# Automatic CNN Model Partitioning for GPU/FPGA-based Embedded Heterogeneous Accelerators using Geometric Programming

Walther Carballo-Hernández, Maxime Pelcat, Maxime Pelcat, François Berry

Journal of Signal Processing Systems
https://doi.org/10.1007/s11265-023-01898-0

# Automatic CNN Model Partitioning for GPU/FPGA-based Embedded Heterogeneous Accelerators using Geometric Programming

**Walther Carballo-Hernández**[1] · **Maxime Pelcat**[1,2] · **François Berry**[1]

Received: 6 December 2022 / Revised: 3 September 2023 / Accepted: 3 October 2023


**Abstract**
Graphics Processing Unit (GPU), dedicated Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA) accelerators are currently platforms of choice for porting Convolutional Neural Networks (CNNs). In this work, an automated Central Processing Unit (CPU)-GPU-FPGA partitioning selection is proposed for a given CNN layer. It is shown that using a Generalized Geometric Programming (GGP) optimization problem formulation, the CPU-GPU-FPGA partitioning problem can be modeled by considering a set of system performance metrics and constraints. Each metric is expressed in a posynomial form depending on CNN hyperparameters and architecture resource models. As for the partitioning method, the state-of-the-art techniques covered are: tiling, grouped convolution and fused-layer. The proposed analytical formalization is then employed to derive a set of objective functions and constraints as a GGP problem. It is demonstrated that it is possible to relax some problem constraints by including a penalization term, and reduce the problem to multiple simpler Geometric Programming (GP) sub-problems. Experimental results targeting an embedded FPGA-GPU platform with CNN layer configurations from state-of-the-art CNN models (AlexNet, VGG16 and ResNet18) show that the simplified problem is solvable in polynomial time with a speed-up gain and energy reduction of around 20% and 15%, respectively, when compared against an arbitrary balanced partitioning. If the models for objective and constraints functions preserve the posynomial form and log-log convexity, it is demonstrated that GGP is an efficient optimization solution to the Design Space Exploration (DSE) problem.

**Keywords** Heterogeneous platform · Convolutional Neural Network · Geometric programming · Mathematical optimization · Embedded design

## 1 Introduction

CNN processing at the edge opens new challenges by requiring embedded systems to support strong computational workloads. This rapid evolution fosters more complex hardware architectures comprising interconnected heterogeneous elements. GPU, dedicated ASIC and FPGA accelerators are currently platforms of choice for porting CNNs, as their programmability and internal parallelism fit well the concurrency and the customization needs of modern CNNs. The hardware/software co-design intricacy increases when logic and memory constraints are taken into consideration concurrently in system DSE with the objective to optimize energy consumption and throughput. CNN inference deployment at the edge with high performance per Watt requires careful co-design of hardware architecture and algorithm. Systems are currently becoming more complex on both sides: architecturally and algorithmically. Embedded platforms include several asymmetrical processing elements and different levels of memory hierarchies. Thus, design solution selection from DSE requires optimization techniques to choose an appropriate partition considering available resources and performance goals. To facilitate DSE on the edge with heterogeneous platforms, an optimization problem formulation is formalized. The resulting objective and constraint

✉ Walther Carballo-Hernández
walcarher@gmail.com

Maxime Pelcat
mpelcat@insa-rennes.fr

François Berry
francois.berry@uca.fr

[1] Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Institut Pascal, Clermont-Ferrand F-63000, France

[2] IETR, UMR CNRS 6164, UMR CNRS 6602, Univ Rennes, INSA Rennes, Rennes 43017-6221, France

Published online: 02 November 2023

functions have the form of a GGP problem. This family of optimization problems are mostly non-convex, and thus do not have a unique optimal point. However, in some cases they can be reduced and solved as GP problems. Since GP problems are fast solvable, in polynomial time, and ensure a global optimal solution, it is strongly desired to transform a GGP into a GP problem when possible. Nevertheless, this is not a trivial task, as it requires a deep expertise on the nature of the problem at hand. In this work, the following contribution is presented:

*Given a system performance objective and constraints, it is shown that a partitioning method of a CNN layer over a set of heterogeneous processing elements can be analytically described, expressing objective and constraints in a posynomial form for individual processing elements. A GGP optimization problem is formulated and a demonstration that the GGP equality constraints can be relaxed so as to solve the problem in the form of a set of simpler GP problems. This relaxation preserves the properties of GP problems, such as the existence of a global optimal solution and the polynomial solving time.*

The paper is organized as follows: In Section 2, an analysis of related state-of-the-art works is presented, focusing on system modeling for Deep Learning (DL), as well as on partitioning, scheduling and optimization techniques. Section 3 explains the context of measurement-based system performance modeling of CNN inference and introduces the concepts of monomial and posynomial. In Section 4, the theory behind GGP and GP is presented and explained, how a GGP problem can be relaxed to a GP problem using a penalization technique based on the condensation solution. In Section 5, GP optimization is applied to common CNN layer configurations and find the optimal partitions. Finally, in Section 6, it is discussed the results and observations.

## 2 Related Works

Since the early years of DL-oriented embedded hardware platforms, research has dedicated large efforts on partitioning machine learning efficiently over several edge devices [1]. The partitioning solutions must take into consideration the hardware profiling, partitioning, scheduling and deployment. Fused-layer is a popular technique permitting two or several layers to be mapped on a same device, reducing inter-device communication. Similarly, a set of containers, such as Docker containers, can be instantiated to a model and treat partitions as cloud services. In [1], a layer-wise containerization of a Deep Neural Networks (DNN) with fused-layer is proposed by using analytical regression models of different DNN configurations and optimizing analytically with dynamic programming. In [2], a DNN is partitioned at a finer granularity, mapping individual neurons to different Internet of Things (IoT) devices. However, the optimization of [2] is focused on reducing inter-device communication using a similar heuristic as Kernighan-Lin heuristic [3], swapping partition nodes in a graph abstraction. This solution is tailored to very constrained resources where communication is a dominating bottleneck. In [4], the previous use case is extended to heterogeneous platforms with different devices including CPUs, FPGAs and GPUs. Therefore, communication channels with several latencies and throughputs are considered in the optimization problem. Vanishree et al., create a Roofline analytical model to choose the appropriate batch partitioning ratio of each device the platform. In [5], data redundancy is exploited on fused-layers for contiguous partitions with the objective to reduce inter-systems communication overhead. For this purpose, the optimization process decides when to allow or when to avoid layer fusing. In the same publication [5], a discussion on partition size and communication overhead is covered. Extending the work of [5], Stahl et al., demonstrate that layer-wise partitioning can be found using Integer Linear Programming (ILP) optimization problems considering resource constraints and minimizing communication [6]. Finally, in [7], an assisting tool solution is introduced for embedded hardware characterization using computation and communication knowledge from heterogeneous platforms. The estimation precision is increased by introducing detailed information of the system for different CNN operations. Then, the scheduler uses a greedy layer-wise mapping as optimization strategy, selecting the most performing device iteratively for each layer. While this hardware-awareness is usually known to the designer, many internal parameters are difficult to acquire in practise or may be hidden to the designer. This solution from [7], however, does not require a performance-based measurement database generation, which in many cases may save some development time.

Same authors in [8] add the consideration of weight-dominated CNN layers for layer fuse. Their objective functions seeks an even weight distribution on several edge devices. However, in heterogeneous systems, this may not be a desired property, since some elements are more efficient with memory access handling or embed more memory.

With respect to this state-of-the-art, the proposed method is less specific to a given deep learning solution. A resource and objective formulation are proposed for the fast optimization of multi-system CNN partitioning that combines resource constraints, performance constraints and performance objectives. The embarrassingly parallel nature of CNNs is exploited to simplify the problem formulation. The proposition is intended to be widely applicable and adaptable to a large set of CNN partitioning problems.

## 3 Monomial and Posynomial Model Formulation

The convolution is the most common operation in convolutional layers on CNN models, therefore it is also the most time consuming and most parameter intensive workload [9]. Many efforts on state-of-the-art works focus on obtaining an analytical behavioural model for this operation on given devices, modeled together with a ReLU activation function. With the selection of the representative structural features, the computation and communication workloads are described. Afterwards, a dataset was created by stochastically exciting the heterogeneous system. Finally, a set of performance models were derived with modeling techniques from these data points. The obtained models are mostly represented in a monomial or posynomial form. It is discussed why this specific form is required to be solvable with GGP in Section 4. In this paper, two metrics or Key Performance Indicators (KPIs) are considered: processing latency (*LAT*) and processing energy (*E*). As many other physical models in electronic devices, these can represent a system in posynomial form. The platform from Fig. 1 presents the experimental setup. A custom embedded heterogeneous platform is employed incorporating an Nvidia® Jetson TX2® embedded CPU-GPU (green), and an Intel® Cyclone 10 GX FPGA (blue) using a dataflow Direct Hardware Mapping (DHM) technique [10]. Interconnection is established by a Peripheral Component Interconnect Express (PCIe) Gen2 x4 (5 GT/s) communication channel (gray). The communication link between CPU-GPU lie on the same System-on-Chip (SoC) die, therefore they share a common external memory. DHM technique offers an energy-efficient but resource-hungry FPGA implementation. The results can however be extended to larger architectures and more constraints, with limited effort.

