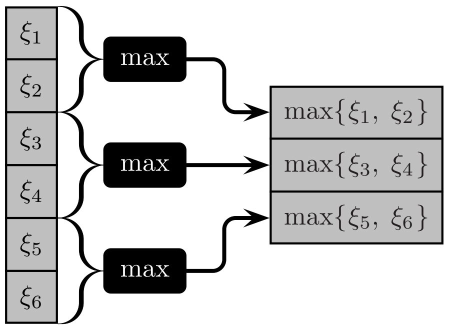


Fig. 3. Illustration of the pooling kernel max that determines the maximum of $k = 2$ adjacent samples. It is here applied in a strided fashion to non-overlapping subsignals of an input signal ξ with $D = 6$ samples, yielding an output signal $\text{Stride}_{\max}(\xi)$ with $\frac{D}{k} = 3$ samples. Since here the dimensionality halves and therefore the dimensionality reduction property is violated, this is *not* a subsignal compatible transformation.

The theory developed herein can of course also be applied to other functions than the pooling kernels encountered in ordinary CNNs. For example, multi-channel convolution where the filter bank is advanced by the receptive field size is essentially f_{conv} from Sect. III-C applied in a strided fashion. Application of convolution where the filter banks are advanced by more than one sample has however no benefit in terms of execution speed for signal-based application. This is discussed at the end of Sect. IV-B when we have developed sufficient theory to analyze this notion.

In this section, we will demonstrate how these functions can be turned into subsignal compatible transformations using a data structure recently introduced as fragmentation by Giusti *et al.* [14]. Here, we will greatly generalize their proposed method and rigorously prove its correctness. As a side effect of our results, we are able to accurately describe the dynamics of the entire signal processing chain, which also includes the possibility of tracking down the position of each processed subsignal in the fragmentation data structure.

Moreover, we analyze under which circumstances the fragment dimensionalities are guaranteed to always be homogeneous. This is a desirable property as it facilitates the application of subsequent operations to signals which all have the same number of samples, rendering cumbersome handling of special cases obsolete and thus resulting in accelerated execution on massively parallel processors. For CNNs this means that conventional tensor convolutions can be used without any modifications whatsoever, which is especially beneficial if a highly-optimized implementation is readily available.

We first state more precisely what the application of a function in a strided fashion means (see Fig. 3 for orientation):

Definition 11. Let M and N be sets, let $k \in \mathbb{N}_1$ be a positive natural number and let $g\colon M^k \to N$ be a function. Then

$$\begin{aligned}&\mathrm{Stride}_g\colon\ \cup_{q=1}^{\infty} M^{kq} \to \cup_1(N),\\ &\xi \mapsto \sum_{i=1}^{\dim_M(\xi)/k} g\left(\mathrm{Subsignal}_k(\xi,\ k(i-1)+1)\right)\cdot e_i^{\dim_M(\xi)/k},\end{aligned}$$

is the operator that applies g in a *strided fashion* to signals where the number of samples is a multiple of k. The subsignal indices are chosen here so that all non-overlapping subsignals are fed through g, starting with the first valid subsignal.

Since it is $k(i-1)+1 \in \{1,\dots,\dim_M(\xi)-k+1\}$ for all $i \in \{1,\dots,\dim_M(\xi)/k\}$, Stride_g is well-defined. We further have $\dim_M(\xi)/\dim_N(\mathrm{Stride}_g(\xi)) = k$ for all ξ in the domain of Stride_g. Since the input dimensionality is here reduced through division with a natural number rather than a subtraction, the dimensionality reduction property cannot be fulfilled unless $k=1$. The situation in which $k=1$ is, however, not particularly interesting since then $\mathrm{Stride}_g = \mathrm{Slide}_g$ which was already handled in Sect. III.

Before continuing with fragmentation, let us discuss multi-channel pooling kernels commonly encountered in CNNs:

Example 12. Suppose we want to process real-valued signals with $m \in \mathbb{N}_1$ channels, that is $M = N = \mathbb{R}^m$, where each channel should be processed independently of the others, and $k \in \mathbb{N}_1$ adjacent samples should be compressed into one output sample. *Average pooling* is then realized by the pooling kernel $g_{\mathrm{avg}}(\xi) := \frac{1}{k}\sum_{\nu=1}^{k}\xi_\nu$, which determines the channel-wise empirical mean value of the samples. Another example is *max-pooling*, where the maximum entry in each channel should be determined. This can be achieved with the pooling kernel $g_{\max}(\xi) := \sum_{\lambda=1}^{m}\left(\max_{\nu=1}^{k}(\xi_\nu)_\lambda\right)\cdot e_\lambda^m$.

A. Fragmentation

The fragmentation operator [14] performs a spatial reordering operation. For its precise analysis, we need to recap some elementary number theory. For all numbers $a \in \mathbb{N}$ and $b \in \mathbb{N}_1$, *Euclidean division* guarantees that there are unique numbers $\mathrm{div}(a,\ b) \in \mathbb{N}$ and $\mathrm{rem}(a,\ b) \in \{0,\dots,b-1\}$ so that

$$a = \mathrm{div}(a,\ b)\cdot b + \mathrm{rem}(a,\ b).$$

Here is a small collection of results on these operators for further reference:

Proposition 13. It is $\mathrm{div}(a,\ 1) = a$ and $\mathrm{rem}(a,\ 1) = 0$ for all $a \in \mathbb{N}$. Moreover, $\mathrm{div}(a+bc,\ c) = \mathrm{div}(a,\ c) + b$ and $\mathrm{rem}(a+bc,\ c) = \mathrm{rem}(a,\ c)$ for all $a, b \in \mathbb{N}$ and $c \in \mathbb{N}_1$.

If the fragmentation operator is applied to a signal, it puts certain samples into individual fragments, which can be grasped as signals themselves. If a collection of fragments is fragmented again, a larger collection of fragments results. The total number of samples is, however, left unchanged after these operations. For the sake of convenience, we will here use matrices as concrete data structure for fragmented signals, where columns correspond to fragments and rows correspond to signal samples.

Let us fix some notation. If M is a set and $a, b \in \mathbb{N}_1$, then $M^{a\times b}$ denotes the set of all matrices with a rows and b columns with entries from M. In our context, this represents a collection of b fragments where each signal has a samples. For $\xi \in M^{a\times b}$ we write $\mathrm{rdim}_M(\xi) = a$ and $\mathrm{cdim}_M(\xi) = b$. Further, $\xi_{i,\ j}$ is the entry in the i-th row and j-th column of ξ where $i \in \{1,\dots,a\}$ and $j \in \{1,\dots,b\}$. The transpose of ξ is written as ξ^T.

The vectorization operator [19] stacks all the columns of a matrix on top of another:

Definition 14. Let M be a set and $a, b \in \mathbb{N}_1$. The *vectorization operator* $\mathrm{vec}_{a\times b}\colon M^{a\times b} \to M^{ab}$ is characterized by $\mathrm{vec}_{a\times b}(\xi)_j = \xi_{\mathrm{rem}(j-1,\ a)+1,\ \mathrm{div}(j-1,\ a)+1}$ for all indices $j \in \{1,\dots,ab\}$ and all matrices $\xi \in M^{a\times b}$. The *inverse vectorization operator* $\mathrm{vec}_{a\times b}^{-1}\colon M^{ab} \to M^{a\times b}$ is given by $\mathrm{vec}_{a\times b}^{-1}(\xi)_{i,\ j} = \xi_{(j-1)a+i}$ for all indices $i \in \{1,\dots,a\}$, $j \in \{1,\dots,b\}$ and all vectors $\xi \in M^{ab}$.

It is straightforward to verify that these two operators are well-defined and inversely related to one another. With their help we may now define the fragmentation operator:

Definition 15. Let M be a set and $k \in \mathbb{N}_1$. For arbitrary vector dimensionalities $q \in \mathbb{N}_1$ and numbers of input fragments $s \in \mathbb{N}_1$ we write

$$\begin{aligned}&\mathrm{Frag}_k\colon M^{kq\times s} \to M^{q\times ks},\\ &\qquad \xi \mapsto \left(\mathrm{vec}_{ks\times q}^{-1}\left(\mathrm{vec}_{s\times kq}\left(\xi^T\right)\right)\right)^T,\end{aligned}$$

for the *fragmentation operator*.

Here, k equals the corresponding parameter from the application of a function in a strided fashion. Frag_k is clearly well-defined, and the number of output fragments is ks. Next consider this operator that undoes the ordering of the fragmentation operator:

Definition 16. Let M be a set, let $k \in \mathbb{N}_1$, and let $q \in \mathbb{N}_1$ denote a vector dimensionality and $s \in \mathbb{N}_1$ a number of output fragments. Then

$$\begin{aligned}&\mathrm{Defrag}_k\colon M^{q\times ks} \to M^{kq\times s},\\ &\qquad \xi \mapsto \left(\mathrm{vec}_{s\times kq}^{-1}\left(\mathrm{vec}_{ks\times q}\left(\xi^T\right)\right)\right)^T,\end{aligned}$$

is called the *defragmentation operator*.

We note that Defrag_k is well-defined and the number of input fragments must equal ks. Fragmentation and defragmentation are inversely related, that is $\mathrm{Defrag}_k \circ \mathrm{Frag}_k = \mathrm{id}_{M^{kq\times s}}$ and $\mathrm{Frag}_k \circ \mathrm{Defrag}_k = \mathrm{id}_{M^{q\times ks}}$. An illustration of the operations performed during fragmentation and defragmentation is depicted in Fig. 4.

Fragmentation is merely a certain reordering operation:

Lemma 17. Let M be a set, $k, q, s \in \mathbb{N}_1$ and $\xi \in M^{kq\times s}$. It is then $\mathrm{rdim}_M(\mathrm{Frag}_k(\xi)) = \frac{1}{k}\cdot\mathrm{rdim}_M(\xi)$, $\mathrm{cdim}_M(\mathrm{Frag}_k(\xi)) = k\cdot\mathrm{cdim}_M(\xi)$, and

$$\mathrm{Frag}_k(\xi)_{\mu,\ \nu} = \xi_{\mathrm{div}((\mu-1)ks+\nu-1,\ s)+1,\ \mathrm{rem}((\mu-1)ks+\nu-1,\ s)+1}$$

for all indices $\mu \in \{1,\dots,q\}$ and $\nu \in \{1,\dots,ks\}$.

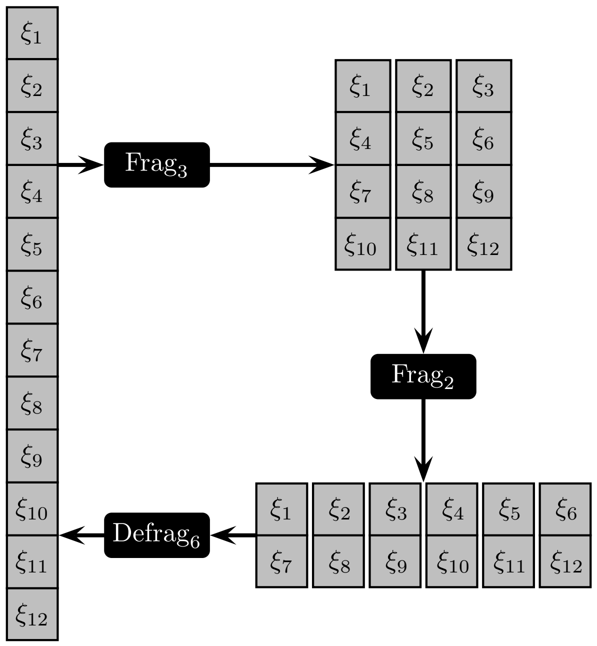


Fig. 4. Illustration of the fragmentation and defragmentation operators. The left-hand side shows an input signal ξ with $q_0 = 12$ samples in a single fragment, that is $s_0 = 1$. Fragmentation with the parameter $k_1 = 3$ yields a signal with $q_1 = \frac{q_0}{k_1} = 4$ samples in each of $s_1 = k_1 s_0 = 3$ fragments, shown at the upper right in the graphics. A second application of fragmentation with $k_2 = 2$ results in $s_2 = k_2 s_1 = 6$ fragments with $q_2 = \frac{q_1}{k_2} = 2$ samples each, depicted at the lower right. A complete defragmentation with parameter $k^* = k_1 k_2 = 6$ yields the original input signal ξ again, with $q_3 = k^* q_2 = 12$ samples in a single fragment, that is $s_3 = \frac{s_2}{k^*} = 1$.

