**Internal distribution code:**

(A) [ - ] Publication in OJ
(B) [ - ] To Chairmen and Members
(C) [ - ] To Chairmen
(D) [ X ] No distribution

# Datasheet for the decision of 9 October 2023

**Case Number:** T 0523/21 - 3.5.06

**Application Number:** 17185902.8

**Publication Number:** 3282397

**IPC:** G06N3/04, G06N3/063, G06F17/15, G06K9/00, G06T11/60, G06T5/20

**Language of the proceedings:** EN

**Title of invention:**
ZERO COEFFICIENT SKIPPING CONVOLUTION NEURAL NETWORK ENGINE

**Applicant:**
Vivante Corporation

**Headword:**
Zero-skipping convolution/VIVANTE

**Relevant legal provisions:**
EPC Art. 56

**Keyword:**
Inventive step - (no)

**Decisions cited:**

**Catchword:**

Europäisches Patentamt
European Patent Office
Office européen des brevets

**Beschwerdekammern**

**Boards of Appeal**

**Chambres de recours**

Boards of Appeal of the
European Patent Office
Richard-Reitzner-Allee 8
85540 Haar
GERMANY
Tel. +49 (0)89 2399-0
Fax +49 (0)89 2399-4465

**Case Number: T 0523/21 - 3.5.06**

# D E C I S I O N
# of Technical Board of Appeal 3.5.06
# of 9 October 2023

| | |
|---|---|
| **Appellant:** (Applicant) | Vivante Corporation<br>2150 Gold Street, Suite 200<br>San Jose, CA 95002 (US) |
| **Representative:** | Betten & Resch<br>Patent- und Rechtsanwälte PartGmbB<br>Maximiliansplatz 14<br>80333 München (DE) |
| **Decision under appeal:** | **Decision of the Examining Division of the European Patent Office posted on 15 December 2020 refusing European patent application No. 17185902.8 pursuant to Article 97(2) EPC.** |

**Composition of the Board:**

| | |
|---|---|
| **Chairman** | M. Müller |
| **Members:** | T. Alecu |
| | B. Müller |

## Summary of Facts and Submissions

I. The appeal is against the decision of the Examining Division. The Appellant requests that the decision be set aside and that a patent be granted on the basis of the main request, or of one of three auxiliary requests, all filed with the grounds of appeal. The main request and the third auxiliary requests are "*except for one minor change in claim 1 to correct a clerical mistake*" identical to the main request, respectively the fourth auxiliary request, underlying the decision.

II. These two requests were refused for lack of inventive step. The decision cited inter alia:

D1: RAHMAN ATUL ET AL: "Efficient FPGA acceleration of Convolutional Neural Networks using logical-30 compute array",
D2: HAN SONG ET AL: "EIE: Efficient Inference Engine on Compressed Deep Neural Network",
D8: YU-HSIN CHEN ET AL: "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks".

III. This decision was announced in oral proceedings before the Board on 9 October 2023.

IV. Claim 1 of the main request defines:

*A method for performing convolution, the method comprising:*
*providing an array of coefficients C(kz, Z, kx, ky) defining a Z dimension, a kz dimension, a kx dimension, and a ky dimension;*

*providing a plurality of input images each corresponding to an index in the kz dimension;*
*(a) selecting, by an electronic device, a next tile position of a plurality of tile positions as the current tile position;*
*(b) selecting, by the electronic device, a next input image I(kzl) of the plurality of input images as the current image and an index kzl corresponding to the current image as a current kz index;*
*(c) loading, by the electronic device, a tile of the current image I(kzl) at the current tile position into a buffer as the current tile;*
*(d) individually and sequentially performing, by the electronic device, for each coefficient of at least a portion of coefficients C(kzl, Z, kx, ky) of the array of coefficients having an index in the kz dimension equal to kzl, wherein the at least the portion of the coefficients of the array of coefficients having the current kz index comprises only non-zero coefficients of the coefficients of the array of coefficients having the current kz index:*
*setting a ky shift amount of the current tile according to a ky index of the each coefficient in the ky dimension;*
*setting a kx shift amount of the current tile according to a kx index of the each coefficient in the kx dimension;*
*applying the ky shift amount to the current tile and applying the kx shift amount to the current tile in the buffer to obtain a shifted tile;*
*multiplying (314) the shifted tile by the each coefficient such that each row of the shifted tile is multiplied by the each coefficient in parallel by a series of multipliers to obtain a set of products;*

*adding (316) the set of products to set of accumulated sums stored in an accumulation buffer to obtain a set of updated sums; and*
*overwriting the set of accumulated sums with the set of updated sums;*
*(e) performing (b) through (d) until all input images of the plurality of images have been processed according to (b) through (d);*
*(f) outputting current values of the accumulated sums as an output image;*
*(g) performing (a) through (f) until all tile positions of the plurality of tile positions has been processed according to (a) through (f);*
*wherein the ky shift amount and kx shift amount are selected such that the accumulated sums are a three dimensional convolution of the current tile with the plurality of images upon completion of (f).*

V. Claim 1 of the first auxiliary request differs from that of the main request by specifying that "*the coefficients are obtained from one or more sparse kernels*".