GP is a an optimization technique that is useful to solve large scale problems by formalizing them into not-too-restrictive mathematical models. The system performance models must comply with GP specific analytical formulation based on two forms of expressions. The first form to consider is the *monomial*. A monomial has the form presented in Eq. 1:

$$u(\boldsymbol{X}) = c \prod_{i=1}^{n} x_i^{a_i} \tag{1}$$

Where the function $u : \mathbb{R}^n \rightarrow \mathbb{R}$ maps an input feature vector $\boldsymbol{X}$ (such as $\boldsymbol{X}$ in previous section) to a real value. $c$ is strictly positive $c > 0$, $a_i \in \mathbb{R}$ and the domain is also strictly positive $\mathcal{D}(u) = \mathbb{R}^n_{++}$, or explained differently, the input feature vector $\boldsymbol{X}$ must be fully composed of non-zero positive real values. As a second condition, a *posynomial* is a linear combination of monomials as shown in Eq. 2:

$$v(\boldsymbol{X}) = \sum_{k=1}^{K} c_k u(\boldsymbol{X}) = \sum_{k=1}^{K} c_k \prod_{i=1}^{n} x_{ik}^{a_{ik}} \tag{2}$$

Where, similarly to Eq. 1, each element $c_k$ is strictly positive, $c_k > 0$. To fully exploit the posynomials in the GP context, the performance system models for latency (*LAT*) and energy (*E*) must follow these rules. These performance metrics, when evaluated for CNNs, tend to fit well the GP theory. Indeed, as CNNs layers heavily parallelize, their cost in terms of energy and time tend to be proportional to the product of their structural dimensions, leading to monomial formulations. Moreover, the cost of a complete algorithm will tend to be the sum of individual layers costs, leading to posynomial formulations. These intuitions motivate the study of GP for CNN partitioning optimization.

## 4 Optimization Problem Formulation

After selecting analytical modeling and partitioning technique, the methodology further proceeds with the formal definition of the problem formulation of a CNN layer. The solution of the optimization problem heavily depends on the nature of the objective function and constraint choice. Mathematical properties, such as, curvature

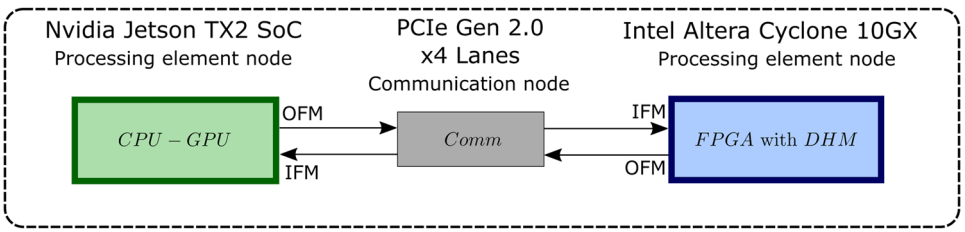



**Figure 1** Architecture model as for the heterogeneous setup with three computing elements: A Nvidia® Jetson TX2® CPU-GPU (green) SoC and an Intel® Cyclone10GX FPGA (blue) interconnected through x4 lanes of a PCIe Gen2 link (gray). Each processing element and communication node has an Input Feature Map (IFM) and a Output Feature Map (OFM) associated to it required for metric performance measuring on each device.

and monotonicity have a critical role on the analytical or numerical tools to be exploited. Furthermore, expertise and a-priori knowledge on the problem is key to mathematically manipulate the problem and transform it from a non-convex to a convex function [11].

In this section, it is described how the posynomial models of Section 3 are used to formulate the optimization problem as a set of GP problems. Nevertheless, because GP does not allow monomial expressions as equality constraints, this violates GP problem formulation. Therefore, a *relaxation* technique based on objective function *penalization* is proposed. The added penalization term is obtained by the transformation of the equality constraints to posynomials, this technique is known as *condensation* [12]. This reformulation provides a quick numerical solution in tenths of iterations, being each iteration solvable in polynomial time with interior-point methods using *CVXPY 1.0* library [13]. Interior-point methods approximate a numerical solution by implicitly adding the inequality constraints to the objective function. Some mathematical transformations are applied to convert the problem to a convex and differentiable one and solvable by Newton's method. The solution converges to a set of solutions towards the optimal solution per each iteration, this trajectory is known as the *central path* [11].

### 4.1 GGP Formulation of the Heterogeneous CNN Layer Partitioning

GP problems are used in multiple domains for their great versatility. Because usually GP are non-convex but easily transformed into convex problems, they have gained interest in optimization formulation, since GP convex problems have been proven to be fast solvable [11]. The fact that the GP solution is analytically found makes it highly desirable but the formulation of the problem can be difficult. The strict mathematical conditions on the objective and constraint functions require specific analytical forms. A GP problem has the form of Eq. 3:

$$\begin{aligned} &\text{minimize } f_0(\boldsymbol{X}) \\ &\text{subject to } f_i(\boldsymbol{X}) \leq 1;\ i = 1, 2, \cdots, p \\ &\qquad\quad h_i(\boldsymbol{X}) = 1;\ i = 1, 2, \cdots, m \end{aligned} \tag{3}$$

Where the objective function $f_0(\boldsymbol{X})$ and the inequality constraint functions $f_i(\boldsymbol{X})$ shall be posynomials (Eq. 2 from Section 3), and the equality constraints $h_i(\boldsymbol{X})$ shall be monomials (Eq. 1 from Section 3) [11]. If the problem is non-convex, it is possible to perform mathematical transformations to convert it to the convex form [11].

The GGP formulation of the GPU-FPGA partitioning of a CNN layer can be formalized below in Eq. 4:

$$\begin{aligned} &\text{minimize } LAT_{Het}(\boldsymbol{X}) \\ &\text{subject to } ALM_F(\boldsymbol{X_F}) \leq ALM_{max} \\ &\qquad\quad ALUT_F(\boldsymbol{X_F}) \leq ALUT_{max} \\ &\qquad\quad LAB_F(\boldsymbol{X_F}) \leq LAB_{max} \\ &\qquad\quad M20K_F(\boldsymbol{X_F}) \leq M20K_{max} \\ &\qquad\quad \boldsymbol{X_F} + \boldsymbol{X_G} = \boldsymbol{X} \end{aligned} \tag{4}$$

Where $\boldsymbol{X}$ is the input feature vector containing the structural features of a CNN layer. These are $H_I$ (height of the IFM), $W_I$ (width of the IFM), $C$ (channels of the IFM), $k$ (size of kernel filter) and $N$ (number of kernel filters). $\boldsymbol{X_F}$ and $\boldsymbol{X_G}$ are the structural input feature vectors and sub sets of $\boldsymbol{X}$ to be allocated on the FPGA and GPU, respectively. $LAT_{Het}(\boldsymbol{X})$ is the posynomial model of the execution latency. $ALM_F(\boldsymbol{X_F})$, $ALUT_F(\boldsymbol{X_F})$, $LAB_F(\boldsymbol{X_F})$ and $M20K_F(\boldsymbol{X_F})$ are posynomial resource model inequalities for the target FPGA, respecting the maximum logic and memory element count of the device. In the experiments (Section 5), the Intel® FPGA resources were selected: Adaptive Logic Modules (ALMs), Adaptive Look-Up Tables (ALUTs), Logic Array Blocks (LABs) and M20K memories.

Not only does the objective function directly depend on the taken partitioning strategy, but also the constraints last equality. With the purpose of forcing the algorithm to converge to non-trivial solutions (e.g. executing nothing), the equality constraint $\boldsymbol{X_F} + \boldsymbol{X_G} = \boldsymbol{X}$ is added. As an example, for the grouped convolution, the number of channels on the FPGA ($C_F$) and the number of channels on the GPU ($C_G$) must match the total number of channels before splitting, while keeping other features constant, or $\boldsymbol{C_F} + \boldsymbol{C_G} = \boldsymbol{C}$. Unfortunately, this equality constraint is a sum of monomials, therefore a posynomial. For GP, this definition violates the monomial equality constraint from Eq. 3. A GP is a particular case of GGP, which accepts posynomials as equality constraints, but the GGP problem is no longer convex nor a method to find an optimal solution is known. In this case, multiple heuristics have been proposed to convert a GGP to a GP using penalization terms [14]. In next section, a condensation solution to relax the posynomial equality constraints to the given use-case is proposed.

### 4.2 GGP Penalization by Equality Constraints Condensation

As discussed in previous section, Eq. 4 includes equality constraints that violate the conditions of GP. Nevertheless, these expressions can be transformed from posynomials to single set of monomials of the following form using the condensation technique proposed in [12]. These new expressions $\bar{h}$ are computed following the function 5:

$$\bar{h}(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \prod_{t=1}^{T} \left[ \frac{u_t(\boldsymbol{X})}{\epsilon_t(\hat{\boldsymbol{X}})} \right]^{\epsilon_t(\hat{\boldsymbol{X}})} \tag{5}$$

Where $\hat{\boldsymbol{X}}$ is a feasible solution starting point. For example, an *a-priori* solution $\hat{\boldsymbol{X}}$ can be a naive partitioning that respects the problem constraints. $u_t, t \in T$ are each of the $T$ monomial terms (Eq. 1) in all posynomial equality constraints $h_i$ (in the form $v$ of Eq. 2). Finally, the exponent $\epsilon_t(\hat{\boldsymbol{X}}) = \frac{u_t(\hat{\boldsymbol{X}})}{h_i(\hat{\boldsymbol{X}})}$ is the result of the *arithmetic–geometric inequality* approximation, transforming any posynomial to a monomial. Since $\epsilon_t$ is a ratio, then the basic property $\bar{h}(\boldsymbol{X}, \hat{\boldsymbol{X}}) \leq h(\boldsymbol{X})$ of the arithmetic–geometric inequality applies. $\bar{h}$ is known as a posynomial condensed to a monomial. Notice that, because $\bar{h}$ has all the properties of a monomial, these constraints cab be relaxed by including a penalization term in the objective function, while keeping the posynomial conditions and changing the equality to inequality constraints:

$$\begin{aligned} &\text{minimize } f_0(\boldsymbol{X}) + \sum_{k=1}^{m} \alpha_k \bar{h}_k(\boldsymbol{X}, \hat{\boldsymbol{X}})^{-1} \\ &\text{subject to } f_i(\boldsymbol{X}) \leq 1;\ i = 1, 2, \cdots, p \\ &\qquad\qquad h_i(\boldsymbol{X}) \leq 1;\ i = 1, 2, \cdots, m \end{aligned} \tag{6}$$

Where each $\alpha_k$ is the penalization weight of the original objective function. Note that these weights add parameters to be tuned in the optimization process. Now $f_i$ and $h_i$ constraint functions are both posynomial inequalities, as a result of the relaxation by condensation. Although now the problem is GP-compliant, a new problem arises. The penalization weights $\alpha_k$ must be carefully selected for each equality constraint. For this purpose, several heuristics exist [14]. In simple cases, $\alpha_k$ selection can be solved numerically by incrementally changing the weight values, this process is known as *tightening* [12]. When the tightened condensed constraints approximate the desired value, the solution is accepted, and that value for $\alpha_k$ is selected. This technique is similar to other *primal-dual optimization problems (duality)*, like Lagrangian-based optimization techniques [11]. Where, in order to solve the primal formulation of a possibly non-convex problem without any modification, the dual problem must be first formulated. These dual optimization problems incorporate, after some manipulation or transformation, the constraints into the objective function as penalization. For dual problem formulation, the posynomial condensation into a monomial from the penalization is based on the result of an approximation using the algebraic manipulation of the arithmetic–geometric inequality. Solving the dual problem is more tractable than solving the primal problem [15].