Proof. The dimensionality statements are obvious by the definition of Frag_k. To prove the identity, let $\mu \in \{1, \dots, q\}$ and $\nu \in \{1, \dots, ks\}$. One yields

$$\begin{aligned}&\operatorname{Frag}_k(\xi)_{\mu,\,\nu}\\ \overset{\text{D. 15}}{=}\ &\operatorname{vec}^{-1}_{ks\times q}\left(\operatorname{vec}_{s\times kq}\left(\xi^T\right)\right)_{\nu,\,\mu} \overset{\text{D. 14}}{=} \operatorname{vec}_{s\times kq}\left(\xi^T\right)_{(\mu-1)ks+\nu}\\ \overset{\text{D. 14}}{=}\ &\left(\xi^T\right)_{\operatorname{rem}((\mu-1)ks+\nu-1,\ s)+1,\ \operatorname{div}((\mu-1)ks+\nu-1,\ s)+1},\end{aligned}$$

and the claim follows. □

Similar properties are fulfilled by defragmentation:

Lemma 18. Let M be a set. Let $k, q, s \in \mathbb{N}_1$ be positive natural numbers and $\xi \in M^{q\times ks}$ a fragmented signal. We have that $\operatorname{rdim}_M(\operatorname{Defrag}_k(\xi)) = k \cdot \operatorname{rdim}_M(\xi)$, $\operatorname{cdim}_M(\operatorname{Defrag}_k(\xi)) = \frac{1}{k} \cdot \operatorname{cdim}_M(\xi)$, and

$$\begin{aligned}&\operatorname{Defrag}_k(\xi)_{\mu,\,\nu}\\ &= \xi_{\operatorname{div}((\mu-1)s+\nu-1,\ ks)+1,\ \operatorname{rem}((\mu-1)s+\nu-1,\ ks)+1}\end{aligned}$$

for all indices $\mu \in \{1, \dots, kq\}$, $\nu \in \{1, \dots, s\}$.

Proof. Completely analogous to Lemma 17. □

As already outlined in Fig. 4, compositions of the fragmentation operator are equivalent to a single fragmentation with an adjusted parameterization:

Remark 19. Let M be a set and $k_1, k_2, q, s \in \mathbb{N}_1$. Then $\operatorname{Frag}_{k_2}(\operatorname{Frag}_{k_1}(\xi)) = \operatorname{Frag}_{k_1 k_2}(\xi)$ for all $\xi \in M^{k_1 k_2 q\times s}$.

Proof. Let $\xi \in M^{k_1k_2q\times s}$ be a fragmented signal. We define $A := \operatorname{Frag}_{k_1}(\xi) \in M^{k_2 q\times k_1 s}$, $B := \operatorname{Frag}_{k_2}(A) \in M^{q\times k_1k_2s}$, and $C := \operatorname{Frag}_{k_1k_2}(\xi) \in M^{q\times k_1k_2s}$. Since B and C are of equal size, it is enough to show entry-wise equivalence. Let $\mu \in \{1, \dots, q\}$ and $\nu \in \{1, \dots, k_1k_2s\}$. With Lemma 17 we have that $C_{\mu,\,\nu} = \xi_{\mu_C,\,\nu_C}$ and $B_{\mu,\,\nu} = A_{\mu_B,\,\nu_B} = \xi_{\mu_A,\,\nu_A}$ using the indices

$$\begin{aligned}\mu_C &:= \operatorname{div}((\mu-1)k_1k_2s+\nu-1,\ s)+1,\\ \nu_C &:= \operatorname{rem}((\mu-1)k_1k_2s+\nu-1,\ s)+1,\\ \mu_B &:= \operatorname{div}((\mu-1)k_1k_2s+\nu-1,\ k_1s)+1,\\ \nu_B &:= \operatorname{rem}((\mu-1)k_1k_2s+\nu-1,\ k_1s)+1,\\ \mu_A &:= \operatorname{div}((\mu_B-1)k_1s+\nu_B-1,\ s)+1,\\ \nu_A &:= \operatorname{rem}((\mu_B-1)k_1s+\nu_B-1,\ s)+1.\end{aligned}$$

We thus only have to show that $\mu_C = \mu_A$ and $\nu_C = \nu_A$. It is

$$\begin{aligned}\mu_A = \operatorname{div}(\quad &\operatorname{div}((\mu-1)k_1k_2s+\nu-1,\ k_1s)\cdot k_1s\\ &+\operatorname{rem}((\mu-1)k_1k_2s+\nu-1,\ k_1s),\quad s)+1,\end{aligned}$$

which equals μ_C since $\operatorname{div}(a,\ b)\cdot b + \operatorname{rem}(a,\ b) = a$ for all a, b. Completely analogous follows $\nu_A = \nu_C$. □

It follows immediately that fragmentation is a commutative operation:

Remark 20. If M denotes a set, $k_1, k_2, q, s \in \mathbb{N}_1$ are natural numbers and $\xi \in M^{k_1k_2q\times s}$ is a fragmented signal, then $\operatorname{Frag}_{k_2}(\operatorname{Frag}_{k_1}(\xi)) = \operatorname{Frag}_{k_1}(\operatorname{Frag}_{k_2}(\xi))$.

Proof. Obvious with Remark 19 as multiplication in $\mathbb{N}_1$ is commutative. □

B. Relationship between Fragmentation, Functions Applied in a Strided Fashion and Subsignal Compatible Transformations

We are almost ready for analyzing how functions applied in a strided fashion fit into the theory of subsignal compatible transformations. The outcome of a subsignal compatible transformation applied to a fragmented signal is defined in the natural way:

Definition 21. Let M, N be sets and $T\colon \cup_c(M) \to \cup_1(N)$ a subsignal compatible transformation with dimensionality reduction constant $c \in \mathbb{N}_1$. Let $\xi \in M^{D\times s}$ be a fragmented signal with $D \in \mathbb{N}_1$ samples in each of the $s \in \mathbb{N}_1$ fragments, where $D \geq c$. For $\lambda \in \{1, \dots, s\}$ we write $\xi^{(\lambda)} := \sum_{\nu=1}^{D} \xi_{\nu,\,\lambda} \cdot e_\nu^D \in M^D$ for the individual fragments. The output of T applied to ξ is then defined as

$$T(\xi) := \left[T\left(\xi^{(1)}\right), \dots, T\left(\xi^{(s)}\right)\right] \in N^{(D-c+1)\times s},$$

that is T is applied to all the fragments independently.

Since there is no data dependency between fragments, subsignal compatible transformation evaluation is here straightforward to parallelize over all output samples. Let us now formally introduce the concept of a processing chain, which captures and generalizes all the dynamics of a CNN, and two notions of its application for signal processing:

Definition 22. We call a collection of the following objects a *processing chain*: A fixed subsignal dimensionality

$B \in \mathbb{N}_1$, a number of layers $L \in \mathbb{N}_1$, a sequence of sets $M_0, \dots, M_L, N_1, \dots, N_L$, and for each $j \in \{1, \dots, L\}$ subsignal compatible transformations $T_j \colon \cup_{c_j}(M_{j-1}) \to \cup_1(N_j)$ with dimensionality reduction constant $c_j \in \mathbb{N}_1$ and functions $g_j \colon N_j^{k_j} \to M_j$ where $k_j \in \mathbb{N}_1$. For $j \in \{0, \dots, L\}$ we define $\mathrm{EvalStride}_j \colon M_0^B \to \cup_1(M_j)$,

$$\rho \mapsto \begin{cases} \rho, & \text{if } j = 0, \\ \mathrm{Stride}_{g_j}(T_j(\mathrm{EvalStride}_{j-1}(\rho))), & \text{if } j > 0, \end{cases}$$

as the function that applies the processing chain in a *strided* fashion, and $\mathrm{EvalSlide}_j \colon \cup_B(M_0) \to \cup_1(M_j)$,

$$\xi \mapsto \begin{cases} \xi, & \text{if } j = 0, \\ \mathrm{Frag}_{k_j}(\mathrm{Slide}_{g_j}(T_j(\mathrm{EvalSlide}_{j-1}(\xi)))), & \text{if } j > 0, \end{cases}$$

as the function that applies the processing chain in a *sliding* fashion. We note that these two chains of functions are *not* well-defined unless additional divisibility conditions are fulfilled, detailed below.

The number B here represents the extent of the region that is fed into a CNN, or in other words the entire network's receptive field size. This size is a design parameter of the network and depends on the concrete definitions of all of its layers. The functions T_j in a processing chain can be substituted with the appropriate layer types discussed earlier, like convolutions or non-linearities, or compositions thereof. Pooling kernels and other functions applied in a strided fashion to non-overlapping blocks can be plugged into a processing chain via the g_j functions. The recursive definitions of $\mathrm{EvalStride}_j$ and $\mathrm{EvalSlide}_j$ represent the alternating evaluation of a subsignal compatible transformation and a function applied in a strided fashion up to the specified layer index j.

The rationale for the EvalStride operator is the naive *subsignal-based application* of a CNN: Here, the CNN is applied in the ordinary way to signals of length equal to the network's receptive field size B. According application of the network using a sliding window approach involves extraction of all feasible overlapping subsignals of length B and feeding them through the network independently of each other.

The EvalSlide operator differs from EvalStride in that it corresponds to the *signal-based application* of a CNN: No overlapping subsignals have to be processed separately here, preventing redundant computations. Instead, the entire input signal is processed in one go sharing intermediate computation results among adjacent subsignals. Using EvalSlide, the g_j functions are applied in a sliding rather than a strided fashion, followed by a fragmentation operation.

Definition 22 thus describes a recipe how a CNN can be transformed from subsignal-based application to signal-based application. A concrete example will be discussed in Sect. VI. First, we will concentrate on a theoretical justification that this method indeed produces the correct outcome under all circumstances. The next result states when the application of a processing chain is well-defined, and it proves that the result of the EvalStride operator applied to a subsignal of a larger signal can be found within the result of EvalSlide applied to the entire signal. This then implies that both approaches output the very same values and hence verify EvalSlide involves no accuracy loss whatsoever.

Lemma 23. Suppose we are given a processing chain with the same notation as in Definition 22. Assume k_j divides $\dim_{N_j}(T_j(\mathrm{EvalStride}_{j-1}(\rho)))$ for all $j \in \{1, \dots, L\}$ and all $\rho \in M_0^B$, that is the application of the processing chain in a strided fashion should be well-defined.

Let $k_j^* := \prod_{\nu=1}^{j} k_\nu$ for $j \in \{0, \dots, L\}$ denote the *stride products* for all the layers, which implies $k_0^* = 1$. Let $D \in \mathbb{N}_1$, $D \geq B$, be a signal dimensionality so that the number of subsignals $D - B + 1$ of length B is divisible by k_L^*, and let $\xi \in M_0^D$ be the considered signal. Then the application of the processing chain in a sliding fashion to ξ is well-defined, and we can yield additional statements:

Let $u_j := \dim_{M_j}(\mathrm{EvalStride}_j(\mathrm{Subsignal}_B(\xi,\ i))) \in \mathbb{N}_1$ for all $j \in \{0, \dots, L\}$ be an abbreviation for the dimensionality of the intermediate representations in each layer of the EvalStride cascade. We note that these numbers are actually independent of any subsignal index i. Furthermore, let us define $U_j^{\mathrm{col}} := \mathrm{cdim}_{M_j}(\mathrm{EvalSlide}_j(\xi)) \in \mathbb{N}_1$ and $U_j^{\mathrm{row}} := \mathrm{rdim}_{M_j}(\mathrm{EvalSlide}_j(\xi)) \in \mathbb{N}_1$ for $j \in \{0, \dots, L\}$ as abbreviations for the number of fragments and the fragmented signal dimensionality, respectively, after each layer using the EvalSlide operator. Then for all $i \in \{1, \dots, D - B + 1\}$ and all $j \in \{0, \dots, L\}$ the following holds:

(a) $u_j = \frac{1}{k_j^*}\left(B - \sum_{\mu=1}^{j} k_{\mu-1}^*(c_\mu - 1)\right)$.

(b) $U_j^{\mathrm{row}} = \frac{1}{k_j^*}\left(D - k_j^* + 1 - \sum_{\mu=1}^{j} k_{\mu-1}^*(c_\mu - 1)\right)$ and $U_j^{\mathrm{col}} = k_j^*$.

(c) $U_j^{\mathrm{row}} - u_j + 1 = \frac{1}{k_j^*}(D - B + 1)$. In other words, the number of distinct subsignals with u_j samples in each fragment of the fragmented signals equals the original number of distinct subsignals divided by the corresponding number of fragments.

(d) For all $\mu \in \{1, \dots, u_j\}$ we have that

$$\begin{aligned} &\mathrm{EvalStride}_j(\mathrm{Subsignal}_B(\xi,\ i))_\mu \\ =\ &\mathrm{EvalSlide}_j(\xi)_{\mathrm{div}(i-1,\ k_j^*)+\mu,\ \mathrm{rem}(i-1,\ k_j^*)+1}. \end{aligned}$$

Here, the latter can also be understood as one sample of the $\mathrm{Subsignal}_{u_j}$ operator applied to a certain fragment of $\mathrm{EvalSlide}_j(\xi)$.

