Universiteit
Leiden
The Netherlands

System-level design for efficient execution of CNNs at the edge

Minakova, S.

Citation
Minakova, S. (2022, November 24). *System-level design for efficient execution of CNNs at the edge*. Retrieved from https://hdl.handle.net/1887/3487044

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3487044

Note: To cite this publication please use the final published version (if applicable).

Summary

Convolutional Neural Networks (CNNs) are biologically inspired computational models, characterized with the ability to handle large, unstructured data. Due to this ability, CNNs excel at tasks such as image classification, image segmentation, natural language processing, and are widely used to perform these tasks in applications such as navigation, facial recognition, medical images analysis, and others. Nowadays, many CNN-based applications are executed on edge platforms: mobile phones, tablets, cameras, etc. This stands in contrast to the more common practice in which CNN-based applications are executed on data centers (in the cloud). Unlike execution in the cloud, execution at the Edge does not require transmission of the collected data (e.g., images from a CCTV camera) over the Internet, and thus guarantees higher responsiveness and security.

However, execution of CNN-based applications at the Edge is challenging due to requirements posed on the CNNs by the application and the target edge platform. Among these requirements, the most common are high accuracy, high throughput, low latency, low memory cost, and low energy cost. These requirements make the design of a CNN executed at the Edge a complex task. Typically, this task is performed using the state-of-the-art (SOTA) design flow. The SOTA design flow explores CNNs with different architectures and parameters and tries to find a CNN which adheres to all the requirements posed on it.

While taking a good care of *what* is executed at the Edge (i.e., which architecture and which parameters does a CNN have), the SOTA design flow does not explore *how* a CNN or CNN-based application is executed (i.e., how it utilizes computational, memory, and energy resources available on the edge platform). Instead, the SOTA design flow adopts limitations that negatively affect the design of CNNs and CNN-based applications executed at the Edge. The first limitation is that a CNN is always executed layer-by-layer. This sequential manner of CNN execution is widespread due to its simplicity, but cannot guarantee efficient utilization of the resources available on the edge

platform. Consequently, a CNN designed using the SOTA design flow may utilize the limited resources of an edge platform inefficiently. The second limitation is that a CNN-based application only uses one CNN to perform its task. Due to this limitation, the SOTA design flow lacks the means for inter-CNN optimizations and run-time adaptivity, which are important to some CNN-based applications.

In this thesis, we aim to relax the two aforementioned limitations and reduce their negative impact on the design of CNN-based applications executed at the Edge. To this end, we extend the SOTA design flow and propose four novel methodologies within the extended design flow.

The first two methodologies focus on relaxing the first limitation. These methodologies find and enforce a non-sequential manner of CNN execution to ensure efficient utilization of the platform resources by a CNN. The first methodology efficiently distributes (maps) the computations within a CNN to the computational resources of a target edge platform and thereby increases the CNN throughput. The second methodology splits the data exchanged between CNN layers into parts and reuses the platform memory among the data parts, thus reducing the memory footprint of the CNN.

The last two methodologies focus on relaxing the second limitation. These methodologies optimize CNN-based application beyond optimizing the individual CNNs. The third methodology introduces run-time adaptivity into a CNN-based application. This enables for the design and efficient execution of an application which needs can change at run-time. The fourth methodology performs joint memory optimization of multi-CNN applications (applications that use multiple CNNs to perform their task). Thus, the methodology offers high rates of memory compression to fit multi-CNN applications into the limited memory resources of an edge platform.