In the following Section 5, we empirically demonstrate that it is possible to find the penalization parameters within a few iterations for different CNN layer configurations. Since each iteration is a GP problem, it is solvable in polynomial time. Thus, the final solution remains upper-bounded by the polynomial time complexity for the selected partitioning techniques.

## 5 Experimental Results

In this section, the results of the optimization problem definition of Section 4 with different CNN layer configurations are presented. The first results presented here consider a platform with only an embedded FPGA with a GPU, afterwards the objective function is easily modified to include more compute elements with different communication links between a CPU-GPU-FPGA platform. This is also the case for the constraints functions, which can be extended to include other embedded devices constraints.

### 5.1 Single Layer Optimization

Exploiting the heterogeneity of an embedded platform with a single FPGA-GPU coupling and a single communication link, an optimization problem is formulated in the form of Eq. 7 with the models of Section 3 and the objective function. Figure 2 depicts the setup test for allocation of the obtained partitions on a single CNN layer. This setup considers a sequential execution of the FPGA and GPU workloads to minimize the latency of an individual CNN layer. That is, the objective function is the sum of processing times on each device, considering the transfer time of the intermediate Feature Maps (FMs)

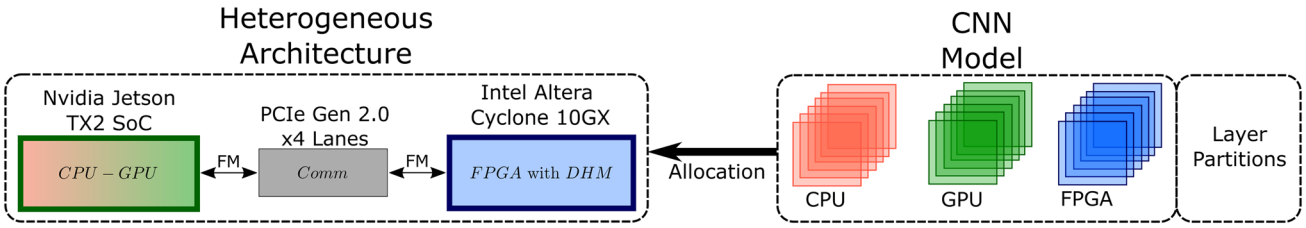



**Figure 2** **Setup 1:** Single layer setup of a single CNN layer allocation.

from one node to the other ($LAT_{Comm}$). Since the host shares the same memory between the CPU and GPU, but not for the FPGA; the FM with the structural features $\boldsymbol{X_F}$ must be transferred to the FPGA accelerator. Therefore, the communication latency depends on the shape of this partitioned IFM, or $LAT_{Comm}(\boldsymbol{X_F})$.

In Eq. 7, the selected partitioning technique is the grouped convolution. Therefore, the equality constraint only considers the channels on each device ($C_F$ and $C_G$ ) to match the total number of channels on the layer ($C$).

$$\begin{aligned} \text{minimize} \quad & LAT_F(\boldsymbol{X_F}) + LAT_G(\boldsymbol{X_G}) + LAT_{Comm}(\boldsymbol{X_F}) \\ \text{subject to} \quad & ALM_F(\boldsymbol{X_F}) \leq ALM_{max} \\ & ALUT_F(\boldsymbol{X_F}) \leq ALUT_{max} \\ & LAB_F(\boldsymbol{X_F}) \leq LAB_{max} \\ & M20K_F(\boldsymbol{X_F}) \leq M20K_{max} \\ & C_F + C_G = C \end{aligned} \tag{7}$$

As discussed in Section 4, the posynomial equality constraint violates the requirement for a GP solution. However, the constraint can be relaxed to an inequality by adding the penalization term based on the condensed posynomial. By incorporating this penalization term, the latency objective function increases smoothly when the grouped convolution constraints are not met. This is, when the solution chosen from the optimization problem does not compute the number of channels in the convolution layer. If the equality constraints are relaxed to inequality constraints, some solutions that were not originally feasible are now possible, and even optimal. In the case of the grouped convolution, if a constraint *equality* is relaxed to a *less or equal*, the obvious and most trivial solution to minimize the latency, would simply be processing the fewest number of IFM channels as possible. Namely, processing one single channel on the FPGA ($C_F = 1$) and one on the GPU ($C_G = 1$). Evidently, this is not a desired solution, since the IFM tensor is not fully processed. First, for this purpose, the condensed penalization term $\bar{h}(C_F, C_G, \hat{C_F}, \hat{C_G})$ must be obtained. Then, the equality constraint on the number of channels is removed. Using Eq. 5, with $T = 2$, since the constraint consists of two monomials, the posynomial is condensed to a monomial in the following Eq. 8:

$$\bar{h}(C_F, C_G, \hat{C_F}, \hat{C_G}) = \left[ \frac{\frac{C_F}{C}}{\frac{\hat{C_F}}{\hat{C_F}+\hat{C_G}}} \right]^{\frac{\hat{C_F}}{\hat{C_F}+\hat{C_G}}} \left[ \frac{\frac{C_G}{C}}{\frac{\hat{C_G}}{\hat{C_F}+\hat{C_G}}} \right]^{\frac{\hat{C_G}}{\hat{C_F}+\hat{C_G}}} \tag{8}$$

This penalization term is a monomial that includes the channel parameters of each device. Thus by adding it to the objective function, it preserves its posynomial form. To incorporate this term in the objective function, it is only necessary to add the penalization with a penalization weight, $\alpha\bar{h}$. As for an instance, $LAT_F(\boldsymbol{X_F}) + LAT_G(\boldsymbol{X_G}) + LAT_{Comm}(\boldsymbol{X_F}) + \alpha\bar{h}(C_F, C_G, \hat{C_F}, \hat{C_G})$. For one equality constraint condensation, only one penalization weight $\alpha$ is required ($m = 1$ for Eq. 6). Consequently, it is feasible to sweep over different values of $\alpha$ and tighten the relaxed inequality constraint, until $\frac{C_F+C_G}{C}$ approximates the unity.

Figure 3, shows an example with an input tensor sequential processing with grouped convolution partitioning of configuration size of $H_I = W_I = 112$, $C_I = 16$, $k = 1$ and $N = 32$ for latency minimization. For a grouped convolution partitioning, all features equal for each device are kept, except for the number of channels. Resource constraints are taken into account. Each bar represents a partitioning iteration chosen by the solution of the GP solver with a given $\alpha$ value. The blue bars are the channels mapped on the FPGA, while the green bars are mapped to the GPU. By increasing the value of $\alpha$, the constraint (red line) is tightened at each iteration, penalizing the objective function until the normalized relaxed constraint function approximates 1. For this instance, $\alpha = 170$, represented by the orange dashed line, is the first value to satisfy the constraint with $C_F = 14$ and $C_G = 2$. Additionally, for this test a balanced feasible solution is chosen for the constraint condensation. This is, the number of channels is equally distributed for both processing devices ($\hat{C_F} = \hat{C_G} = 8$). An important observation from Fig. 3 is that multiple solutions fulfill the tightened equality constraint. Therefore, there are infinite feasible solutions after iteration 17. However, as shown in Fig. 4, since the latency (solid purple line) has a non-decreasing monotonous nature, the following solutions perform worse than the first accepted iteration (dashed orange line). The solutions found on each iteration are found in polynomial time with respect of the inputs $\boldsymbol{X}$, $\boldsymbol{X_F}$ and $\boldsymbol{X_G}$. As a consequence, the solution of each GP problem remains bounded by polynomial time and, fixing a step size on $\alpha$, so is the iterative solution.

The GGP objective function can be easily modified to incorporate several computing devices with their respective communication bus linked to the other devices. Equation 9 shows an example of an objective function reformulation with the CPU latency model inclusion ($LAT_C(\boldsymbol{X_C})$). Where $X_C$ are the features of the CNN layer deployed on the CPU. Also communication links $LAT^{C-G}_{Comm}(\boldsymbol{X_{C-G}})$ and $LAT^{C-F}_{Comm}(\boldsymbol{X_{C-F}})$ are incorporated to consider the latency of the communication overhead of inter-device FM transfers between CPU and both the other devices. Additionally, the condensation and penalization term ($\bar{h}$) in the objective function must also include the features of the modified equality constrain ($C_C + C_F + C_G = C$).