Proof. (a) Let $i \in \{1, \dots, D - B + 1\}$ be arbitrary and define $\rho := \mathrm{Subsignal}_B(\xi,\ i) \in M_0^B$ as an abbreviation. It is

$$u_0 = \dim_{M_0}(\mathrm{EvalStride}_0(\rho)) = \dim_{M_0}(\rho) = B,$$

and the right-hand side of the claim trivially equals B for $j = 0$. Carrying out induction we yield for $j - 1 \to j$:

$$\begin{aligned} u_j &\overset{\text{D. 22}}{=} \dim_{M_j}(\mathrm{Stride}_{g_j}(T_j(\mathrm{EvalStride}_{j-1}(\rho)))) \\ &\overset{\text{D. 11}}{=} \frac{1}{k_j}\dim_{N_j}(T_j(\mathrm{EvalStride}_{j-1}(\rho))) \\ &\overset{\text{DRP}}{=} \frac{1}{k_j}\left(\dim_{M_{j-1}}(\mathrm{EvalStride}_{j-1}(\rho)) - c_j + 1\right) \\ &\overset{\text{IH}}{=} \frac{1}{k_j}\left(\frac{1}{k_{j-1}^*}\left(B - \sum_{\mu=1}^{j-1} k_{\mu-1}^*(c_\mu - 1)\right) - (c_j - 1)\right) \end{aligned}$$

$$= \tfrac{1}{k_j k_{j-1}^*}\left(B - \sum_{\mu=1}^{j-1} k_{\mu-1}^*(c_\mu - 1) - k_{j-1}^*(c_j - 1)\right),$$

where IH denotes substitution of the induction hypothesis. Hence, the claimed expression follows since $k_j^* = k_j k_{j-1}^*$. We note that u_j is indeed a natural number because k_j divides $\dim_{N_j}(T_j(\text{EvalStride}_{j-1}(\rho)))$ by requirement.

(b) Besides the statements on U_j^{row} and U_j^{col} we here show that the application of the processing chain in a sliding fashion is well-defined using induction for j. Considering $j = 0$, $\text{EvalStride}_0(\xi) = \xi$ is trivially well-defined and by definition it is $\xi \in M_0^D = M_0^{D\times 1}$. Therefore, $U_0^{\text{row}} = D$ and $U_0^{\text{col}} = 1$ which equals the claimed expressions since $k_0^* = 1$.

For $j - 1 \to j$, we first demonstrate that k_j divides $\chi := \text{rdim}_{M_j}(\text{Slide}_{g_j}(T_j(\text{EvalSlide}_{j-1}(\xi))))$ which implies well-definedness since the fragmentation operator can then indeed be applied. We find that

$$\begin{aligned}
\chi &\overset{\text{D. 5}}{=} \text{rdim}_{N_j}(T_j(\text{EvalSlide}_{j-1}(\xi))) - k_j + 1\\
&\overset{\text{DRP}}{=} \text{rdim}_{M_{j-1}}(\text{EvalSlide}_{j-1}(\xi)) - c_j + 1 - k_j + 1\\
&= U_{j-1}^{\text{row}} - c_j + 1 - k_j + 1\\
&\overset{\text{IH}}{=} \tfrac{1}{k_{j-1}^*}\left(D - k_{j-1}^* + 1 - \sum_{\mu=1}^{j-1} k_{\mu-1}^*(c_\mu - 1)\right)\\
&\quad + \tfrac{1}{k_{j-1}^*}\left(-k_{j-1}^*(c_j - 1) - k_j^* + k_{j-1}^*\right)\\
&= \tfrac{1}{k_{j-1}^*}\left(D - k_j^* + 1 - \sum_{\mu=1}^{j} k_{\mu-1}^*(c_\mu - 1)\right).
\end{aligned}$$

By requirement on the signal length D there exists a number $t \in \mathbb{N}_1$ so that $D - B + 1 = k_L^* t$. Substitution yields

$$\begin{aligned}
\chi &= \tfrac{1}{k_{j-1}^*}\left(B - \sum_{\mu=1}^{j} k_{\mu-1}^*(c_\mu - 1) + k_L^* t - k_j^*\right)\\
&= k_j u_j + k_j \cdots k_L \cdot t - k_j.
\end{aligned}$$

Proposition 13 implies that k_j divides χ since $u_j \in \mathbb{N}_1$ as we have seen in (a), hence the processing chain can be applied until the j-th layer. With Lemma 17 we have $U_j^{\text{row}} = \frac{1}{k_j}\chi$, which immediately yields the claimed expression. As only fragmentation influences the number of columns in the processing chain application, it follows that $U_j^{\text{col}} = k_j U_{j-1}^{\text{col}}$ with Lemma 17, which shows the claimed identity.

(c) Using (a) and (b) we obtain

$$U_j^{\text{row}} - u_j + 1 = \tfrac{1}{k_j^*}\left(D - k_j^* + 1 - B\right) + 1 = \tfrac{D-B+1}{k_j^*},$$

which is a natural number as the number of subsignals was required to be divisible by k_L^*, implying divisibility by k_j^*.

(d) We prove this by induction for j. For $j = 0$, the left-hand side equals $\text{Subsignal}_B(\xi,\ i)_\mu = \xi_{i+\mu-1}$ using Definition 1 for all $i \in \{1, \dots, D - B + 1\}$ and all $\mu \in \{1, \dots, B\}$. Since $k_0^* = 1$ we find with Proposition 13 that the right-hand side is $\xi_{\text{div}(i-1,\ 1)+\mu,\ \text{rem}(i-1,\ 1)+1} = \xi_{i-1+\mu,\ 1}$, hence both sides are equal.

Let us now turn to $j - 1 \to j$. Let $\mu \in \{1, \dots, u_j\}$ be arbitrary, let $i \in \{1, \dots, D - B + 1\}$ be a fixed subsignal index and write $\tau := \text{EvalStride}_{j-1}(\text{Subsignal}_B(\xi,\ i)) \in M_{j-1}^{u_{j-1}}$ as an abbreviation. Considering the left-hand side of the claim we obtain

$$\begin{aligned}
&\text{EvalStride}_j(\text{Subsignal}_B(\xi,\ i))_\mu\\
&\overset{\text{D. 22}}{=} \text{Stride}_{g_j}(T_j(\tau))_\mu\\
&\overset{\text{D. 11}}{=} g_j\left(\sum_{\nu=1}^{k_j} T_j(\tau)_{k_j(\mu-1)+\nu} \cdot e_\nu^{k_j}\right)\\
&\overset{\text{T. 6}}{=} g_j\left(\sum_{\nu=1}^{k_j} T_j\left(\sum_{\lambda=1}^{c_j} \tau_{k_j(\mu-1)+\nu+\lambda-1} \cdot e_\lambda^{c_j}\right) \cdot e_\nu^{k_j}\right)\\
&\overset{\text{IH}}{=} g_j\left(\sum_{\nu=1}^{k_j} T_j\left(\sum_{\lambda=1}^{c_j} \text{EvalSlide}_{j-1}(\xi)_{\psi,\ \omega} \cdot e_\lambda^{c_j}\right) \cdot e_\nu^{k_j}\right),
\end{aligned}$$

where $\psi := \text{div}(i - 1,\ k_{j-1}^*) + k_j(\mu - 1) + \nu + \lambda - 1 \in \mathbb{N}_1$ and $\omega := \text{rem}(i - 1,\ k_{j-1}^*) + 1 \in \mathbb{N}_1$ are abbreviations.

Let $\pi := \text{EvalSlide}_{j-1}(\xi) \in M_{j-1}^{U_{j-1}^{\text{row}} \times U_{j-1}^{\text{col}}}$ be an abbreviation for the analysis of the right-hand side of the claim. We have that

$$\begin{aligned}
&\text{EvalSlide}_j(\xi)_{\text{div}(i-1,\ k_j^*)+\mu,\ \text{rem}(i-1,\ k_j^*)+1}\\
&\overset{\text{D. 22}}{=} \text{Frag}_{k_j}(\text{Slide}_{g_j}(T_j(\pi))_{\text{div}(i-1,\ k_j^*)+\mu,\ \text{rem}(i-1,\ k_j^*)+1}\\
&\overset{\text{L. 17}}{=} \text{Slide}_{g_j}(T_j(\pi))_{\text{div}(\phi,\ k_{j-1}^*)+1,\ \text{rem}(\phi,\ k_{j-1}^*)+1},
\end{aligned}$$

where the number of input fragments to Frag_{k_j} was k_{j-1}^* as shown in (b) and where we have defined

$$\phi := \left(\text{div}(i - 1,\ k_j^*) + \mu - 1\right) k_j k_{j-1}^* + \text{rem}(i - 1,\ k_j^*) \in \mathbb{N}.$$

By the definition of the operators from Euclidean division follows that $\phi = i - 1 + (\mu - 1)\, k_j k_{j-1}^*$. Proposition 13 implies

$$\begin{aligned}
\text{div}(\phi,\ k_{j-1}^*) &= \text{div}(i - 1,\ k_{j-1}^*) + k_j(\mu - 1), \text{ and}\\
\text{rem}(\phi,\ k_{j-1}^*) + 1 &= \text{rem}(i - 1,\ k_{j-1}^*) + 1 = \omega.
\end{aligned}$$

Hence

$$\begin{aligned}
&\text{EvalSlide}_j(\xi)_{\text{div}(i-1,\ k_j^*)+\mu,\ \text{rem}(i-1,\ k_j^*)+1}\\
&\overset{\text{D. 5}}{=} g_j\left(\sum_{\nu=1}^{k_j} T_j(\pi)_{\text{div}(i-1,\ k_{j-1}^*)+k_j(\mu-1)+\nu,\ \omega} \cdot e_\nu^{k_j}\right)\\
&\overset{\text{T. 6}}{=} g_j\left(\sum_{\nu=1}^{k_j} T_j\left(\sum_{\lambda=1}^{c_j} \pi_{\psi,\ \omega} \cdot e_\lambda^{c_j}\right) \cdot e_\nu^{k_j}\right),
\end{aligned}$$

which equals the left-hand side of the claim as we have seen earlier and thus the proof is finished. □

In conclusion, the result of the EvalStride operator applied to an arbitrary subsignal of an input signal emerges contiguously in the result of the EvalSlide operator which processes the entire signal in one go. The concrete position in the fragmentation data structure can be determined with Lemma 23(d). An example for this is depicted in Fig. 5. We will see later how defragmentation can be used eventually to restore the expected order of the resulting samples. It is further intuitively clear that EvalSlide is much more efficient than

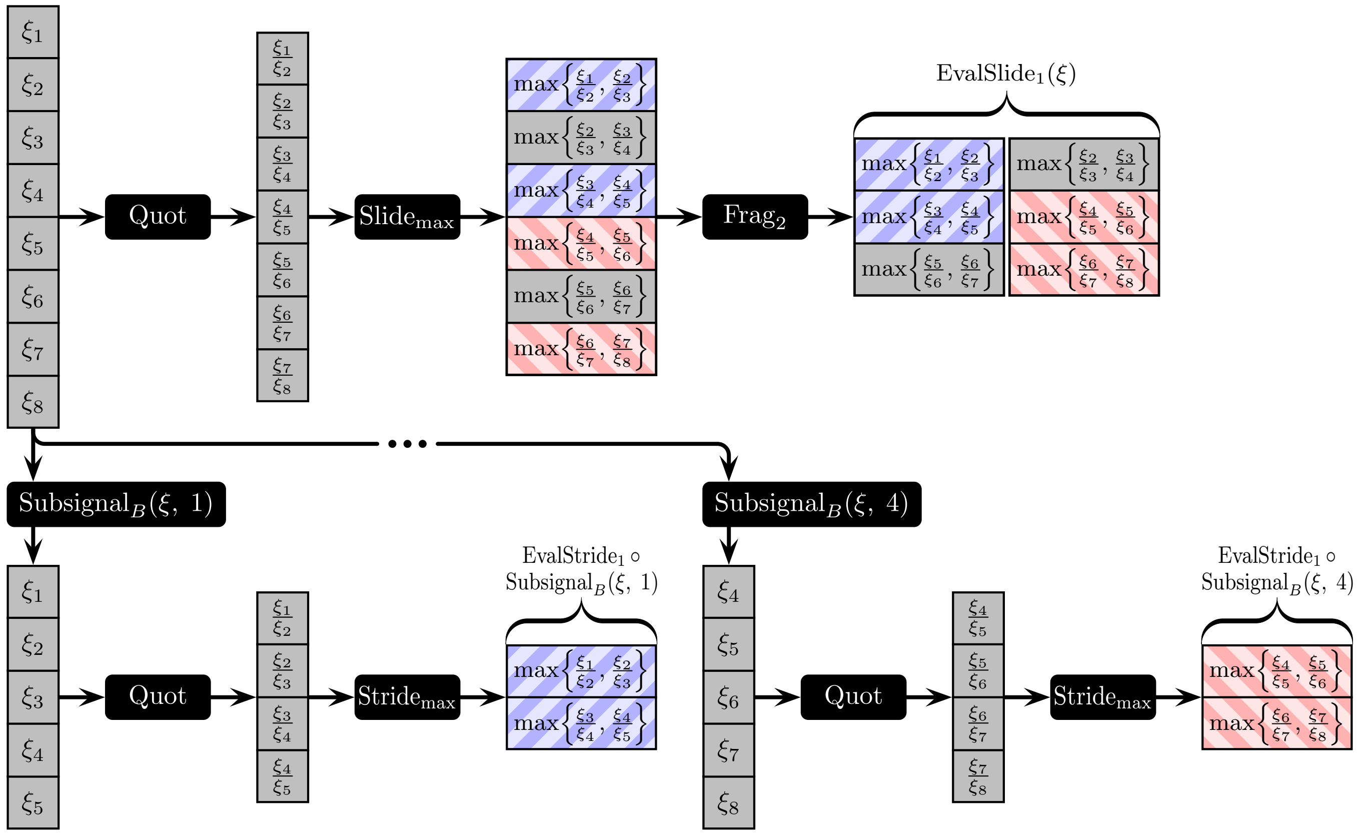


Fig. 5. Example of the two notions of a one-layered ($L = 1$) processing chain with receptive field size $B = 5$ applied to an input signal ξ with $D = 8$ samples. The processing chain consists of the Quot operator as subsignal compatible transformation followed by max-pooling with a stride of $k_1 = 2$. In this example, the processing chain is well-defined and all divisibility requirements are met. The upper part shows the result of EvalSlide evaluated on ξ: Here, max-pooling is applied to Quot(ξ) in a sliding fashion, followed by fragmentation producing two fragments. The lower part shows the outcome of EvalStride applied to two different subsignals of length B. Max-pooling is here used in a strided fashion which halves dimensionality. It is evident that the outcome of EvalStride can be found within $\text{Slide}_{\max}(\text{Quot}(\xi))$, although the individual output signals are interleaved (highlighted with a colored pattern in the graphics). The reordering through fragmentation corrects interleaving so that the outcome is always available contiguously. A second layer in the processing chain has then the ability to further process independent contiguous signals, which is particularly effective on real computing machines.