VI. Claim 1 of the second auxiliary request differs from that of the main request in that the "*electronic device*" is replaced with a "*general-purpose processor*".

VII. Claim 1 of the third auxiliary request differs from that of the main request in that the "*electronic device*" is replaced with a "*computing device*" and by the addition of the following feature "*wherein the computing device includes a graphics-processing unit, GPU, and wherein steps (c) through (d) are performed on the GPU*".

## Reasons for the Decision

*The application*

1. The application relates to methods and systems for performing matrix convolution, for use in Convolutional Neural Networks (CNNs). According to the application in paragraph 3, this "*require*[s] *a very high computation and memory bandwidth*" and "*One way to reduce the requirement is to zero prune the coefficients and skip the computation when a coefficient is zero*". Known systems implementing that concept are based on matrix multiplication, which, requiring duplication of the 2D image data, "*increases the already very high memory bandwidth requirement*" (paragraph 17).

1.1 The application proposes "*an improved approach*" (paragraph 4). Paragraph 18 explains: "*Our solution combines zero coefficient skipping with a novel convolution engine. This dramatically lowers both computation and memory bandwidth. Traditional convolution is done by moving the whole kernel across the input 2D image to generate one result at a time. Our engine applies only one kernel coefficient to a big portion (tile) of the input 2D image each time. Since only one kernel coefficient is used at a time, multiplication by the zero coefficient is skipped to achieve much higher performance. If the kernel is compressed ahead of time, this further allows the use of a low-cost kernel decompressor which decompresses only one coefficient at a time.*"

2. The coefficient is thus applied to all image values that require multiplication by that coefficient before moving on to the next non-zero coefficient, and the results are accumulated at the corresponding output

positions. The method uses input and output tiling, the convolution being separately performed for each tile corresponding to one output tile (see paragraph 34).

2.1 In a 2D convolution (figure 3A) the computation loops (from outer to inner loop) over the stack of images, the 2D tile positions, the set of convolution kernels, and the (non-zero) coefficients of one kernel.

2.2 In a 3D convolution (figure 3B), the computation loops over the 3D output tiles positions, i.e. over the corresponding 2D input tile positions, the stack of images, the set of convolution kernels, and the (non-zero) coefficients of one kernel.

3. The claims of the requests on file relate to a loop arrangement with an outer loop on tile positions and an inner loop on the image stack, corresponding to the 3D convolution case.

*Main request*

4. The Examining Division found a lack of inventive step using D1 as a starting point.

4.1 D1 describes a method of accelerating deep neural networks on FPGAs. It explains in section II.C: "*Computationally a convolutional layer is a mere transformation of a 3D array into another 3D array using a series of 2D convolutions extended into the third dimension [...] computing one output point requires a 3-deep nested loop [...] The body of the loop nest is a simple MAC* [multiply and accumulate] *operation just like in matrix multiplication, and all the loop levels are permutable [...] A key question in mapping this loop nest is which*

*loops to parallelize* [in hardware] *vs. which ones to iterate* [execute sequentially]".

4.2 The solution of D1 (section II.D, figure 1) parallelizes the computation for each output value within a "*compute tile*", i.e. 3D output tiles, while iterating, in order, through images (z coordinate), tile offset positions (rr and cc coordinates), and kernel coefficients (x and y coordinates). The parallelization concerns the multiplications with the values in image data tiles and the accumulation of the results in the output tiles, which allows for data re-use (section III.B, figure 3). D1 optimizes the sizes of the data and output tiles, given the number of MAC hardware units (sections III.C and III.D, figure 4).

*Differences to D1; claim interpretation*

5. The Examining Division considered (decision, reasons 2.2 and 2.3) that there were two differences between claim 1 and D1, those being (paraphrased by the Board):

A) the loop ordering, the claim specifying, an outer loop on (2D) tile positions and an inner loop on images, whereas D1 teaches the opposite; and

B) individually and sequentially processing kernel coefficient values while skipping zero values.

6. The Board notes that D1 teaches in figure 1(a) an individual and sequential processing of the kernel coefficients, and a parallelization of all computations related to each one of the coefficients; this also concords with the Appellant's remark in the statement

of grounds of appeal at 3.12. So difference B) is only that the computations are performed with zero skipping.