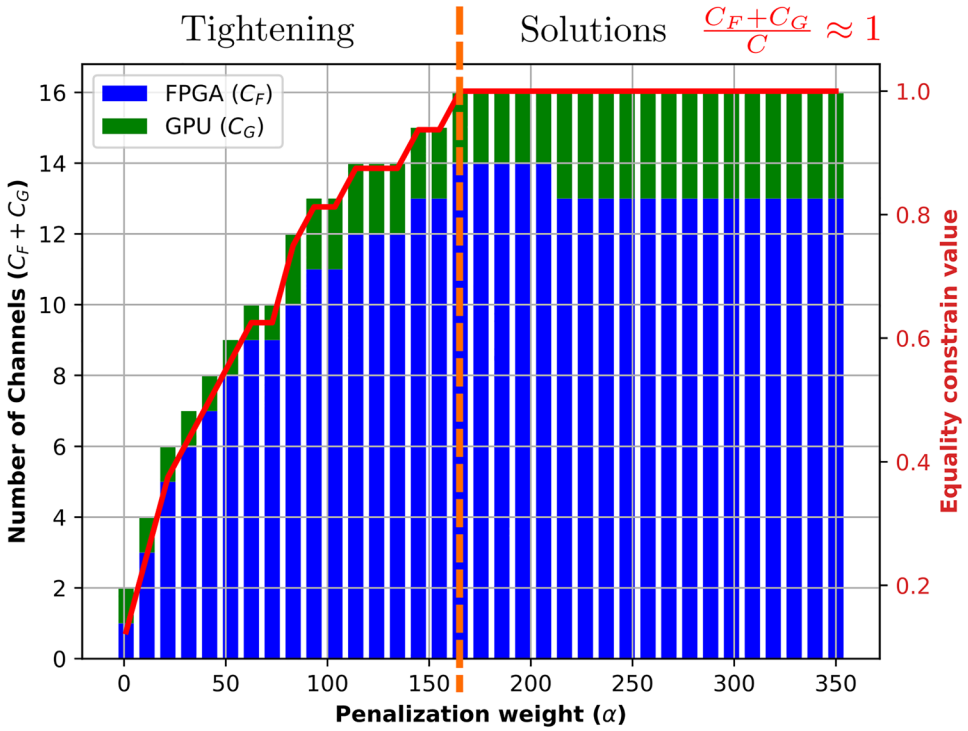



**Figure 3** First iterations of a relaxed GGP sequential grouped convolution partitioning of an input tensor with 16 channels ($C = 16$) with an increasing $\alpha$. The problem is solved as a set of GP problems and he tightening only takes a few iterations (iteration 17 with $\alpha = 170$) to find an acceptable solution. Each step is in polynomial time and total optimisation lasts less than a couple of hundred of milliseconds.

$$\begin{aligned} &\text{Computation} && LAT_C(\boldsymbol{X_C}) + LAT_F(\boldsymbol{X_F}) + LAT_G(\boldsymbol{X_G}) + \\ &\text{Communication} && LAT^{F-G}_{Comm}(\boldsymbol{X_{F-G}}) + LAT^{C-G}_{Comm}(\boldsymbol{X_{C-G}}) + LAT^{C-F}_{Comm}(\boldsymbol{X_{C-F}}) + \\ &\text{Penalization} && \alpha\bar{h}(C_C, C_F, C_G, \hat{C}_C, \hat{C}_F, \hat{C}_G) \end{aligned} \tag{9}$$

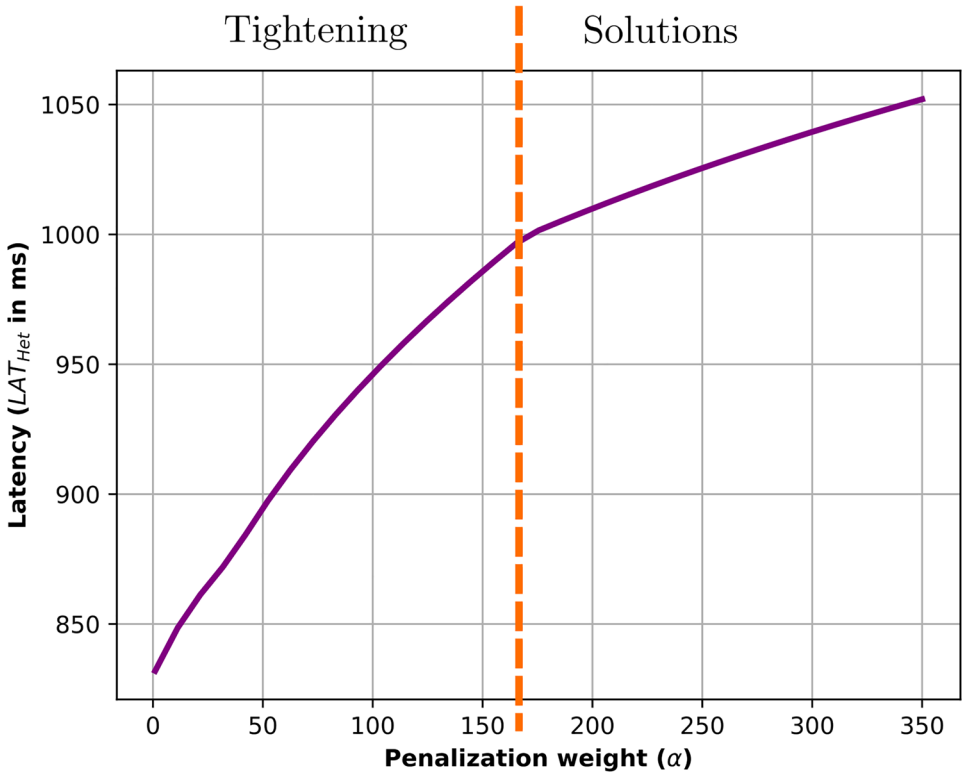



**Figure 4** Heterogeneous objective function per iteration (without penalization term) from problem in Eq. 7.

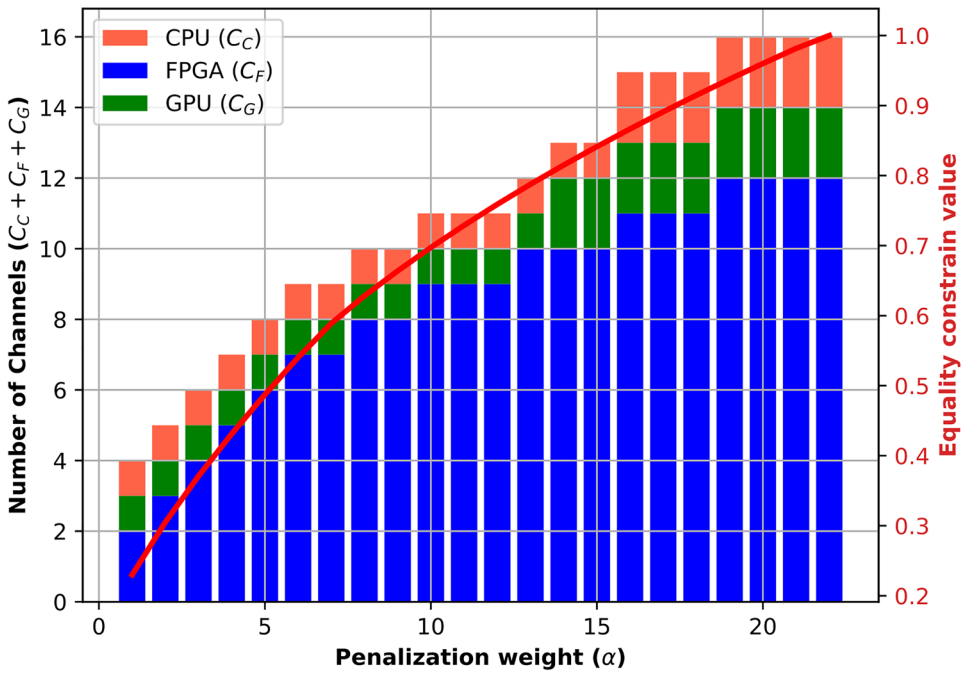



**Figure 5** Relaxed GGP sequential grouped convolution partitioning of an input tensor with 16 channels ($C = 16$) with an increasing $\alpha$ over a CPU-GPU-FPGA network.

Figure 5 shows an example of a partitioning by relaxation and tightening over the same channel-wise layer partitioning of the Fig. 3. The red bars represent the partition hosted in the CPU, the green bars those on the GPU and finally, the blue bars represent the channels. The found solution fits the partitions mostly on the FPGA, until the equality constraint is tight enough to fulfill the desired value ($C = 16$).

### 5.2 Full CNN Model Optimization

In previous Subsection, the limited size of the problem still allowed greedy methods to solve single-layer optimization partitioning and scheduling. An approximate solution can be found by simply mapping the biggest partition to the fastest device with available resources. In this specific use-case, the FPGA dominates in both execution time and energy consumption. Therefore, GGP will find an optimal solution similar to greedy algorithms. However, this claim does not take into consideration the complexity of partition mapping in deeper layers. If the algorithm chooses to fit the whole first layer in one device, deeper layers are not considered to be mapped on that device. Since the device has already exhausted its resources or at least until it multiplexes in time, it is not available until it finishes its scheduled workload. Unfortunately, one of the main limitations of DHM is that layers can not be multiplexed in time. Therefore, each layer on full CNN models are individually mapped theoretically on several FPGA accelerators, simulating time-multiplexing with reduced resources. This can be simulated due to the analytical modeling and the presented formulation of layer-wise optimization presented in this section. The setup from the Fig. 6 is analyzed as use-case scenario.

Being $L$ the number of convolutional layers in a CNN model, Eq. 10 represents the optimization problem reformulation for a complete network. The objective function now adds the individual performance model (latency for this example) of each layer $l$ ($l = 1, 2, 3, \ldots, L$). The resource inequality constraints are also modified to include the memory and computing elements utilization on each layer.

$$
\begin{aligned}
&\text{minimize} && \overbrace{\sum_{l=1}^{L} LAT_D(\boldsymbol{X}_{\boldsymbol{D}}^{l})}^{\text{Computation}} + \overbrace{\sum_{l=1}^{L} LAT_{Comm}(\boldsymbol{X}_{\boldsymbol{D}}^{l})}^{\text{Communication}} + \overbrace{\sum_{l=1}^{L} \alpha_l \bar{h}_l(\boldsymbol{X}_{\boldsymbol{D}}^{l}, \hat{\boldsymbol{X}}_{\boldsymbol{D}}^{l})^{-1}}^{\text{Penalization}} \\
&\text{subject to} && \sum_{l=1}^{L} ALM_F(\boldsymbol{X}_{\boldsymbol{F}}^{l}) \leq ALM_{max} \\
&&& \sum_{l=1}^{L} ALUT_F(\boldsymbol{X}_{\boldsymbol{F}}^{l}) \leq ALUT_{max} \\
&&& \sum_{l=1}^{L} LAB_F(\boldsymbol{X}_{\boldsymbol{F}}^{l}) \leq LAB_{max} \\
&&& \sum_{l=1}^{L} M20K_F(\boldsymbol{X}_{\boldsymbol{F}}^{l}) \leq M20K_{max} \\
&&& \sum_{l=1}^{L} \boldsymbol{X}_{\boldsymbol{D}}^{l} = \boldsymbol{X}^{l}
\end{aligned}
\tag{10}
$$

The summation and scaling (linear combination) of posynomial functions is also a posynomial [11, 15]. Thus, the objective and constraint functions are still posynomial and can be solved with GGP. Additionally, notice that from Eq. 10, the number equality constraints also increases linearly with respect to the number of layers in the CNN model.