EvalStride since redundant computations are avoided. We will analyze this rigorously in Sect. V.

Before continuing with the theory, let us discuss two notable special cases. First, a processing chain is of course not required to have a pooling layer following every subsignal compatible transformation, or to even have pooling layers at all. Consider a fixed layer index $j \in \{1, \dots, L\}$. By setting $k_j := 1$ and $g_j \colon N_j \to M_j,\ \tau \mapsto \tau$, where $N_j = M_j$, one sees that Stride_{g_j}, Slide_{g_j} and Frag_{k_j} are the identity functions on their respective domains. In other words, this parameterization of pooling layer j causes it to act like a neutral bypass operation. If, on the other hand, one would want to have two pooling layers one directly after the other, it is completely analogous to achieve a neutral subsignal compatible transformation.

The other special case is to have a convolution which is evaluated in a non-overlapping manner, or equivalently in a strided fashion. This can be achieved by plugging f_{conv} from Sect. III-C into a pooling kernel within a processing chain. Non-overlapping convolution has, however, no advantage in computational complexity if entire input signals should be processed using a sliding window approach: Lemma 23 states that strided fashion has then to be turned into sliding fashion, which is essentially the same as carrying out convolution conventionally by advancing the filter banks by exactly one sample after each evaluation.

C. Defragmentation and Arbitrary Input Signal Length

Lemma 23 requires the length of the input signal to satisfy certain divisibility constraints. For extension of its statements to signals of arbitrary length we need two more operators:

Definition 24. Let $r \in \mathbb{N}$ be a natural number, let M be a set and $\zeta \in M$ be an arbitrary dummy element from M. Then

$$\text{Stuff}_r \colon \ \cup_1(M) \to \cup_{r+1}(M),$$
$$(\xi_1, \dots, \xi_q) \mapsto \sum_{\nu=1}^{q} \xi_\nu \cdot e_\nu^{q+r} + \sum_{\nu=1}^{r} \zeta \cdot e_{q+\nu}^{q+r},$$

is called the *stuffing operator*, which appends r copies of ζ to its argument. Further, we call

$$\text{Trim}_r \colon \ \cup_{r+1}(M) \to \cup_1(M),$$
$$(\xi_1, \dots, \xi_q, \xi_{q+1}, \dots, \xi_{q+r}) \mapsto \sum_{\nu=1}^{q} \xi_\nu \cdot e_\nu^{q},$$

the *trimming operator*, which removes the final r entries from its argument.

The concrete choice of the dummy element ζ does not matter in the following considerations since all output entries which are affected by its choice are trimmed away in the end. We are now in the position to state the main theoretical result of this section:

Theorem 25. Suppose we are given a processing chain with the same notation as in Definition 22, where k_j divides $\dim_{N_j}(T_j(\text{EvalStride}_{j-1}(\rho)))$ for all $j \in \{1, \dots, L\}$ and all $\rho \in M_0^B$, and where $\dim_{M_L}(\text{EvalStride}_L(\rho)) = 1$ for all $\rho \in M_0^B$, that is the output of the entire processing chain applied in a strided fashion consists of exactly one sample.

Let $k_L^* := \prod_{\nu=1}^{L} k_\nu$ denote the stride product of the L-th layer, and let $\tilde{r}\colon \mathbb{N}_1 \to \{0, \dots, k_L^* - 1\}$,

$$\delta \mapsto \begin{cases} 0, & \text{if } k_L^* \text{ divides } \delta - B + 1, \\ k_L^* - \text{rem}(\delta - B + 1,\ k_L^*), & \text{otherwise,} \end{cases}$$

denote the number of dummy samples that have to be padded to an original signal with δ samples to satisfy divisibility requirements. Further define $r\colon \cup_1(M_0) \to \{0, \dots, k_L^* - 1\}$, $\xi \mapsto \tilde{r}(\dim_{M_0}(\xi))$, as an abbreviation that computes the required number of dummy samples in dependence on a concrete original signal ξ.

Consider the function

$$\begin{aligned} &T\colon \cup_B(M_0) \to \cup_1(M_L), \\ &\quad \xi \mapsto \text{Trim}_{r(\xi)}(\text{Defrag}_{k_L^*}(\text{EvalSlide}_L(\text{Stuff}_{r(\xi)}(\xi)))), \end{aligned}$$

which first pads the input signal with as many dummy entries such that each fragmentation operation during application of the processing chain in a sliding fashion comes out even, applies the processing chain in a sliding fashion, defragments the outcome and eventually removes all superfluous entries that emerged from the initial stuffing.

Then T is a subsignal compatible transformation with dimensionality reduction constant B. Furthermore, $T(\rho) = \text{EvalStride}_L(\rho)$ for all $\rho \in M_0^B$ and $T = \text{Slide}_{\text{EvalStride}_L}$.

Proof. We note that $\tilde{r}(\delta)$ is well-defined if k_L^* does not divide $\delta - B + 1$, since then $\text{rem}(\delta - B + 1,\ k_L^*) \in \{1, \dots, k_L^* - 1\}$ and thus $k_L^* - \text{rem}(\delta - B + 1,\ k_L^*) \in \{1, \dots, k_L^* - 1\}$.

Let $\xi \in \cup_B(M_0)$ and write $\tilde{D} := \dim_{M_0}(\xi)$. We have $D := \dim_{M_0}(\text{Stuff}_{r(\xi)}(\xi)) = \tilde{D} + r(\xi)$. If k_L^* divides $\tilde{D} - B + 1$ it is $r(\xi) = 0$, so $\text{rem}(\dim_{M_0}(\text{Stuff}_{r(\xi)}(\xi)) - B + 1,\ k_L^*) = 0$. If on the other hand k_L^* does not divide $\tilde{D} - B + 1$, then $r(\xi) > 0$ and furthermore

$$\begin{aligned} &\dim_{M_0}(\text{Stuff}_{r(\xi)}(\xi)) - B + 1 \\ &= \tilde{D} + k_L^* - \text{rem}(\tilde{D} - B + 1,\ k_L^*) - B + 1. \end{aligned}$$

Hence $\text{rem}(\dim_{M_0}(\text{Stuff}_{r(\xi)}(\xi)) - B + 1,\ k_L^*) = 0$ due to the idempotence of the rem operator. Therefore, in every case is the number of subsignals of $\text{Stuff}_{r(\xi)}(\xi)$ with B samples divisible by k_L^*, as required for application of Lemma 23.

Lemma 23 guarantees that $\pi := \text{EvalSlide}_L(\text{Stuff}_{r(\xi)}(\xi))$ is well-defined. With Lemma 23(b) follows $\text{cdim}_{M_L}(\pi) = k_L^*$. We demanded $\dim_{M_L}(\text{EvalStride}_L(\rho)) = 1$ for all $\rho \in M_0^B$, hence $\text{rdim}_{M_L}(\pi) = \frac{1}{k_L^*}(D - B + 1)$ with Lemma 23(c). Therefore, $\text{Defrag}_{k_L^*}(\pi)$ is well-defined with exactly one output fragment, which has the same number of samples as we have subsignals of length B in the stuffed input signal: $\dim_{M_L}(\text{Defrag}_{k_L^*}(\pi)) = D - B + 1$.

We have that $\dim_{M_L}(\text{Defrag}_{k_L^*}(\pi)) \geq r(\xi) + 1$ because $D = \tilde{D} + r(\xi)$ where $\tilde{D} \geq B$, thus $\text{Trim}_{r(\xi)}(\text{Defrag}_{k_L^*}(\pi))$ is well-defined. Since the trimming operator reduces dimensionality by $r(\xi)$ we find

$$\dim_{M_0}(\xi) - \dim_{M_L}(T(\xi)) = \tilde{D} - (D - B + 1 - r(\xi)) = B - 1.$$

Therefore, T fulfills the dimensionality reduction property with dimensionality reduction constant B.

With Theorem 6 it is thus enough to demonstrate that

$$T\left(\text{Subsignal}_B(\xi,\ i)\right) = T(\xi)_i \in M_L$$

for all $i \in \{1, \dots, \tilde{D} - B + 1\}$ for T to be subsignal compatible. We first show that $T(\rho) = \text{EvalStride}_L(\rho)$ for all $\rho \in M_0^B$, which we will then use to show the weakened exchange property.

Let $\rho \in M_0^B$, then $\text{Subsignal}_B(\text{Stuff}_{r(\rho)}(\rho),\ 1) = \rho$ by the definition of the stuffing operator. We know that $T(\rho)$ consists of a single sample as the dimensionality reduction constant of T is B, hence $T(\rho) = T(\rho)_1$. Extraction of the very first sample of the result of the trimming operator is here equal to the extraction of the very first sample of the trimming operator's argument. Therefore,

$$\begin{aligned} &T(\rho) \\ =\ &\text{Defrag}_{k_L^*}(\text{EvalSlide}_L(\text{Stuff}_{r(\rho)}(\rho)))_{1,\ 1} \\ \overset{\text{L. 18}}{=}\ &\text{EvalSlide}_L(\text{Stuff}_{r(\rho)}(\rho))_{\text{div}(0,\ k_L^*)+1,\ \text{rem}(0,\ k_L^*)+1} \\ \overset{\text{L. 23(d)}}{=}\ &\text{EvalStride}_L(\text{Subsignal}_B(\text{Stuff}_{r(\rho)}(\rho),\ 1))_1 \\ =\ &\text{EvalStride}_L(\rho). \end{aligned}$$

Let us now return to the exchange property. As before let $\xi \in M_0^{\tilde{D}}$, and let $i \in \{1, \dots, \tilde{D} - B + 1\}$ be an arbitrary subsignal index. We can omit the trimming operator as before and yield

$$\begin{aligned} &T(\xi)_i \\ =\ &\text{Defrag}_{k_L^*}(\text{EvalSlide}_L(\text{Stuff}_{r(\xi)}(\xi)))_{i,\ 1} \\ \overset{\text{L. 18}}{=}\ &\text{EvalSlide}_L(\text{Stuff}_{r(\xi)}(\xi))_{\text{div}(i-1,\ k_L^*)+1,\ \text{rem}(i-1,\ k_L^*)+1} \\ \overset{\text{L. 23(d)}}{=}\ &\text{EvalStride}_L(\text{Subsignal}_B(\text{Stuff}_{r(\xi)}(\xi),\ i))_1 \\ \overset{\text{D. 24}}{=}\ &\text{EvalStride}_L(\text{Subsignal}_B(\xi,\ i)) \\ =\ &T\left(\text{Subsignal}_B(\xi,\ i)\right). \end{aligned}$$

Hence T is a subsignal compatible transformation due to Theorem 6. Theorem 4 finally implies $T = \text{Slide}_{\text{EvalStride}_L}$. □

We can thus conclude that CNNs can be turned into efficiently computable subsignal compatible transformations using the EvalSlide operator regardless of the input signal's dimensionality. One could suspect that stuffing the input signal with dummy samples might have a negative effect on the efficiency. However, the number of stuffed samples is always less than the stride product k_L^* of the final layer and hence very small for reasonably sized CNNs.

Moreover, stuffing guarantees that all fragments encountered during evaluation are homogeneous. This enables tensors to be used as the sole data structure for input data, intermediate representations and computation results. This is far more efficient than storing each fragment individually, especially on massively parallel processors where then simple parallelized implementations can achieve maximum throughput.

V. Computational Complexity Analysis

In this section, a detailed theoretical analysis of the computational complexity of processing chain evaluation is carried out. As introduced formally in Definition 22, this corresponds to the alternating application of subsignal compatible transformations and functions applied in a strided fashion. For measuring the computational complexity of the EvalStride and EvalSlide operators, we count the function evaluations required for computing the output of each layer. Regarding the subsignal compatible transformations we consider the unique functions that generate the transformations due to Theorem 6.

We show that EvalSlide requires at most the same number of function evaluations as EvalStride in each layer. This then implies that EvalSlide is more efficient on the global scale of an entire processing chain where all the layers are evaluated subsequently. Further, it is shown that the theoretical speedup can be factorized into simple expressions, facilitating statements on the effect of individual parameters.