7. The Appellant was of the opinion (statement of grounds of appeal, 3.13 to 3.17) that "*further differences exist*[ed] *related* [to] *the setting of a shift amount and applying a corresponding shift to the "current tile" along the kx and ky dimensions.*" This set of differences was labeled C).

7.1 The Appellant argued that these differences existed because in D1 the position at which the data was accessed in the inner nested loop did not depend on the value of the kernel coefficient, but "*follow*[ed] *a strict and ordinary iteration scheme of the inner nested loops in figure 1(a) of D1 ("HW UNROLL"). There* [wa]*s hence no shift or individual advancement of the iteration scheme, which one may call a "shift" based on observed values of a kernel coefficient or a position of a non-zero element*" (statement of grounds of appeal, 3.14, last sentence).

7.2 The Appellant further argued that because the claim stated that each row of the shifted tile was multiplied in parallel, and that the set of products was accumulated, this could not be found in the innermost loop of figure 1(a) of D1, but had to be searched in the outer loops (statement of grounds of appeal, 3.15).

8. The Board notes the following about the set of differences C).

8.1 The hardware unroll loop in figure 1 of D1 (the boxed area) is what is parallelized (see first paragraph of section II.D). So the input image portion (image tile) needed to compute the output tile, is brought into

position for parallel computation ("*shifted*") in accordance with the *x* and *y* positions defined by the corresponding outer loops for parallel multiplication and accumulation.

8.2 The Appellant's argument seems to be that in D1 there is no "*applying*" of a *kx* and *ky* (*x* and *y* in D1) shift, *depending on the kernel value* at that position, but that the shift results from a strict incremental approach while looping through the kernel in D1. The Board is not convinced that the claim wording is sufficient to justify the existence of this difference. Even if it did: if zero values are skipped in D1, which is what difference B) says, then the corresponding unroll loop will be skipped and the next shift will be to the *x* and *y* where the value is non-zero. Thus difference C) is a direct consequence of feature B).

8.3 There is therefore no need to consider separately the set of differences C).

9. During the oral proceedings before the Board the Appellant also argued that the feature "*for each coefficient of at least a portion of coefficients [..] of the array of coefficients [..] wherein the at least the portion of the coefficients of the array of coefficients [..] comprises only non-zero coefficients*" should be read to mean that only an array of the non-zero coefficients is used. The zero coefficients were not even transmitted to the parallel computational elements. This was a further difference to D1.

10. The Board does not believe that this reading, which corresponds to some form of compressed kernels, is implied by the claim wording. The claim only defines how "a *portion of coefficients*" of the full array are

used, and leaves completely open whether the full array is sent to the computational units and then the non-zero coefficients are used (thus a portion of the full array), or whether only the non-zero portion is sent. Thus, the claim covers the usage of both compressed and non-compressed kernels.

*Obviousness*

11. The Examining Division was of the opinion that the two differences A) and B) could be considered separately for the purposes of assessing obviousness (decision 2.4). It was of the opinion that difference A) was an obvious alternative design choice, depending on the expected reuse patterns, and that difference B) was obvious in view of D8, considering the objective technical problem of reducing energy consumption.

12. D8 proposes an architecture for energy-efficient data-flow for convolutional networks networks, and the Examining Division referred to section V.E of D8 where it is stated: "*The architecture can also exploit sparsity by (1) only performing data reads and MACs on non-zero values and (2) compressing the data to reduce data movement. Details on these techniques are described in [41]. This brings additional energy savings on top of the efficient dataflow presented in this paper.*"

13. The Appellant disagreed, stating that the loop ordering claimed was not motivated by data reuse, but (statement of grounds of appeal, 3.9) "*rather directed to reduce the number of total operations necessary to carry out the computation*" and that (statement of grounds of appeal, 3.10) "*the choice of "selecting a next tile position" before "selecting a next input image" in the loop nest is motivated and beneficial in the context of*

*the aspect of zero skipping: For a selected tile position in a given a kernel (outer loop), the same shift amount can be applied to the corresponding tile of all images (inner loop)*" so the amount of computation related to the shift computation was reduced. This also showed that the combination of the two features was not a mere juxtaposition (statement of grounds of appeal, 3.11).

14. Also, if only data reuse was considered, it would be beneficial to maintain the arrangement of D1 (statement of grounds of appeal, 3.8).

15. The Board remarks that the application makes no considerations as to the amount of the computations relating to the shift and loop ordering.

15.1 The Board is not convinced that the claimed loop ordering brings any advantages in terms of computational effort. Whether moving from 2D tile to 2D tile in the same image or to a following image in the stack, a new memory address needs to computed. Neither the claim nor the description specifies any form of address caching from image to image, so as to apply the same shift. That may not be appropriate anyway, given that the claimed convolutional kernel is 3D (even if for a current image (kz) a kernel value at kz, kx and ky coordinates is zero, the value at kz+1, but same kx and ky coordinates, need not be zero).