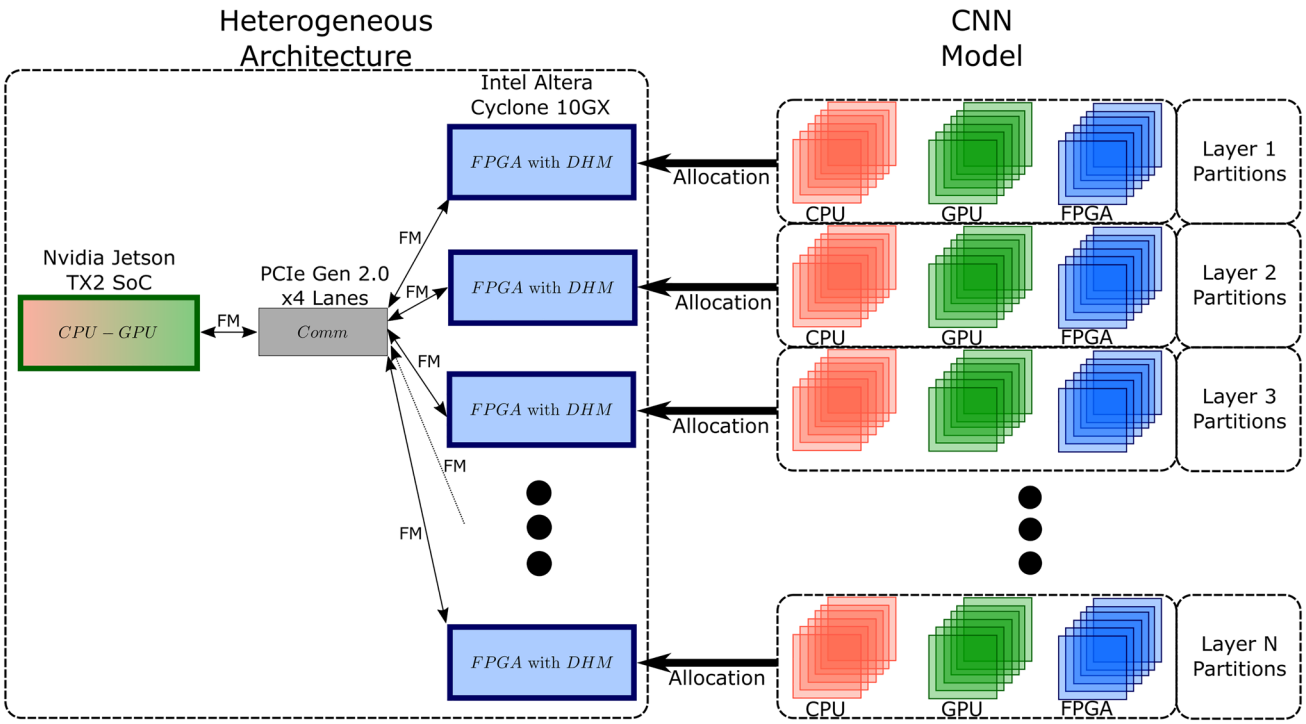



**Figure 6** **Setup 2:** Multi-layer setup of a full CNN model with several layers allocated. The heterogeneous architecture is theoretically simulated.

Therefore, several penalization weights ($\alpha_l$) and penalization function ($\bar{h}_l$) must be handled to iterative tighten the reformulated objective function. Each parameter $\alpha_l$ can be individually tightened as presented in previous subsection, until each constraint is fulfilled. Similarly, it is possible to increase simultaneously all the parameters on each step and individually stop each one when that specific equality constraint is fulfilled.

Although most of partitioning techniques are covered with the formulation of Eq. 10 (tiling, grouped convolution or channel-wise loop unrolling and depth-wise separable convolution), there is still one that can not be adapted to this formulation. The fused-layer considers that some FMs are not transferred between devices, in both sequential and concurrent execution. However, in Eq. 10, every single layer is considered to output an OFM that is intercommunicated between devices. The fused-layer technique consists in selecting which OFMs remain on the device to be computed as IFMs for the next layer, eliminating this way, the need of communication overhead. Consequently, a strategy must be chosen to reduce some terms of communication models $LAT_{Comm}(\boldsymbol{X}_{\boldsymbol{D}}^{\boldsymbol{l}})$. Since communication links are usually modelled with linear functions, ILP is a simple enough solution to explore all the combinations in polynomial time [6, 8]. In Eq. 11, each communication term in the heterogeneous objective function from Eq. 10 is multiplied by the Heaviside function $S(x) \in \mathbb{B}$, also known as step function. This way, the optimization technique chooses between keeping the FM in the device and skip communication ($S(x) = 0$) or to transfer the tensor ($S(x) = 1$) on each layer.

$$\text{minimize} \quad \sum_{l=1}^{L} LAT_{D}(\boldsymbol{X}_{\boldsymbol{D}}^{\boldsymbol{l}}) + \sum_{l=1}^{L} S(\boldsymbol{X}_{\boldsymbol{D}}^{\boldsymbol{l}}) \cdot LAT_{Comm}(\boldsymbol{X}_{\boldsymbol{D}}^{\boldsymbol{l}}) \tag{11}$$

Although the formulation from Stahl et al. [6, 8] is simple and solvable with ILP, it does not consider the execution computation time. Therefore, this is only useful in cases where the Computation-to-Communication Ratio (CCR) is low, result of a hardware platform heavily bounded by communication. On the other hand, while Eq. 11, is a generalization and extension of their work it is, from a GGP perspective, unsolvable. This is because the formulation from Eq. 11 presents several drawbacks. The most important being, that the shifted Heaviside function $H(x)$, from Eq. 12, is not a smooth differentiable function. Thus, it can not be solvable using interior-point algorithms, that depend on iterative gradient evaluation [15].

$$H(x) = \begin{cases} 0 & x < 1 \\ 1 & x \geq 1 \end{cases} \tag{12}$$

As a consequence, analytical approximations of the step function must be used. Well-known sigmoid-like functions, such as the logistic function or trigonometric functions like

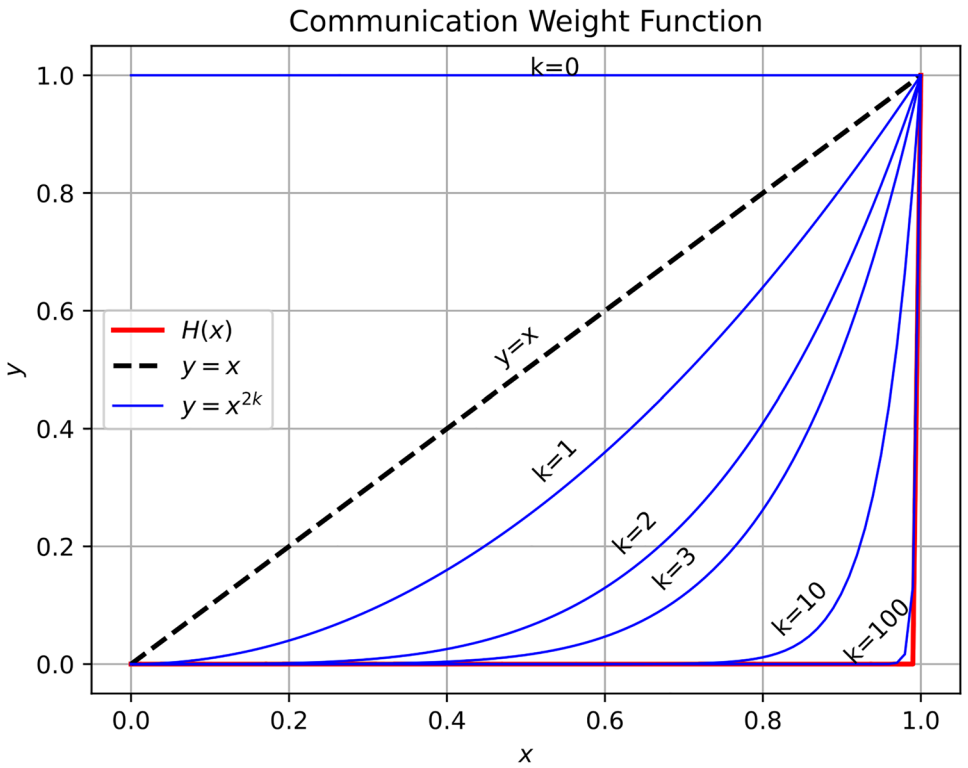



**Figure 7** Communication weight function $S(x) = x^{2k}$ for different values of $k$.

*arctan* and *tanh*, are suitable candidates [11]. Approximation techniques such as, cubic and spline interpolations are also commonly employed to smooth and clip or bound the proposed function [16]. Nevertheless, as discussed in Section 3, to preserve the posynomial properties, both terms in the product $S(x)$ and $LAT(\boldsymbol{X_D})$, must be also posynomials. Furthermore, convexity and monotonicity must be also preserved to be solvable with GGP. This restricts the number of usable functions for fused-layer formulation, since they must follow the algebraic form. Thus, in Eq. 13, a simple exponential algebraic function is defined as communication weight function.

$$S(x) = x^{2k}, k \in \mathbb{N} \tag{13}$$

Figure 7 shows many algebraic communication weight functions (blue solid line) for different values of $k \in \mathbb{N}$. Notice that, the bigger the value of $k$ is, the better the approximation is to the Heaviside function $H(x)$ (red solid line). Additionally, using a symmetrical algebraic function (like with an even exponent) $k$ can take also negative integer values. For simplicity purposes, we restrict $k$ to take only natural numbers, $\mathbb{N}$. Another important remark is that, this is only true for the interval $0 \leq x \leq 1$. Therefore, these newly introduced interval constraints must be also considered on the full CNN optimization problem formulation.