A. Identification of the Number of Function Evaluations

Suppose we are in the situation of Lemma 23 where an input signal $\xi \in M_0^D$ and a well-defined processing chain are given. We fix an arbitrary layer index $j \in \{1, \dots, L\}$ and begin by analyzing $\mathrm{EvalStride}_j$ for an arbitrary subsignal index $i \in \{1, \dots, D-B+1\}$. As in the proof of Lemma 23(d) we write $\tau := \mathrm{EvalStride}_{j-1}(\mathrm{Subsignal}_B(\xi,\ i)) \in M_{j-1}^{u_{j-1}}$ and find $\mathrm{EvalStride}_j(\mathrm{Subsignal}_B(\xi,\ i)) = \mathrm{Stride}_{g_j}(T_j(\tau))$. Now Theorem 6 implies that there is exactly one function $f_j \colon M_{j-1}^{c_j} \to N_j$ with $T_j = \mathrm{Slide}_{f_j}$. Thus

$$T_j(\tau) \overset{\text{D. 5}}{=} \sum_{\mu=1}^{u_{j-1}-c_j+1} f_j\left(\mathrm{Subsignal}_{c_j}(\tau,\ \mu)\right) \cdot e_\mu^{u_{j-1}-c_j+1},$$

therefore $u_{j-1} - c_j + 1$ function evaluations of f_j have to be carried out. Considering the strided application of g_j we first note that $u_{j-1} - c_j + 1 = k_j u_j$ with Lemma 23(a) and further

$$\begin{aligned}&\mathrm{Stride}_{g_j}(T_j(\tau))\\ \overset{\text{D. 11}}{=}&\sum_{\nu=1}^{u_j} g_j\left(\mathrm{Subsignal}_{k_j}(T_j(\tau),\ k_j(\nu-1)+1)\right) \cdot e_\nu^{u_j}.\end{aligned}$$

Here, u_j evaluations of g_j are necessary. Since all function evaluations have to be carried out for each of the $D - B + 1$ possible subsignals, the total number of function evaluations increases proportionately with this factor.

Redundant computations are avoided if $\mathrm{EvalSlide}_j$ is used instead. We here analyze the complexity of processing an individual fragment in layer j. The overall complexity is then yielded by multiplication with the number of input fragments U_{j-1}^{col}. Similar to the proof of Lemma 23(d), let $\lambda \in \{1, \dots, U_{j-1}^{\mathrm{col}}\}$ be a fragment index and define

$$\pi := \sum_{\kappa=1}^{U_{j-1}^{\mathrm{row}}} \mathrm{EvalSlide}_{j-1}(\xi)_{\kappa,\ \lambda} \cdot e_\kappa^{U_{j-1}^{\mathrm{row}}} \in M_{j-1}^{U_{j-1}^{\mathrm{row}}}$$

as an abbreviation for the λ-th input fragment. The output of the j-th layer is

$$\mathrm{EvalSlide}_j(\xi) = \mathrm{Frag}_{k_j}(\mathrm{Slide}_{g_j}(T_j(\mathrm{EvalSlide}_{j-1}(\xi)))).$$

We here neglect the complexity of the fragmentation operator since it is merely a structured permutation with very little overhead. Considering a single fragment, we obtain

$$T_j(\pi) \overset{\text{D. 5}}{=} \sum_{\mu=1}^{U_{j-1}^{\mathrm{row}}-c_j+1} f_j\left(\mathrm{Subsignal}_{c_j}(\pi,\ \mu)\right) \cdot e_\mu^{U_{j-1}^{\mathrm{row}}-c_j+1},$$

accounting for $U_{j-1}^{\mathrm{row}} - c_j + 1$ evaluations of f_j. Application of g_j in a sliding fashion yields

$$\begin{aligned}&\mathrm{Slide}_{g_j}(T_j(\pi))\\ \overset{\text{D. 5}}{=}&\sum_{\nu=1}^{U_{j-1}^{\mathrm{row}}-c_j-k_j+2} g_j\left(\mathrm{Subsignal}_{k_j}(T_j(\pi),\ \nu)\right) \cdot e_\nu^{U_{j-1}^{\mathrm{row}}-c_j-k_j+2},\end{aligned}$$

which requires $U_{j-1}^{\mathrm{row}} - c_j - k_j + 2$ evaluations of g_j, which equals $k_j U_j^{\mathrm{row}}$ evaluations due to Lemma 23(b).

B. Analysis for the Subsignal Compatible Transformation Evaluation Component

For determination of the resulting speedup when redundant computations are avoided, we take the ratio of the number of function evaluations required for the naive approach $\mathrm{EvalStride}_j$ to the number needed when the input signal is processed in one go using $\mathrm{EvalSlide}_j$. Considering f_j this yields

$$S_{f_j} := \frac{(D-B+1)(u_{j-1}-c_j+1)}{U_{j-1}^{\mathrm{col}}(U_{j-1}^{\mathrm{row}}-c_j+1)},$$

where the number of subsignals and the number of fragments was included in the numerator and denominator, respectively. Lemma 23(b) and Lemma 23(c) now imply

$$\frac{D-B+1}{U_{j-1}^{\mathrm{col}}} = U_{j-1}^{\mathrm{row}} - u_{j-1} + 1,$$

and by substituting this and carrying out minor algebraic manipulation one sees that

$$S_{f_j} = 1 + \frac{(U_{j-1}^{\mathrm{row}} - u_{j-1})(u_{j-1} - c_j)}{U_{j-1}^{\mathrm{row}} - u_{j-1} + 1}.$$

Since $1 \leq c_j \leq u_{j-1} \leq U_{j-1}^{\mathrm{row}}$ using Lemma 23(c) it follows that $S_{f_j} \geq 1$, which means $\mathrm{EvalSlide}_j$ requires at most the same number of applications of f_j as $\mathrm{EvalStride}_j$. Merely in the special cases where the extent u_{j-1} of the region fed into layer j equals the length of the signal fragments ($u_{j-1} = U_{j-1}^{\mathrm{row}}$) or the dimensionality reduction constant of the subsignal compatible transformation ($u_{j-1} = c_j$) the speedup attains unity, indicating that both approaches require the same number of function evaluations.

If S_{f_j} is understood as a function in dependence on the signal dimensionality D, then in the expression we derived the only quantity that depends on D is U_{j-1}^{row} since c_j is constant and u_{j-1} is independent of D as can be seen from Lemma 23(a). As Lemma 23 requires $D - B + 1$ to be divisible by k_L^*, the next larger feasible signal dimensionality is $D + k_L^* =: D_+$. Subtracting S_{f_j} evaluated for signal dimensionality D from its value for an extended signal dimensionality D_+ yields

$$\begin{aligned} & S_{f_j}(D_+) - S_{f_j}(D) \\ &= \frac{(u_{j-1} - c_j)(u_{j-1} - c_j + 1)(U_{j-1}^{\text{row}}(D_+) - U_{j-1}^{\text{row}}(D))}{(U_{j-1}^{\text{row}}(D_+) - c_j + 1)(U_{j-1}^{\text{row}}(D) - c_j + 1)}. \end{aligned}$$

Now Lemma 23(b) implies that $U_{j-1}^{\text{row}}(D_+) - U_{j-1}^{\text{row}}(D) = \frac{k_L^*}{k_{j-1}^*}$, therefore $S_{f_j}(D_+) \geq S_{f_j}(D)$ and thus the speedup increases if the signal dimensionality is increased. In the limit case of arbitrarily large input signals one obtains $\lim_{D \to \infty} S_{f_j}(D) = u_{j-1} - c_j + 1$, that is the speedup asymptotically attains a finite value. From this it is evident that greatest speedups can be achieved for large regions of interest u_{j-1} and small dimensionality reduction constants c_j.

C. Analysis for the Strided Function Evaluation Component

Let us now turn to the function g_j which is applied in a strided fashion. Incorporating the number of subsignals and fragments for EvalStride_j and EvalSlide_j, respectively, we obtain the ratio

$$S_{g_j} := \frac{(D - B + 1)u_j}{U_{j-1}^{\text{col}} k_j U_j^{\text{row}}} = 1 + \frac{(U_j^{\text{row}} - u_j)(u_j - 1)}{U_j^{\text{row}}}.$$

With Lemma 23(c) we have that $S_{g_j} \geq 1$, hence here the complexity of EvalSlide_j is also less than that of EvalStride_j. The speedup is unity only if $u_j = U_j^{\text{row}}$ or $u_j = 1$.

Grasping S_{g_j} as a function of D and denoting the next larger signal dimensionality with $D_+ := D + k_L^*$, one obtains

$$S_{g_j}(D_+) - S_{g_j}(D) = \frac{u_j(u_j - 1)(U_j^{\text{row}}(D_+) - U_j^{\text{row}}(D))}{U_j^{\text{row}}(D_+) U_j^{\text{row}}(D)} \geq 0.$$

Thus speedups increase with larger signal dimensionality. In the limit case of arbitrarily large input signals it is $\lim_{D \to \infty} S_{g_j}(D) = u_j$, hence here the speedup bounded from above as well.

D. Discussion

We have derived simple expressions that imply the number of function evaluations required by EvalSlide is always less or equal than that of EvalStride in each layer. Therefore this also holds for the subsequent evaluation of all the layers. We have further identified the special cases in which the computational complexity of EvalSlide matches that of EvalStride. Moreover, we have demonstrated that the speedup becomes more significant for larger input signals, although its growth is not unbounded.

Our analysis was restricted to the amount of necessary function evaluations only and neglected the parallelization potential of the individual approaches. On a massively parallel processor, the throughput of the EvalSlide approach might be substantially lower than that of EvalStride since here coarse-grained parallelism on the subsignal level can be exploited facilitating load balancing on thousands of parallel computing cores. However, we demonstrate through experiments in the next section that, as predicted by the theoretical analysis, in practice EvalSlide is orders of magnitude faster than EvalStride.

VI. Practical Considerations and Experimental Evaluation

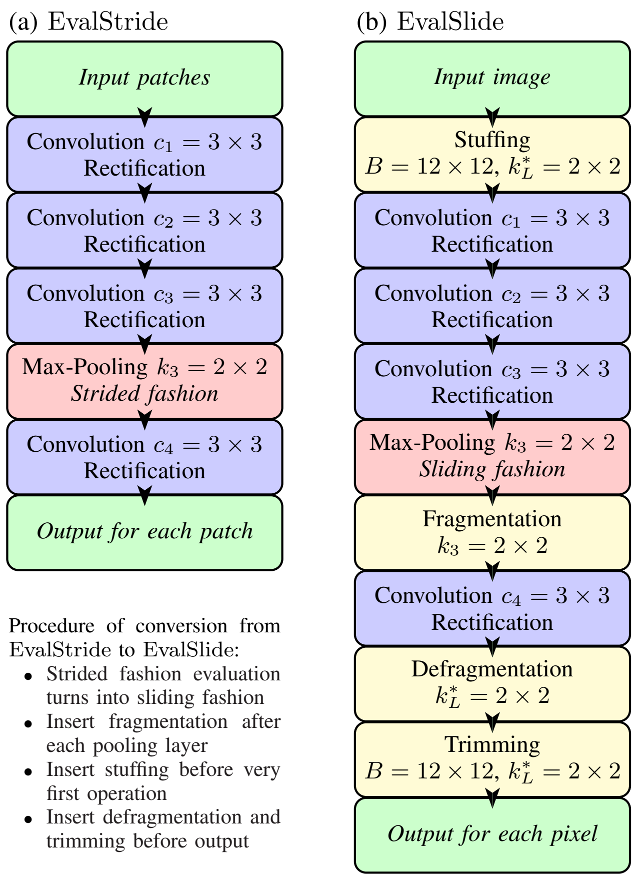
This section discusses practical considerations when using the theory proposed in this paper and reports the results of numerical experiments carried out to confirm significant speedups can be achieved when real processors are employed. For this, CNNs with different architectures were applied to entire two-dimensional images using the EvalStride and EvalSlide operators. For the latter, the methodology proposed in Theorem 25 has been employed, that is the original images were stuffed before EvalSlide was applied, and the outcome was defragmented and trimmed. All degrees of freedom of the CNNs were initialized with random numbers and the CNNs were applied to random image data. This is no restriction to the generality of our results since here we concentrate on an analysis of numerical correctness and runtime measurements rather than on concrete performance in a real use case.

A. Generalization to 2D Image Data

The generalization of our theory to two spatial dimensions for image processing is straightforward. In this context, we will refer to two-dimensional subsignals as *patches* as is common in image processing. It is sufficient to use 4D tensors as the only data structure representing input data, intermediate representations and computation results: One dimension accounts for the feature map index and two dimensions account for the spatial position within each feature map. The fourth dimension represents an image index. For EvalStride this corresponds to the patch index and for EvalSlide this is the fragment index. Although patch indices and fragment indices are two-dimensional in image processing, it is straightforward to linearize these indices into one dimension since the number of patches and fragments, respectively, can be determined beforehand for each spatial dimension. As operations are carried out independently both for patches and fragments (see Definition 21) this perfectly matches the meaning of the image index dimension in common 4D tensor processing [20].