15.2 So the Board does not see that this difference contributes to solving any technical problem, other than, perhaps, providing an alternative loop arrangement for computing convolutions. For the same reasons, the Board sees no interaction with the context of zero-skipping.

15.3 The Board is of the opinion that the skilled person would consider such an alternative based at least on the knowledge present in D1 that all loops are permutable (see quotation in 4.1 above), and finds feature A) to be obvious in the context of D1.

16. Regarding difference B), the Appellant argued first (statement of grounds of appeal, 3.22) that the implementation of zero-skipping in D1 would lead to major modifications that the skilled person would not implement in D1. More precisely: "*The inner parts for the loop structure shown in D1 in figure 1(a) are indicated as "HW UNROLL", which means that their structure has direct correspondence to structures in hardware of D1, there shown in figures 2, and 3. For example, the implementation of features of difference (C) related to applying a shift amount would lead to significant design changes in the way in which input data is loaded an made available to the 3D array of MAC units shown in Fig. 3, which the skilled person would not undertake during a routine design activity.*"

16.1 Considering D8, the Appellant argued that "*Aspect (2) does not provide a hint to any of the differences (A) to (C), since compressing is not specified by any of those differences. Aspect (1) suggests avoiding data reads and MACs at an abstract level but does not provide any hint how the skilled person could implement these suggestions in the context of D1.*"

17. The Board remarks that the parallelization scheme in section II.D of D1 is a conceptual one, and applicable to various architectures. The algorithm of figure 1(a) does not fix, or imply, the architecture of figure 3. At this conceptual level, the implementation of zero skipping is trivial, and only requires a conditional

statement before the unroll box. Even if the statement in D8 is abstract, it is sufficient to render zero-skipping obvious; the Board is further of the opinion that, for efficiency purposes, skipping multiplications or additions with zero is a matter of common general knowledge (see also D2 as discussed below).

17.1 The Board has otherwise no doubts that the skilled person is able to implement this conditional statement in the FPGA-based architecture of D1.

17.2 The Board finds therefore feature B) to be obvious in the context of D1.

18. Even if considering the claim to imply the use of compressed kernels (see 10 above), the Board is of the opinion that feature B) would still be obvious.

18.1 The Appellant argued that under this reading a conditional statement before the unroll loop of D1 would lead to an implementation using non-compressed kernels.

18.2 That is true, but the Board notes that the use of compressed kernels is also known in the art (for reducing data transfers) and therefore obvious to be used in D1 as well. D8 as quoted above (point 12) already points to that (the second aspect); further, D2 teaches zero-skipping using compressed kernels (see page 245, left column, bottom paragraph and the compressed representation on the same page, right column, section III.B). The Board notes that the context of D2 is the same as that of D1 and D8, i.e. efficient implementation of neural networks (see title).

19. In conclusion, the Board agrees with the Examining Division that the claimed subject matter of claim 1 is obvious to the skilled person starting from D1.

*Auxiliary requests*

20. Claim 1 of the first auxiliary request defines that the method is applied to sparse kernels. The Appellant argued that this feature emphasised the advantages of the method.

20.1 The Board is of the opinion that the skilled person would apply the method of D1 to all types of kernels, sparse or non-sparse. Hence the conclusion as to lack of inventive step in respect of claim 1 of the main request remains valid.

21. The second and third auxiliary requests define that the method is executed on a "*general purpose*" processor, respectively a *graphics processing unit* (GPU).

21.1 The Appellant argued that the focus of D1 was an efficient implementation for FPGAs only and therefore could not be the closest prior art for assessing obviousness of these requests, or, alternatively, that the skilled person could not arrive at the invention starting from D1. There was no hint in D1 for the skilled person to implement the method of D1 on either of the two architectures. In particular, GPUs allowed for efficient parallel processing, so a technical effect was related to the selection of this architecture.

22. The Board disagrees. In principle, any document may be considered as a starting point for the skilled person. The relevant question is not whether a certain document

may be considered as "closest prior art" in an obviousness analysis, but what the skilled person would do starting from that document.

22.1 While it is correct that the focus of D1 is a FPGA implementation, it is also clear to the skilled person reading D1 that the conceptual C code of D1, figure 1a, can be implemented on any platform supporting parallel processing, and the skilled person would consider implementation on other well known architectures suitable to that purpose. This includes a general-purpose processor or a GPU, as the Examining Division also stated (decision, 9.13).

22.2 Thus, the conclusion as to lack of inventive step remains valid for claims 1 of these requests as well.

## Order

## For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

L. Stridde Martin Müller

Decision electronically authenticated