Although, formulation from Eq. 13 solves the approximation problem in a posynomial form that can be solved by GGP, it introduces a new heuristic value $k$. As shown in Fig. 8, the choice of this value has a direct impact on the function approximation. The $L2$-Norm is chosen to visually and numerically evaluate the error difference between the Heaviside function and the communication weight function ($||H(x) - S(x)||$). For the full CNN model optimization formulation, a value of $k = 100$ is selected with an $L2$-Norm error of around 0.12. It is important to considerate that a big value of $k$ can cause numerical issues that complicates convergence with no substantial difference between solutions.

Finally, the communication weight function $S(x)$ is included in the GGP formulation for the full model optimization. Equation 14 shows the modified objective function with the interval constraints of the approximation domain of $S(x)$. Since, $S(x)$ is a smooth differentiable posynomial, the weighted communication terms are still posynomials [11, 15]. Thus, this can be solved with the interior-point techniques, typical of GGP solutions. Considering that the derivative is mostly 0 for almost any value, except for values close to 1, any gradient-based function is heavily penalized by choosing features around 1. Additionally, the objective function also increases with the number of $S(\boldsymbol{X_D^l})$ for each layer that the features are transferred to another device.

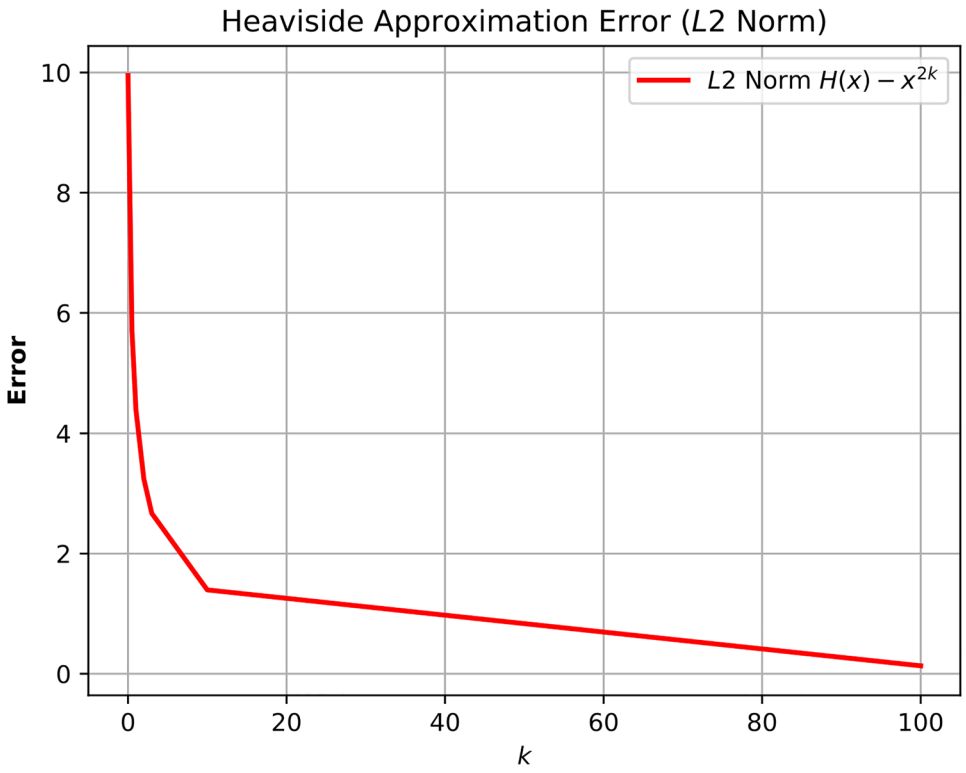



**Figure 8** Approximation error function based on *L2*-Norm.

$$
\begin{aligned}
\text{minimize} \quad & \overbrace{\sum_{l=1}^{L} LAT_D(\boldsymbol{X}_{\boldsymbol{D}}^{l})}^{\text{Computation}} + \overbrace{\sum_{l=1}^{L} S(\boldsymbol{X}_{\boldsymbol{D}}^{l}) \cdot LAT_{Comm}(\boldsymbol{X}_{\boldsymbol{D}}^{l})}^{\text{Weighted Communication}} + \overbrace{\sum_{l=1}^{L} \alpha_l \bar{h}_l(\boldsymbol{X}_{\boldsymbol{D}}^{l}, \hat{\boldsymbol{X}}_{\boldsymbol{D}}^{l})^{-1}}^{\text{Penalization}} \\
\text{subject to} \quad & \left.\begin{aligned}
&\sum_{l=1}^{L} ALM_F(\boldsymbol{X}_{\boldsymbol{F}}^{l}) \leq ALM_{max} \\
&\sum_{l=1}^{L} ALUT_F(\boldsymbol{X}_{\boldsymbol{F}}^{l}) \leq ALUT_{max} \\
&\sum_{l=1}^{L} LAB_F(\boldsymbol{X}_{\boldsymbol{F}}^{l}) \leq LAB_{max} \\
&\sum_{l=1}^{L} M20K_F(\boldsymbol{X}_{\boldsymbol{F}}^{l}) \leq M20K_{max}
\end{aligned}\right\} \text{FPGA Resource Constraints} \\
& \left.\sum_{l=1}^{L} \boldsymbol{X}_{\boldsymbol{D}}^{l} = \boldsymbol{X}^{l}\right\} \text{Partitioning Constraints} \\
& \left.0 \leq \frac{\boldsymbol{X}_{\boldsymbol{D}}^{l}}{\boldsymbol{X}^{l}}\right\} \text{Interval Constraints}
\end{aligned}
\tag{14}
$$

To evaluate the full model optimization partitioning, Fig. 9 shows the resulting partitions per layer from Eq. 14 formulation on three CNN model configurations. For instance, the partitioning was processed channel-wise with the heterogeneous GConv partitioning on a full model. For visualization purposes, the normalized number of channels computed per device is presented instead of the actual number of channels per layer. The channel layer-wise partitioning for AlexNet [17] is presented in Subfigure 9a. Subfigure 9b shows the channel distribution for VGG16 [18]. Finally, in Subfigure 9c, the resulting partitioning for ResNet18 [19] is presented. The latency of each device and the heterogeneous platform are compared against a single-device solution. The single-device is the CPU on the embedded platform with neither inter-device communication nor partitioning. This means, that the results of the dashed gray line (– –) represent

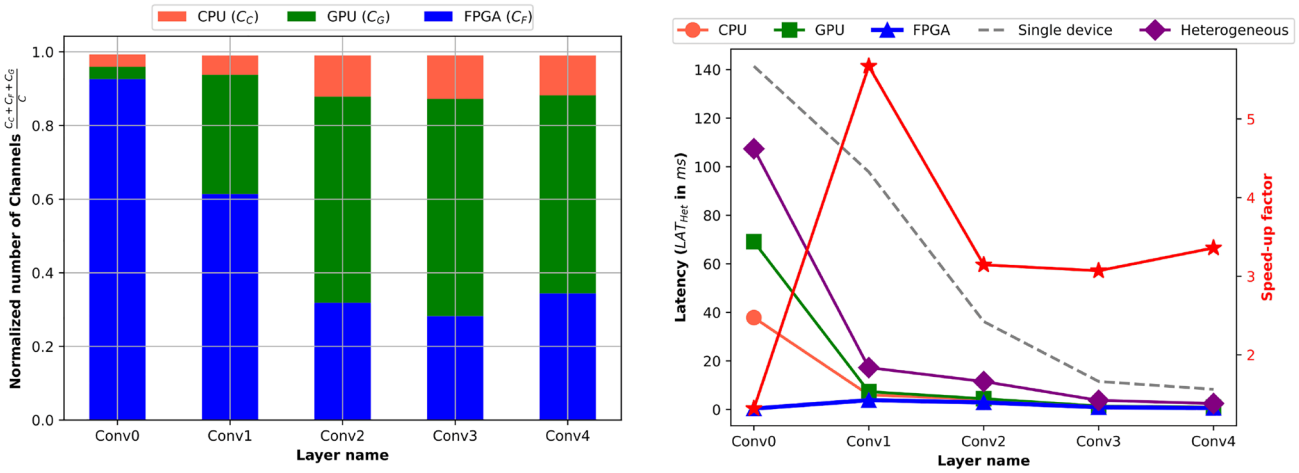
**Table 1** Optimization state-of-the-art comparison.

| Work | Edge platform | Optimization method | Partitioning Technique | Objective function | Constraints | Models | Gain |
|---|---|---|---|---|---|---|---|
| Zhao et al. [5] | 6 Raspberry Pi 3 Model B (CPU) | Work stealing/ sharing | Fused tile | Latency minimization | Memory footprint | YOLOv2 | 1.7× ∼ 3.5×) |
| | | | | | Communication overhead | | |
| de Oliveira and Borin [2] | Up to 63 CPU-based IoT clusters | Graph partitioning and swapping | Layer-wise | Inference rate maximization | Memory footprint | LeNet | 1.8×) |
| | | | Greedy | Communication minimization | Load balance | | |
| | | | METIS[a] | | | | |
| Stahl et al. [6] | Simulation of up to 7 devices | ILP | Layer-wise | Communication minimization | Partitioning selection | YOLOv2 | 1.15×) |
| | | | Fused layer | | | | |
| Zhou et al. [25] | 8 Raspberry Pi 3 Model B+ (CPU) | DP | Layer-wise | Comp/comm minimization | - | YOLOv2 | 1.5× ∼ 3.4×) |
| Stahl et al. [8] | 6 Raspberry Pi 4 (CPU) | ILP | Layer-wise | Communication minimization | Partitioning selection | YOLOv2 | Up to 2.8×) |
| | | | | | | AlexNet | Up to 1.2×) |
| | | | Fused layer | | | VGG16 | Up to 2.4×) |
| | | | | | | GoogLeNet | Up to 1.7×) |
| Zeng et al. [26] | 4 Raspberry Pi Jetson TX2 GPU Desktop CPU | Relaxed ILP to LP | Single- and Multi-layer | Energy minimization | Latency deadlines | AlexNet | 0.66×) |
| | | | | | | VGG16 | 0.64×) |
| | | | | | | GoogLeNet | 0.46×) |
| | | | | | | MobileNet | 0.25×) |
| This[b] | Jetson TX2 (CPU-GPU) + Cyclone10 GX (FPGA) | Relaxed GGP to GP | Tile-wise | Latency or Energy comp/comm minimization | Resources and partitioning | AlexNet | 3.1× ∼ 5.7×) |
| | | | Channel-wise | | | VGG16 | 2.8× ∼ 6.4×) |
| | | | Fused layer | | | ResNet18 | 2.8× ∼ 6.3×) |