B. Experimental Methodology

The CNNs used for evaluation were created with a varying number of convolutional layers $L \in \{1, \dots, 15\}$ using the following scheme. Similar to [21], each convolutional layer was parameterized for a filter size of 3×3 pixels, where for simplicity the number of output feature maps was always set to 128. The output of each convolutional layer was fed through a rectification non-linearity [22]. A 2×2 max-pooling layer was inserted after each three pairs of convolution and


(a) EvalStride
Input patches
Convolution $c_1 = 3 \times 3$ Rectification
Convolution $c_2 = 3 \times 3$ Rectification
Convolution $c_3 = 3 \times 3$ Rectification
Max-Pooling $k_3 = 2 \times 2$ Strided fashion
Convolution $c_4 = 3 \times 3$ Rectification
Output for each patch
(b) EvalSlide
Input image
Stuffing $B = 12 \times 12$, $k_L^* = 2 \times 2$
Convolution $c_1 = 3 \times 3$ Rectification
Convolution $c_2 = 3 \times 3$ Rectification
Convolution $c_3 = 3 \times 3$ Rectification
Max-Pooling $k_3 = 2 \times 2$ Sliding fashion
Fragmentation $k_3 = 2 \times 2$
Convolution $c_4 = 3 \times 3$ Rectification
Defragmentation $k_L^* = 2 \times 2$
Trimming $B = 12 \times 12$, $k_L^* = 2 \times 2$
Output for each pixel

Procedure of conversion from EvalStride to EvalSlide:

- Strided fashion evaluation turns into sliding fashion
- Insert fragmentation after each pooling layer
- Insert stuffing before very first operation
- Insert defragmentation and trimming before output

Fig. 6. Example network architecture for $L = 4$ targeted at the processing of two-dimensional images. For EvalStride (left-hand side) suitable for patch-based application, this architecture consists of four 3×3 convolutional layers, where after the third convolutional layer 2×2 max-pooling is carried out in a strided fashion. The right-hand side shows how this architecture has been transformed for the EvalSlide approach suitable for image-based application: Pooling is now applied in a sliding fashion and followed by fragmentation. The input image is stuffed to guarantee fragmentation always comes out even and hence homogeneous 4D tensors can be used as the only data structure. The effects of stuffing and fragmentation are undone in the end, yielding an output for each pixel in the input image. Here, B denotes the entire CNN's receptive field size and k_L^* is the stride product of the entire network.

rectification layers, unless the pooling layer would be the final layer of the network. This procedure also resembles the proposed architecture of [21]. Figure 6 depicts a network architecture as it was used for the EvalStride operator and how it was transformed to account for image-based application using the EvalSlide operator.

The experiments were run on a massively parallel GPU implementation and on a parallel CPU implementation. For the GPU variant, an NVIDIA GeForce GTX 980 Ti graphics card was used. Convolutions, rectification transfer functions and max-pooling in a strided fashion were realized through cuDNN [20], a software library provided by the manufacturer of the graphics card to achieve maximum efficiency. The remaining operations required for EvalSlide were implemented as custom NVIDIA CUDA kernels. The CPU implementation employed an Intel Core i7-5930K processor. Convolutions and max-pooling in a sliding fashion were sped up with the Intel Integrated Performance Primitives software library [23]. Parallel execution was here achieved through OpenMP [24].

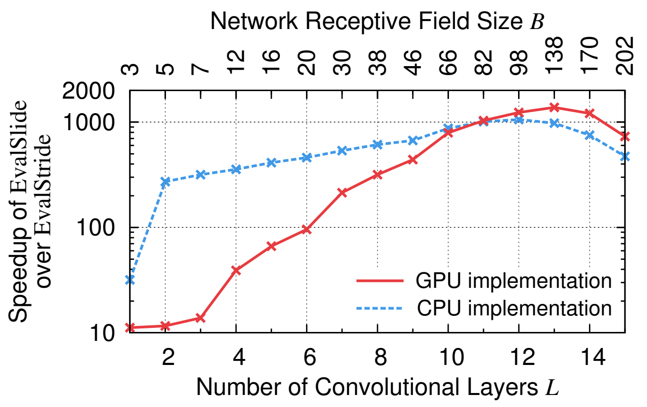


Fig. 7. Speedup factors of EvalSlide over EvalStride as measured on GPU and on CPU in dependence on the number of convolutional layers L in the networks. The second axis on top depicts the resulting receptive field size B of the entire network in either spatial dimension. Note that these values are non-linear in L since pooling layers were inserted at regular intervals in the networks. The measurements indicate that especially for deep networks huge speedups can be yielded if images are processed in one go.

We used random images with a height of 240 pixels, a width of 320 pixels and a single feature map as input for the networks. First, it was verified numerically using twenty random images that EvalStride and EvalSlide produced the same results for all network architectures. Then, the time required for carrying out both operators on GPU and on CPU was measured and the ratio taken to determine the speedup. All measurements were repeated twenty times on twenty distinct input images, and the average of the resulting four hundred runs was used for further evaluation.

For EvalStride, we did neither include the time required for extracting all feasible patches and storing them in a dedicated tensor nor did we include the time required for assembling the final output tensor in the measurements. Due to memory constraints, the tensor with all the patches had to be split in batches before processing on the GPU. The batch size was maximized with respect to the available GPU memory to ensure maximum throughput of the graphics card. For EvalSlide, we did include the time required for stuffing, fragmentation, defragmentation and trimming in the measurements. Here, splitting into batches was not necessary since redundancies in overlapping patches are avoided and the memory demands were therefore very low. Any measured speedups are hence biased in favor of EvalStride as here only computations but no overhead in organizing data structures were considered.

C. Experimental Results and Discussion

Figure 7 shows the achieved speedups both for GPU and CPU of EvalSlide over EvalStride in dependence on the number of convolutional layers L in the network. Although the input images were rather small, a decent speedup could be determined even for very flat networks. Even for $L = 1$ significant speedups could be measured although here the theoretical number of function evaluations is equal for both operators. However, there is still a huge memory redundancy in the tensor storing all the patches necessary for EvalStride, which is disadvantageous in terms of memory throughput. While the CPU implementation achieves speedup factors beyond one hundred for $L \geq 2$, the GPU implementation requires deeper networks with $L \geq 7$ for a similar speedup. This is because of the GPU's superior parallelization capabilities which facilitate an overproportional throughput of the $\mathrm{EvalStride}$ operator where the patches can be processed in parallel.

We further note a peak in the speedup for $L = 13$ and $L = 12$ for GPU and CPU, respectively. If deeper networks are used, the relative speedup decreases but remains on a high level still. The reason for this is the large receptive field size B of such deep networks: Since the patch size almost matches the image dimensions, there are only relatively few patches compared to the situation of a smaller receptive field size. As predicted by the theoretical results from Sect. V, the degree of redundancy of $\mathrm{EvalStride}$ decreases in this case resulting in a decreased relative speedup.

Finally, we compared execution times of $\mathrm{EvalSlide}$ using the GPU implementation versus the CPU implementation. Averaging over $L \in \{1, \dots, 15\}$ we yielded a relative speedup of the GPU implementation of factor 89 over the CPU implementation, demonstrating the massive parallelization potential.

VII. Conclusions

This paper introduced and analyzed the concept of subsignal compatible transformations, functions that allow exchanging subsignal extraction with function evaluation without any effect on the outcome. In doing so, it was demonstrated rigorously how CNNs can be applied efficiently without accuracy loss to large signals using a sliding window approach while eliminating redundant computations and special case treatment. A theoretical analysis has proven the computational complexity of processing an input signal in one go is inferior to subsignal-based application, which was subsequently verified through numerical experiments. The arguments given in the rigorous proofs can be used to verify whether a given implementation is correct, or they can serve as basis for algorithmic realizations. In either case is an extensive mathematical framework available that facilitates analysis of rapid exact signal scanning schemes.

Appendix
Multi-Scale Transformations

The main part of the paper has shown how CNNs can be efficiently evaluated on entire images through the theory of subsignal compatible transformations. We now consider functions that take multiple spatial resolutions of a single signal as input. Since here the context of local regions is incorporated additionally, this approach has proven highly effective in classification tasks [12]. We here assume that the number of samples considered at any one time is fixed for all scale levels. This facilitates the design of scale-invariant representations, for example by using the same classifier for all scales of the input [12].

A signal is downscaled by application of a lowpass filter to reduce aliasing artifacts, followed by a downsampling operator which returns a subset of equidistant samples. When a subsignal is extracted from a downscaled input signal, it should contain a downscaled copy of the corresponding subsignal from the original input signal. This requires boundary-handling of the input signal, since for example the very first subsignal cannot be extended to allow for a larger context by means of only the original samples.

In the following, let $\mathbb{Z}$ denote the integers and let $\lceil \cdot \rceil$ denote the ceiling function that rounds up its argument to the next larger natural number. Let us first formalize the concepts of boundary handling and subsignal extraction subject to boundary handling:

Definition 26. Let M be a set, let $d \in \mathbb{N}_1$ denote a subsignal dimensionality and let $R \in \mathbb{N}$ be a boundary size.

(a) A function $\vartheta \colon \cup_1(M) \times \mathbb{Z} \to M$ is called *boundary-handling function* if and only if $\vartheta(\xi,\ \nu) = \xi_\nu$ for all $\xi \in \cup_1(M)$ and all $\nu \in \{1, \dots, \dim_M(\xi)\}$.

(b) We call the function

$$\mathrm{Pad}_R^\vartheta \colon \cup_1(M) \to \cup_{1+2R}(M),$$
$$\xi \mapsto \sum_{\nu=1}^{\dim_M(\xi)+2R} \vartheta(\xi,\ \nu - R) \cdot e_\nu^{\dim_M(\xi)+2R},$$

which extends signals at both ends with R samples subject to the boundary-handling function ϑ the *padding operator*.

(c) The function

$$\mathrm{SubsignalPad}_{(d,R)}^\vartheta \colon$$
$$\bigcup_{D=d}^{\infty} \Big(M^D \times \{1, \dots, D-d+1\} \Big) \to M^{d+2R},$$
$$(\xi,\ i) \mapsto \sum_{\nu=1}^{d+2R} \vartheta(\xi,\ i + \nu - R - 1) \cdot e_\nu^{d+2R},$$

that extracts subsignals subject to the boundary-handling function ϑ and implicitly pads R samples at both ends is called the *padded subsignal extraction operator*.

Next, we define how we extract downscaled subsignals using the concepts just introduced:

Definition 27. Let M be a set and let $k \in \mathbb{N}_1$ denote a downsampling step size.

(a) We call

$$\mathrm{Down}_k \colon \cup_1(M) \to \cup_1(M),$$
$$\xi \mapsto \sum_{\nu=1}^{\lceil \dim_M(\xi)/k \rceil} \xi_{k(\mu-1)+1} \cdot e_\nu^{\lceil \dim_M(\xi)/k \rceil},$$

the *downsampling operator*, which extracts samples from equidistant locations.

(b) The function

$$\mathrm{MultiScaleIndex}_k \colon \mathbb{N}_1 \to \mathbb{N}_1$$
$$i \mapsto k \cdot \mathrm{div}(i-1,\ k) + 1,$$

is called the *multi-scale subsignal index transformation*.

(c) Suppose $H \colon M^h \to M$ is a lowpass filter kernel of size $h \in \mathbb{N}_1$, where $h \geq k$ should hold to avoid aliasing artifacts. Further, let $d \in \mathbb{N}_1$ be a subsignal dimensionality,

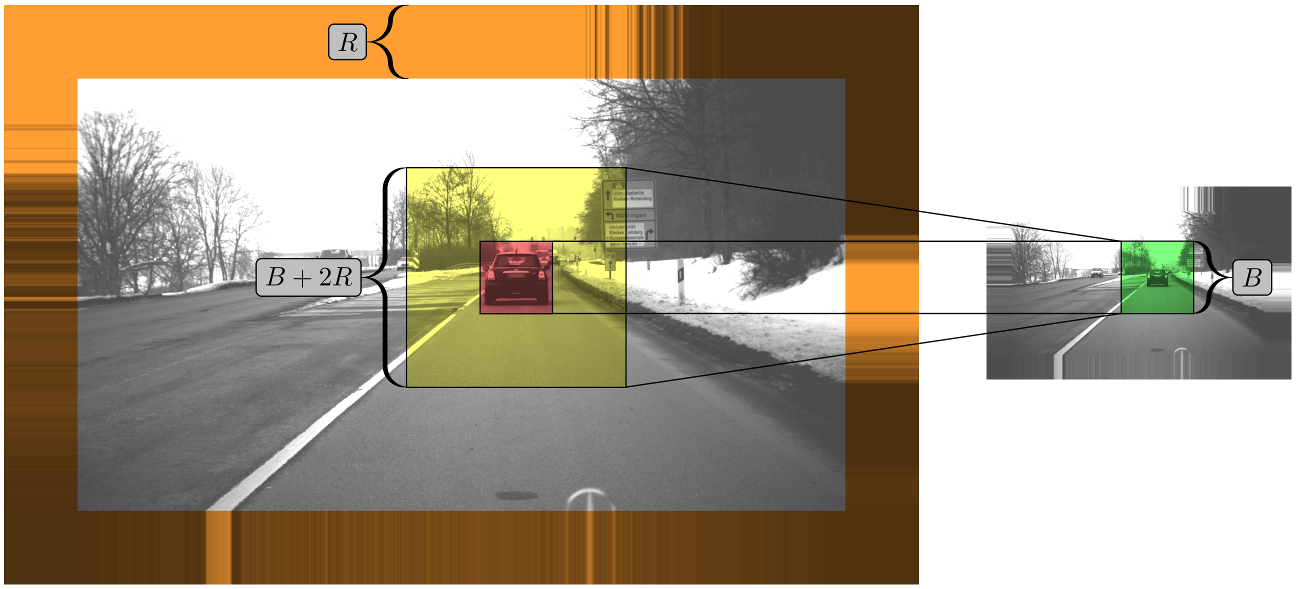


Fig. 8. Illustration of the multi-scale analysis approach detailed in Lemma 28. The original image (gray area on the left-hand side) is padded with R pixels (orange area) using the Neumann boundary condition. Downscaling of the padded image with a scale factor of $k = 3$ yields the small image on the right-hand side. The red area in the original image is a region of interest with B pixels in either dimension, it can be extracted conventionally using the Subsignal operator. The SubsignalPad operator extracts an extended region of interest with $B + 2R$ pixels in either dimension (yellow area), which provides more context than the original region of interest. If that extended region is downscaled, it is guaranteed by the choice of R that the resulting signal possesses B samples in either dimension. This downscaled, padded region is equivalent to application of the MultiScaleSubsignal operator. Its outcome can also be found in the downscaled, padded image (green area) if the indices are adjusted through the MultiScaleIndex function. Figure best viewed in color.