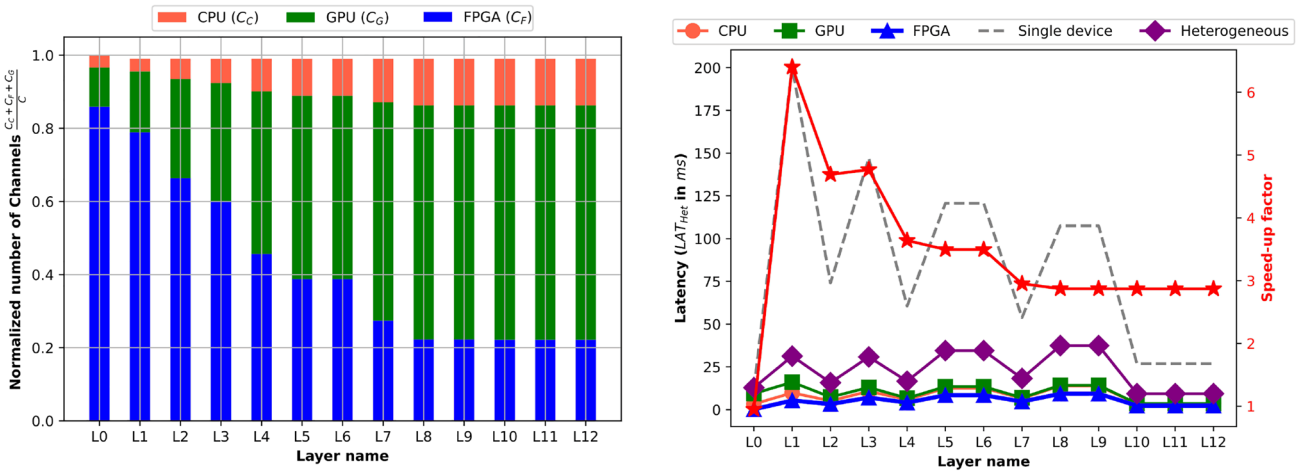
[a]Open-source software for automatic large graph partitioning

[b]Using theoretical architecture from Setup 2

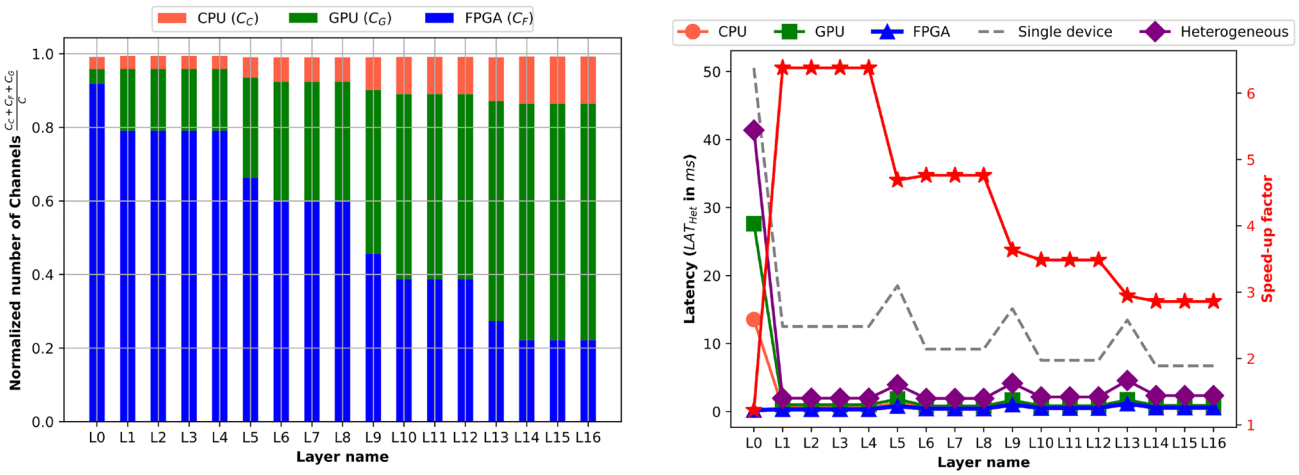
the full layer execution on CPU, while the purple solid line with diamond markers (−♦) represents the obtained heterogeneous solution. The solid red line with star markers (−⋆) represents the speed-up factor of the heterogeneous platform for each accelerated layer partition; compared against the single-device with no partition optimization nor acceleration. CPU partition is represented with an orange solid line with circle markers(−●), while the GPU partition performance is represented with a green solid line with square markers(−■) and the FPGA partition performance is represented with a blue solid line with triangle markers (−▲). The model configurations are based on ImageNet dataset [20] with an input image with dimensions $3 \times 224 \times 224$. From these results it can be observed that, compared against single layer channel optimization from previous Subsection, the resource utilization on the FPGA is distributed through all layers. Instead of mapping all channels of the first layers on the custom logic, as greedy-based algorithm, GGP optimization maps channel through deeper layers. However, as seen in the three Subfigures from Fig. 9, the GGP optimization formulation favors the first CNN layers to be directly mapped on the FPGA, which have a higher CCR. In the three cases, the input image was mapped over 80% on the FPGA (i.e. 61 channels from 64 for ResNet18). Deeper layers are bound by the communication overhead, which is a critical part of inter-device tensor transfers on heterogeneous platforms. Similarly, as evidence of this high CCR preference scalability, the three models present a different number of




(a) AlexNet.




(b) VGG16.




(c) ResNet18.

**Figure 9** Resulting GConv channel-wise optimized partitions for AlexNet Fig. 9a, VGG16 Fig. 9b and ResNet Fig. 9c.

layers, but also a similar result. This is, smaller partition percentages for deeper layers on FPGA accelerator and a workload dominated mostly by GPU (round 70% i.e. 352 channels from 512 for ResNet18). It is also important to mention, that in some cases a feasible solution can not be found if the resource constraints are too strict [6, 8]. Nevertheless, because of the mathematical properties of GGP, when a solution is found, this one is the *optimal* solution.

Table 1 displays the results from Fig. 9 using the Setup 2 from Fig. 6. These results are compared to state-of-the-art related works addressing partitioning optimization for CNNs models on the edge, for both simulated/theoretical and real-world case scenarios. Many approaches have been proposed for optimization formulation depending on the objective and constraint functions. Additionally, different partitioning schemes have an impact on the partition selection. Zhao et al. [5] propose a workload sharing and stealing for a Raspberry Pi 3 CPU cluster based on the memory capabilities of each node. The memory footprint and communication overhead are treated as constraints in the scheduling problem formulation, achieving a time execution speed-up from $1.7\times \sim 2.5\times$ for YOLOv2 [21]. As seen in Table 1, this technique, as many other partitioning methods, is inspired by the fused-layer [22]. For bigger edge device networks, de Oliveira and Borin [2], proposed the treatment of the hardware architecture network as a graph, taking advantage this way, of graph theory and operations to modify the graph. The authors in [2] demonstrate that their technique is less effective while using greedy techniques and hand-made layer-wise partitions. The graphs include a form of communication heterogeneity by allowing different bandwidths on the WIFI link between nodes. In [2], authors optimize the latency of inference by balancing the workload on LeNet [23] with a speed up of $1.8\times$. Stahl et al. [6] present a convex ILP optimization formulation aiming for communication overhead reduction. The authors obtained from this formulation a binary selection of layers to fuse for a platform simulation. Authors test the partitioning on YOLOv2 with a speed-up of 15%. Stahl et al. [8] extended their work for a physical platform consisting of multiple Raspberry Pis 4 on YOLOv2, AlexNet [17], VGG16 [18] and GoogLeNet [24] with a speed-up factor up to $2.8\times$, $1.2\times$, $2.4\times$ and $1.7\times$; respectively Dynamic Programming (DP). With a similar hardware network on the edge, Zhou et al. [25] proposed an unconstrained DP problem formulation that also includes the computation in the objective function achieving for YOLOv2 and VGG16a speed-up of around $1.5\times \sim 3.4\times$ and $1.1\times \sim 2.3\times$, respectively. Relaxing objective and constraint functions are a common method to accelerate solution evaluation and selection. In [26], Zeng et al. reduce a ILP to a Linear Programming (LP) problem by modifying some integer variables to continuous variables. Approximating this way, to a local minima solution. Additionally, in [26], the authors address the energy consumption of their heterogeneous platform by integrating direct energy measurements constrained to latency deadlines.

From Fig. 9, it has been demonstrated that by relaxing the problem formulation, similarly to [26]), it is possible to obtain a speed-up gain similar to the state-of-the-art works. As many of the discussed works [5, 6, 8, 26], the partitioning techniques include a mixture of layer-wise schemes with fused layer selection. However, the constraint function, for this section includes the logic and memory resources of the programmable logic of the FPGA. These considerations extend the capabilities of the partitioning optimization for hardware DSE. This custom logic awareness contrasts to other approaches for heterogeneous platforms. For instance, in the above discussed solutions, it is mostly focused on a fixed amount of memory resources, which is a common decision for CPU-GPU edge platforms. Furthermore, since not only latency can be modeled as a posynomial, but also the energy, the modification of the objective function is feasible for energy optimization. As observed in Fig. 9, the optimization solution tends to map shallower CNN layers on the FPGA for the three models. These layers are not only the most computational intensive in terms of number of Multiply and ACcumulates (MACs), but also the tensor to communicate are lighter [2]. This means, that these layers have a high CCR, which allow a suitable mapping on custom logic. However, since the communication overhead on heterogeneous systems is substantial, the first memory transfers without layer fusion introduces a considerable latency in the first layer to FPGA accelerator. Nevertheless, even considering this slow transfer the optimized heterogeneous partition solution still outperforms the single-device CPU solution. The speed-up factor is then reduced for deeper layers with values that ranges between $3.1\times \sim 5.7\times$, $2.8\times \sim 6.4\times$, and $2.8\times \sim 6.3\times$ for AlexNet, VGG16 and ResNet18, respectively.

Although, the previous examples mostly focus on latency performance, it is possible to obtain the energy metric performance in Joules ($E(\cdot)$) by simply integrating power in Watts ($P(\cdot)$) over the processing or latency time window in seconds ($LAT(\cdot)$) of one single FM. If power is constant, this is simplified to $E(\cdot) = P(\cdot) \times LAT(\cdot)$. As demonstrated in a previous work from the authors, constant power is often the case for power performance estimation at when maximum computing or communication capabilities are capped [27]. Therefore, any speed-up in latency translates in a direct reduction of energy consumption by the same factor. This is specially true for embedded devices, since layers from models like AlexNet, VGG16 and ResNet18, contain enough computation and communication to cap power dissipation of each accelerator. In cases where the accelerated partition is small enough that the power model ($P(\cdot)$) is not constant, then a similar analysis must be done. While the constrains

remain the same from Eq. 14, the objective function must be updated as show in Eq. 15:

$$\text{minimize} \quad \overbrace{\sum_{l=1}^{L} E_D(\boldsymbol{X}_{\boldsymbol{D}}^l)}^{\text{Computation}} + \overbrace{\sum_{l=1}^{L} S(\boldsymbol{X}_{\boldsymbol{D}}^l) \cdot E_{Comm}(\boldsymbol{X}_{\boldsymbol{D}}^l)}^{\text{Weighted Communication}} + \overbrace{\sum_{l=1}^{L} \alpha_l \bar{h}_l(\boldsymbol{X}_{\boldsymbol{D}}^l, \hat{\boldsymbol{X}}_{\boldsymbol{D}}^l)^{-1}}^{\text{Penalization}} \tag{15}$$

Assuming then constant power dissipation, the energy reduction factors remain similar to the latency speed-up factor: $3.1\times \sim 5.7\times$, $2.8\times \sim 6.4\times$, and $2.8\times \sim 6.3\times$ for AlexNet, VGG16 and ResNet18, respectively.

## 6 Conclusions

This work has proposed an automated method for CPU-GPU-FPGA partition selection of a given CNN layer. It has been shown that the partitioning problem can be modeled within the GGP framework, modeling each system performance metric in a posynomial form depending on CNN hyperparameters and architecture resource modeling. Well-known partitioning techniques in the state-of-the-art have been analyzed for layer-wise partitioning: tiling, grouped convolution, depth-wise separable convolutions and fused layers. An analytical formalization is then employed to derive a set of objective functions and constraints as a GGP problem, solvable in polynomial time without requiring a heuristic. It has been demonstrated that it is possible to relax some equality constraints by including a penalization term based on posynomial condensation, and reduce it as multiple simpler GP sub-problems. Experimental results targeting an embedded CPU-FPGA-GPU platform with state-of-the-art CNN layer configurations have demonstrated that the simplified problem is solvable in polynomial time.

**Funding** This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 765866.

**Data Availability** Authors declare that the data within this article can be generated from: https://zenodo.org/doi/10.5281/zenodo.10055869. Available data from a specific hardware platform has been described in: https://doi.org/10.1145/3594870.

## Declarations

**Competing Interests** All authors certify that they have no conflicts of interest to declare relevant to the content of this article.

## References

1. Zhou, L., Wen, H., Teodorescu, R., & Du, D. H. C. (2019). Distributing deep neural networks with containerized partitions at the edge. In *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*. USENIX Association, Renton, WA. https://www.usenix.org/conference/hotedge19/presentation/zhou
2. de Oliveira, F. M. C., & Borin, E. (2019). Partitioning convolutional neural networks to maximize the inference rate on constrained IoT devices. *Future Internet 2019: Innovative Topologies and Algorithms for Neural Networks 11*(10), 209. https://doi.org/10.3390/fi11100209
3. Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal, 49*(2), 291–307. https://doi.org/10.1002/j.1538-7305.1970.tb01770.x
4. Vanishree, K., George, A., Gunisetty, S., Subramanian, S., Kashyap, S., & Purnaprajna, M. (2020). CoIn: Accelerated CNN co-inference through data partitioning on heterogeneous devices. In *6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. https://doi.org/10.1109/ICACCS48705.2020.9074444
5. Zhao, Z., Barijough, K. M., & Gerstlauer, A. (2018). DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37*(11), 2348–2359. https://doi.org/10.1109/tcad.2018.2858384
6. Stahl, R., Zhao, Z., Mueller-Gritschneder, D., Gerstlauer, A., & Schlichtmann, U. (2019). Fully distributed deep learning inference on resource-constrained edge devices. In *Lecture Notes in Computer Science* (Vol. 11733, pp. 77–90). Springer International Publishing. https://doi.org/10.1007/978-3-030-27562-4_6
7. Busia, P., Minakova, S., Stefanov, T., Raffo, L., & Meloni, P. (2021). ALOHA: A unified platform-aware evaluation method for CNNs execution on heterogeneous systems at the edge. *IEEE Access, 9*, 133289–133308. https://doi.org/10.1109/access.2021.3115243
8. Stahl, R., Hoffman, A., Mueller-Gritschneder, D., Gerstlauer, A., & Schlichtmann, U. (2021). DeeperThings: Fully distributed CNN inference on resource-constrained edge devices. *International Journal of Parallel Programming*. https://doi.org/10.1007/s10766-021-00712-3
9. Cong, J., & Xiao, B. (2014). Minimizing computation in convolutional neural networks. In *Artificial Neural Networks and Machine Learning – ICANN 2014* (pp. 281–290). Springer International Publishing. https://doi.org/10.1007/978-3-319-11179-7_36
10. Abdelouahab, K., Pelcat, M., Sérot, J., Bourrasset, C., & Berry, F. (2017). Tactics to directly map CNN graphs on embedded FPGAs. *IEEE Embedded Systems Letters, 9*(4), 113–116. https://doi.org/10.1109/LES.2017.2743247
11. Boyd, S., Vandenberghe, L. (2004). Convex optimization. Cambridge University Press, Lieven Vandenberghe. https://doi.org/10.1017/cbo9780511804441
12. Rountree, D. H., & Rigler, A. K. (1982). A penalty treatment of equality constraints in generalized geometric programming. *Journal of Optimization Theory and Applications, 38*(2), 169–178. https://doi.org/10.1007/bf00934080
13. Agrawal, A., Diamond, S., & Boyd, S. (2019). Disciplined geometric programming. *Optimization Letters, 13*(5), 961–976. https://doi.org/10.1007/s11590-019-01422-z
14. Burns, S. A. (1987). Generalized geometric programming with many equality constraints. *International Journal for Numerical Methods Engineering, 24*(4), 725–741. https://doi.org/10.1002/nme.1620240406
15. Boyd, S., Kim, S.-J., Vandenberghe, L., & Hassibi, A. (2007). A tutorial on geometric programming. *Optimisation and Engineering, 8*(1), 67–127. https://doi.org/10.1007/s11081-007-9001-7
16. Auquiert, P., Gibaru, O., & Nyiri, E. (2007). On the cubic l 1 spline interpolant to the heaviside function. *Numerical Algorithms, 46*(4), 321–332. https://doi.org/10.1007/s11075-007-9140-0

17. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Neural Information Processing Systems (NIPS)*.
18. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. Preprint retrieved from https://doi.org/10.48550/ARXIV.1409.1556
19. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. Preprint retrieved from https://doi.org/10.48550/ARXIV.1512.03385
20. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. ISSN: 1063–6919. https://doi.org/10.1109/cvpr.2009.5206848
21. Redmon, J., & Farhadi, A. (2016). Yolo9000: Better, faser, stronger. *Computer Vision and Pattern Recognition (CVPR 2016)*.
22. Alwani, M., Ferdman, M., & Milder, P. (2016). Fused-layer CNN accelerators. *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. https://doi.org/10.1109/micro.2016.7783725
23. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y. (1999). Object Recognition with Gradient-Based Learning. In: *Shape, Contour and Grouping in Computer Vision*. Lecture Notes in Computer Science, vol 1681. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-46805-6_19
24. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. Preprint retrieved from https://doi.org/10.48550/ARXIV.1409.4842
25. Zhou, L., Samavatian, M.H., Bacha, A., Majumdar, S., & Teodorescu, R. (2019). Adaptive parallel execution of deep neural networks on heterogeneous edge devices. In Proceedings of the 4th ACM/IEEE Symposium on Edge Computing. ACM, IEEESymposium on Edge Computing. https://doi.org/10.1145/3318216.3363312
26. Zeng, L., Chen, X., Zhou, Z., Yang, L., & Zhang, J. (2021). CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices. *IEEE/ACM Transactions on Networking, 29*(2), 595–608. https://doi.org/10.1109/tnet.2020.3042320
27. Carballo-Hernández, W., Pelcat, M., Bhattacharyya, S. S., Galán, R. C., & Berry, F. (2023). Flydeling: Streamlined performance models for hardware acceleration of CNNs through system identification. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems, 8*(3), 1–33. https://doi.org/10.1145/3594870