$R \in \mathbb{N}$ a boundary size and $\vartheta\colon \cup_1(M) \times \mathbb{Z} \to M$ a boundary-handling function. Then

$$\begin{aligned}&\text{MultiScaleSubsignal}^{(\vartheta,H)}_{(d,R,k)}\colon\\ &\bigcup_{D=d}^{\infty}\Big(M^D \times \{1,\dots,D-d+1\}\Big) \to \cup_1(M),\\ &(\xi,\ i) \mapsto \text{Down}_k(\text{Slide}_H(\text{SubsignalPad}^{\vartheta}_{(d,R)}(\xi,\ i))),\end{aligned}$$

is called the *multi-scale subsignal extraction operator*.

There are a few requirements so that extraction of downscaled subsignals makes sense. Most importantly is the correct determination of the boundary size R in the definition of the MultiScaleSubsignal operator. It should be chosen so that the extracted subsignals from each scale level are always exactly centered around the corresponding subsignals from the original scale level. It is moreover beneficial if the entire input signal can be downscaled in one go, so that the output of the MultiScaleSubsignal operator equals simple extraction of subsignals from that downscaled signal.

However, if this approach is pursued there are subsignals in the original signal which do not possess a downscaled counterpart in this representation. The MultiScaleIndex function alleviates this problem through computation of an appropriate subsignal index which is guaranteed to possess a downscaled counterpart. Although this is merely an approximation, it is assured that the correct subsignal index in the downscaled signal is always less than one sample off. The next result formalizes these thoughts, an illustration of its statements is given in Fig. 8.

Lemma 28. Let M be a set, and let $k \in \mathbb{N}$ be a downsampling step size where we require that $k \geq 2$. Moreover, let $H\colon M^h \to M$ be a lowpass filter kernel of size $h \in \mathbb{N}_1$, $h \geq k$, and let $B \in \mathbb{N}_1$ be a subsignal dimensionality and suppose $\vartheta\colon \cup_1(M)\times\mathbb{Z} \to M$ is a boundary-handling function.

Define $\tilde{R} := (k-1)B+h-k \in \mathbb{N}$ and $R := \lceil \tilde{R}/2 \rceil \in \mathbb{N}$ as the boundary size. Assume we are given a signal $\xi \in \cup_B(M)$ and let us write $D := \dim_M(\xi)$ as an abbreviation. Ultimately, let $\pi := \text{Down}_k(\text{Slide}_H(\text{Pad}^{\vartheta}_R(\xi))) \in \cup_1(M)$ denote the downscaled signal. Then the following holds:

(a) $\text{Subsignal}_B(\xi,\ i)_\mu = \text{SubsignalPad}^{\vartheta}_{(B,R)}(\xi,\ i)_{\mu+R}$ for all $\mu \in \{1,\dots,B\}$ and all $i \in \{1,\dots,D-B+1\}$, that is the padded subsignals are centered around the original subsignals.

(b) $\dim_M(\pi) = \left\lceil \frac{D-B+1+\text{rem}(\tilde{R},\ 2)}{k} \right\rceil + B - 1$, hence there are at least $\left\lceil \frac{D-B+1}{k} \right\rceil$ subsignals with B samples in π, and at most one additional subsignal.

(c) Let $i \in \{1,\dots,D-B+1\}$ be a subsignal index and write $j := \text{MultiScaleIndex}_k(i)$ as the result of the index transformation. Then $j \in \{1,\dots,D-B+1\}$, $j \leq i$ and $i - j < k$. The index adjustment hence decreases subsignal indices by at most $k-1$ samples with respect to the original scale level.

(d) It is

$$\begin{aligned}&\text{Subsignal}_B(\pi,\ \text{div}(i-1,\ k)+1)\\ &= \text{MultiScaleSubsignal}^{(\vartheta,H)}_{(B,R,k)}(\xi,\ \text{MultiScaleIndex}_k(i))\end{aligned}$$

for all $i \in \{1,\dots,D-B+1\}$. In other words, the subsignals from π equal downscaled subsignals from the original signal ξ where the subsignal index was adjusted through MultiScaleIndex_k.

Proof. (a) If $\mu \in \{1, \dots, B\}$ and $i \in \{1, \dots, D - B + 1\}$, then

$$\begin{aligned}&\operatorname{SubsignalPad}^{\vartheta}_{(B,R)}(\xi,\ i)_{\mu+R}\\ \overset{\text{D. 26}}{=}\ &\vartheta(\xi,\ i+\mu-1) \overset{(\Diamond)}{=} \xi_{i+\mu-1} \overset{\text{D. 1}}{=} \operatorname{Subsignal}_B(\xi,\ i)_\mu,\end{aligned}$$

where in the $(\Diamond)$ step we have used that $i+\mu-1 \in \{1, \dots, D\}$. Here, the boundary handling function evaluates to an original sample of the input signal. Hence all the samples in the middle of $\operatorname{SubsignalPad}^{\vartheta}_{(B,R)}(\xi,\ i)$ stem from the input signal ξ and are *not* subject to boundary conditions.

(b) We first note that

$$2R = \tilde{R} + \operatorname{rem}(\tilde{R},\ 2) = (k-1)B + h - k + \operatorname{rem}(\tilde{R},\ 2)$$

by the definition of the ceiling function, which is marked with $(\Diamond)$ in the following. Therefore

$$\begin{aligned}\dim_M(\pi) &= \dim_M(\operatorname{Down}_k(\operatorname{Slide}_H(\operatorname{Pad}^{\vartheta}_R(\xi))))\\ &\overset{\text{D. 27}}{=} \left\lceil \tfrac{1}{k} \cdot \dim_M(\operatorname{Slide}_H(\operatorname{Pad}^{\vartheta}_R(\xi))) \right\rceil\\ &\overset{\text{D. 5}}{=} \left\lceil \tfrac{1}{k} \left(\dim_M(\operatorname{Pad}^{\vartheta}_R(\xi)) - h + 1\right) \right\rceil\\ &\overset{\text{D. 26}}{=} \left\lceil \tfrac{1}{k} \left(\dim_M(\xi) + 2R - h + 1\right) \right\rceil\\ &\overset{(\Diamond)}{=} \left\lceil \tfrac{D-B+1+\operatorname{rem}(\tilde{R},\ 2)}{k} + B - 1 \right\rceil\\ &= \left\lceil \tfrac{D-B+1+\operatorname{rem}(\tilde{R},\ 2)}{k} \right\rceil + B - 1,\end{aligned}$$

where we have used that $B - 1 \in \mathbb{N}$ in the final step so that this term could be moved outside of the ceiling function. Since $\operatorname{rem}(\tilde{R},\ 2) \in \{0,\ 1\}$ there is at most one superfluous subsignal of length B in π, which is irrelevant in the following discussion.

(c) Let i and $j := k \cdot \operatorname{div}(i-1,\ k) + 1$ be given as in the claim. We clearly have $j \in \mathbb{N}$. Since $\operatorname{div}(i-1,\ k) \geq 0$ follows $j \geq 1$. On the other hand, using Euclidean division we obtain $k \cdot \operatorname{div}(i-1,\ k) + 1 = i - \operatorname{rem}(i-1,\ k) \leq i \leq D - B + 1$ because $\operatorname{rem}(i-1,\ k) \geq 0$. Analogously follows $i - j = \operatorname{rem}(i-1,\ k) \in \{0, \dots, k-1\}$, which proves the claimed inequalities.

(d) Let $i \in \{1, \dots, D - B + 1\}$ be an arbitrary subsignal index. We start by proving that the right-hand side of the claimed identity is indeed of dimensionality B. Let us define

$$\begin{aligned}j &:= \operatorname{MultiScaleIndex}_k(i) \text{ and}\\ \rho &:= \operatorname{MultiScaleSubsignal}^{(\vartheta,H)}_{(B,R,k)}(\xi,\ j)\end{aligned}$$

as abbreviations. Analogously to (b) where we deduced an expression for $2R$, marked with $(\Diamond)$, one obtains

$$\begin{aligned}&\dim_M(\rho)\\ \overset{\text{D. 27}}{=}\ &\left\lceil \tfrac{1}{k} \cdot \dim_M(\operatorname{Slide}_H(\operatorname{SubsignalPad}^{\vartheta}_{(B,R)}(\xi,\ j))) \right\rceil\\ \overset{\text{D. 5}}{=}\ &\left\lceil \tfrac{1}{k} \left(\dim_M(\operatorname{SubsignalPad}^{\vartheta}_{(B,R)}(\xi,\ j)) - h + 1\right) \right\rceil\\ \overset{\text{D. 26}}{=}\ &\left\lceil \tfrac{1}{k} \left(B + 2R - h + 1\right) \right\rceil\\ \overset{(\Diamond)}{=}\ &B - 1 + \left\lceil \tfrac{1+\operatorname{rem}(\tilde{R},\ 2)}{k} \right\rceil.\end{aligned}$$

As $1 + \operatorname{rem}(\tilde{R},\ 2) \in \{1,\ 2\}$ and $k \geq 2$ by requirement, we find $0 < \frac{1}{k}\big(1 + \operatorname{rem}(\tilde{R},\ 2)\big) \leq 1$ and hence $\dim_M(\rho) = B$.

Now let $\mu \in \{1, \dots, B\}$ for comparing both sides of the claim sample-wise. We have

$$\begin{aligned}\rho_\mu &\overset{\text{D. 27}}{=} \operatorname{Slide}_H(\operatorname{SubsignalPad}^{\vartheta}_{(B,R)}(\xi,\ j))_{k(\mu-1)+1}\\ &\overset{\text{D. 5}}{=} H\left(\sum_{\nu=1}^{h} \operatorname{SubsignalPad}^{\vartheta}_{(B,R)}(\xi,\ j)_{k(\mu-1)+\nu} \cdot e^h_\nu\right)\\ &\overset{\text{D. 26}}{=} H\left(\sum_{\nu=1}^{h} \vartheta(\xi,\ j + k(\mu-1) + \nu - R - 1) \cdot e^h_\nu\right).\end{aligned}$$

The corresponding sample of the left-hand side equals

$$\begin{aligned}&\operatorname{Subsignal}_B(\pi,\ \operatorname{div}(i-1,\ k)+1)_\mu\\ \overset{\text{D. 1}}{=}\ &\operatorname{Down}_k(\operatorname{Slide}_H(\operatorname{Pad}^{\vartheta}_R(\xi)))_{\operatorname{div}(i-1,\ k)+\mu}\\ \overset{\text{D. 27}}{=}\ &\operatorname{Slide}_H(\operatorname{Pad}^{\vartheta}_R(\xi))_{k\cdot\operatorname{div}(i-1,\ k)+k(\mu-1)+1}\\ \overset{\text{D. 5}}{=}\ &H\left(\sum_{\nu=1}^{h} \operatorname{Pad}^{\vartheta}_R(\xi)_{j+k(\mu-1)+\nu-1} \cdot e^h_\nu\right)\\ \overset{\text{D. 26}}{=}\ &H\left(\sum_{\nu=1}^{h} \vartheta(\xi,\ j + k(\mu-1) + \nu - R - 1) \cdot e^h_\nu\right),\end{aligned}$$

which is the same as ρ_μ, which proves the claimed identity. □

The ultimate goal is here to analyze functions applied to different scale levels of a signal and propose an efficient evaluation scheme. We have already taken the first step by analyzing the connection between downscaled subsignal extraction and subsignal extraction from a downscaled signal in Lemma 28. The complement of downscaling in this course of action is to repeat samples as many times as samples were omitted during downsampling. This leads to the following definition:

Definition 29. Let M be a set and $k \in \mathbb{N}_1$. Then

$$\begin{aligned}\operatorname{Up}_k\colon\ &\cup_1(M) \to \cup_k(M),\\ &\xi \mapsto \sum_{\mu=1}^{\dim_M(\xi)} \left(\xi_\mu \cdot \sum_{\lambda=1}^{k} e^{k\cdot\dim_M(\xi)}_{k(\mu-1)+\lambda}\right),\end{aligned}$$

is called the *upsampling operator with zero-order hold.*

We can immediately provide a statement on which samples go where during upsampling:

Lemma 30. Let M be a set, $q \in \mathbb{N}_1$ and $\xi \in M^q$. Then it is $\operatorname{Up}_k(\xi)_\nu = \xi_{\operatorname{div}(\nu-1,\ k)+1}$ for all $\nu \in \{1, \dots, kq\}$.

Proof. With Definition 29 there exists $\mu \in \{1, \dots, q\}$ with $\operatorname{Up}_k(\xi)_\nu = \xi_\mu$, where $\nu = k(\mu-1) + \lambda$ and $\lambda \in \{1, \dots, k\}$. One obtains

$$\begin{aligned}k \cdot (\mu-1) + (\lambda-1) + 1 &= (\nu-1) + 1\\ = k \cdot \operatorname{div}(\nu-1,\ k) &+ \operatorname{rem}(\nu-1,\ k) + 1.\end{aligned}$$

Here, $\lambda - 1 \in \{0, \dots, k-1\}$, hence uniqueness of Euclidean division implies $\mu - 1 = \operatorname{div}(\nu-1,\ k)$, and the claim follows. □

We are almost ready for the main result of this section, which states under which circumstances a function that accepts inputs in both the original scale and in a downscaled version can be evaluated efficiently.

Indexing rules are here as follows. Suppose P and Q are sets and $\chi \in \cup_1(P \times Q)$ is a signal with paired samples from $P \times Q$. Then there exists a dimensionality $d \in \mathbb{N}_1$ so that $\chi \in (P \times Q)^d$. Since $(P \times Q)^d \cong P^d \times Q^d$ we can also write χ as pair of signals, say $\chi = (\psi,\ \omega)$ where $\psi \in P^d$ and $\omega \in Q^d$. If $\nu \in \{1, \dots, d\}$ is an index, then we define $\chi_i := (\psi_i,\ \omega_i) \in P \times Q$ as an individual sample.

Theorem 31. Let M, N be sets and let $B \in \mathbb{N}_1$ be a fixed subsignal dimensionality. Let $f \colon M^B \times M^B \to N$ be a function that accepts signals in the original scale and in a downscaled version. Suppose f can be factorized into functions $g_{\text{Org}} \colon M^B \to P$, $g_{\text{Down}} \colon M^B \to Q$ and $g \colon P \times Q \to N$, so that

$$f(\rho,\ \tau) = g\left(g_{\text{Org}}(\rho),\ g_{\text{Down}}(\tau)\right) \text{ for all } \rho, \tau \in M^B,$$

where P and Q are sets.

As in Lemma 28, let $k \in \mathbb{N}$, $k \geq 2$, be a downsampling step size. Further, let $h \in \mathbb{N}_1$, $h \geq k$, and $H \colon M^h \to M$ a lowpass filter kernel and $\vartheta \colon \cup_1(M) \times \mathbb{Z} \to M$ a boundary-handling function. Let $\tilde{R} := (k-1)B + h - k \in \mathbb{N}$ and let $R := \lceil \tilde{R}/2 \rceil \in \mathbb{N}$ denote the required boundary size. Considering a signal $\xi \in \cup_B(M)$, we write $D := \dim_M(\xi)$, $\pi := \text{Down}_k(\text{Slide}_H(\text{Pad}_R^{\vartheta}(\xi))) \in \cup_1(M)$, and moreover

$$\begin{aligned} r := \ &\text{rem}(\tilde{R},\ 2) - \text{rem}(D - B + 1 + \text{rem}(\tilde{R},\ 2),\ k) \\ &+ \begin{cases} 0, & \text{if } k \text{ divides } D - B + 1 + \text{rem}(\tilde{R},\ 2), \\ k, & \text{otherwise.} \end{cases} \end{aligned}$$

Then $r \in \mathbb{N}$ and

$$\begin{aligned} &f\Big(\text{Subsignal}_B(\xi,\ i), \\ &\qquad \text{MultiScaleSubsignal}_{(B,R,k)}^{(\vartheta,H)}(\xi,\ \text{MultiScaleIndex}_k(i))\Big) \\ &= \text{Slide}_g\left(\text{Slide}_{g_{\text{Org}}}(\xi),\ \text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))\right)_i \end{aligned}$$

for all $i \in \{1, \dots, D - B + 1\}$, that is f applied to the subsignals of ξ and certain multi-scale subsignals of ξ equals the samples of g, g_{Org} and g_{Down} applied in a sliding fashion to signals derived from ξ. After g_{Down} has been applied to the downscaled signal π, the result has to be upsampled and the superfluous r trailing entries have to be trimmed away.

Proof. Through application of Lemma 28(b) we have that $\dim_M(\pi) = \left\lceil \frac{D-B+1+\text{rem}(\tilde{R},\ 2)}{k} \right\rceil + B - 1$. As $\dim_M(\pi) \geq B$, $\text{Slide}_{g_{\text{Down}}}(\pi)$ is well-defined and one obtains

$$\begin{aligned} \dim_Q(\text{Slide}_{g_{\text{Down}}}(\pi)) &\overset{\text{D. 5}}{=} \dim_M(\pi) - B + 1 \\ &= \left\lceil \frac{D-B+1+\text{rem}(\tilde{R},\ 2)}{k} \right\rceil. \end{aligned}$$

Therefore one obtains

$$\dim_Q(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi))) \overset{\text{D. 29}}{=} k \cdot \left\lceil \frac{D-B+1+\text{rem}(\tilde{R},\ 2)}{k} \right\rceil.$$

In the case of k dividing $D - B + 1 + \text{rem}(\tilde{R},\ 2)$ follows $\dim_Q(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi))) = D - B + 1 + \text{rem}(\tilde{R},\ 2)$ and by definition it is $r = \text{rem}(\tilde{R},\ 2) \in \mathbb{N}$, hence

$$\dim_Q(\text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))) = D - B + 1.$$

In the other case of k not dividing $D - B + 1 + \text{rem}(\tilde{R},\ 2)$ follows

$$\left\lceil \frac{D-B+1+\text{rem}(\tilde{R},\ 2)}{k} \right\rceil = \text{div}(D - B + 1 + \text{rem}(\tilde{R},\ 2),\ k) + 1.$$

From the definition of Euclidean division one yields

$$\begin{aligned} &\dim_Q(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi))) \\ &= D - B + 1 + \text{rem}(\tilde{R},\ 2) \\ &\quad - \text{rem}(D - B + 1 + \text{rem}(\tilde{R},\ 2),\ k) + k. \end{aligned}$$

We further have by definition

$$r = \text{rem}(\tilde{R},\ 2) - \text{rem}(D - B + 1 + \text{rem}(\tilde{R},\ 2),\ k) + k.$$

As $\text{rem}(D - B + 1 + \text{rem}(\tilde{R},\ 2),\ k) \in \{0, \dots, k-1\}$ follows $r \geq 1$. Thus $r \in \mathbb{N}$ and $\text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))$ is well-defined. We have that $\dim_Q(\text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))) = D - B + 1$ in this case as well.

We find that the number of samples in $\text{Slide}_{g_{\text{Org}}}(\xi)$ equals the number of samples in $\text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))$ in both cases. Since g works on individual samples we yield that

$$\text{Slide}_g\left(\text{Slide}_{g_{\text{Org}}}(\xi),\ \text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))\right)$$

consists of $D - B + 1$ samples from N.

Let $i \in \{1, \dots, D - B + 1\}$ be arbitrary and define

$$\tau := \text{MultiScaleSubsignal}_{(B,R,k)}^{(\vartheta,H)}(\xi,\ \text{MultiScaleIndex}_k(i))$$

as an abbreviation. From Definition 5 follows immediately that $\text{Slide}_{g_{\text{Org}}}(\xi)_i = g_{\text{Org}}(\text{Subsignal}_B(\xi,\ i))$. Considering the second argument to Slide_g, we have

$$\begin{aligned} &\text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))_i \\ =\ &\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi))_i \\ \overset{\text{L. 30}}{=}\ &\text{Slide}_{g_{\text{Down}}}(\pi)_{\text{div}(i-1,\ k)+1} \\ \overset{\text{D. 5}}{=}\ &g_{\text{Down}}(\text{Subsignal}_B(\pi,\ \text{div}(i-1,\ k)+1)) \\ \overset{\text{L. 28(d)}}{=}\ &g_{\text{Down}}(\tau), \end{aligned}$$

where we have used $i \leq D - B + 1$ in the first step so we could eliminate the trimming operator.

Combining these results and using the precondition (PC) that f can be factorized one finds

$$\begin{aligned} &\text{Slide}_g\left(\text{Slide}_{g_{\text{Org}}}(\xi),\ \text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))\right)_i \\ \overset{\text{D. 5}}{=}\ &g\big(\text{Slide}_{g_{\text{Org}}}(\xi)_i,\ \text{Trim}_r(\text{Up}_k(\text{Slide}_{g_{\text{Down}}}(\pi)))_i\big) \\ =\ &g\big(g_{\text{Org}}(\text{Subsignal}_B(\xi,\ i)),\ g_{\text{Down}}(\tau)\big) \\ \overset{\text{PC}}{=}\ &f\Big(\text{Subsignal}_B(\xi,\ i),\ \tau\Big), \end{aligned}$$

which equals the claimed expression. □

Theorem 31 directly provides an algorithm for efficient multi-scale analysis. The functional part operating on the original input signal should be applied in a sliding fashion. If this function can be cast as processing chain, which was discussed in Sect. IV, the there proposed theory can be used for efficient evaluation. The multi-scale subsignal index approximation proved in Lemma 28 facilitates application of the functional part operating on downscaled subsignals in a sliding fashion as well. Therefore, the theory from Sect. IV can be applied here also. We finally note that the generalization of the statements of Theorem 31 to functions that process arbitrarily many different downscaled signals is straightforward provided a proper factorization can be given.

Acknowledgment

This work was supported by Daimler AG, Germany.

References

[1] D. H. Hubel and T. N. Wiesel, "Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex," *Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[2] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network," in *Advances in Neural Information Processing Systems*, vol. 2, 1990, pp. 396–404.

[4] V. Jain and H. S. Seung, "Natural Image Denoising with Convolutional Networks," in *Advances in Neural Information Processing Systems*, vol. 21, 2009, pp. 769–776.

[5] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep Convolutional Neural Network for Image Deconvolution," in *Advances in Neural Information Processing Systems*, vol. 27, 2015, pp. 1790–1798.

[6] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[7] D. C. Cireşan, U. Meier, and J. Schmidhuber, "Multi-Column Deep Neural Networks for Image Classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2013, pp. 1097–1105.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[10] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano, "Toward Automatic Phenotyping of Developing Embryos from Videos," *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1360–1371, 2005.

[11] D. Grangier, L. Bottou, and R. Collobert, "Deep Convolutional Networks for Scene Parsing," in *International Conference on Machine Learning, Workshop on Learning Feature Hierarchies*, 2009.

[12] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[13] R. Vaillant, C. Monrocq, and Y. LeCun, "An Original Approach for the Localization of Objects in Images," in *Proceedings of the International Conference on Artificial Neural Networks*, 1993, pp. 26–30.

[14] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks," in *IEEE International Conference on Image Processing*, 2013, pp. 4034–4038.

[15] H. Li, R. Zhao, and X. Wang, "Highly Efficient Forward and Backward Propagation of Convolutional Neural Networks for Pixelwise Classification," Tech. Rep. arXiv:1412.4526v2, 2014.

[16] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," in *Proceedings of the International Conference on Learning Representations*. arXiv:1312.6229v4, 2014.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[19] H. Neudecker, "Some Theorems on Matrix Differentiation with Special Reference to Kronecker Matrix Products," *Journal of the American Statistical Association*, vol. 64, no. 327, pp. 953–963, 1969.

[20] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cuDNN: Efficient Primitives for Deep Learning," in *Advances in Neural Information Processing Systems, Deep Learning and Representation Learning Workshop*. arXiv:1410.0759, 2014.

[21] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proceedings of the International Conference on Learning Representations*. arXiv:1409.1556, 2015.

[22] T. D. Sanger, "Optimal Unsupervised Learning in Feedforward Neural Networks," Master's thesis, Massachusetts Institute of Technology, 1989.

[23] S. Taylor, *Optimizing Applications for Multi-Core Processors, Using the Intel Integrated Performance Primitives*, 2nd ed. Intel Press, 2007.

[24] L. Dagum and R. Menon, "OpenMP: An Industry-Standard API for Shared-Memory Programming," *IEEE Computational Science & Engineering*, vol. 5, no. 1, pp. 46–55, 1998.