Pruning and Early-Exit Co-Optimization for CNN Acceleration on FPGAs

Guilherme Korol*, Michael Guilherme Jordan*, Mateus Beck Rutzig†,
Jeronimo Castrillon‡§, Antonio Carlos Schneider Beck*
*Institute of Informatics, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
†Electronics and Computing Department, Universidade Federal de Santa Maria (UFSM), Santa Maria, Brazil
‡ Center for Advancing Electronics Dresden, TU Dresden, Dresden, Germany
§ Center for Scalable Data Analytics and Artificial Intelligence, Dresden, Germany
*{gskorol,mgjordan,caco}@inf.ufrgs.br,†mateus@inf.ufsm.br,‡jeronimo.castrillon@tu-dresden.de

***Abstract*—The challenge of processing heavy-load ML tasks, particularly CNN-based ones at resource-constrained IoT devices, has encouraged the use of edge servers. The edge offers performance levels higher than the end devices and better latency and security levels than the Cloud. On top of that, the rising complexity of ML applications, the ever-increasing number of connected devices, and the current demands for energy efficiency require optimizing such CNN models. Pruning and early-exit are notable optimizations that have been successfully used to alleviate the computational cost of inference. However, these optimizations have not yet been exploited simultaneously: while pruning is usually applied at design time, which involves retraining the CNN before deployment, early-exit is inherently dynamic. In this work, we propose AdaPEx, a framework that exploits the intrinsic reconfigurable FPGA capabilities so both can be cooperatively employed. AdaPEx first explores the trade-off between pruning and early-exit at design-time, creating a design space never exploited in the state-of-the-art. Then, AdaPEx applies FPGA reconfiguration as a means to enable the combined use of pruning and early-exit dynamically. At runtime, this allows matching the inference processing to the current edge conditions and a user-configurable accuracy threshold. In a smart IoT application, AdaPEx processes up to 1.32× more inferences and improves EDP by up to 2.55× over the state-of-the-art FPGA-based FINN accelerator.**

***Keywords*—Edge Computing, Adaptive Inference, CNN, FPGA.**

I. Introduction

Due to thermal and energy constraints, many IoT devices are restricted in their processing capabilities, requiring computing-intensive tasks to get processed elsewhere. One alternative is to connect these devices and offload data to Edge servers, which are physically closer than the cloud, resulting in lower latency and increased security. Convolutional Neural Networks (CNNs) are a representative example of such heavy tasks. They are used for many IoT applications, from intelligent manufacturing to smart video surveillance. However, even very powerful Edge servers may not satisfy the current levels of efficiency demanded by modern applications. Therefore, there is a need to not only optimize the hardware platforms, but also to improve the CNN models running on top of them [1–3].

With respect to CNN model optimizations, pruning [4] and early-exit [5] have been shown to be very prominent alternatives. Pruning removes parts of a CNN to improve performance at the cost of accuracy, while Early-Exit adds branches to a CNN so that the inference may finish earlier, reducing the processing time. These techniques have already been independently and successfully employed, but no work has shown the benefits of combining these two optimizations. Combining these strategies is non trivial since, traditionally, pruning is defined and applied before the implemented CNN model is deployed (i.e., at design time) while early-exit is a dynamic technique (i.e., works during the inference). Therefore, the challenges lie (*i*) in enabling their simultaneous use *at runtime* to adapt the inference processing; and (*ii*) in considering their combined impact on accuracy and inference costs.

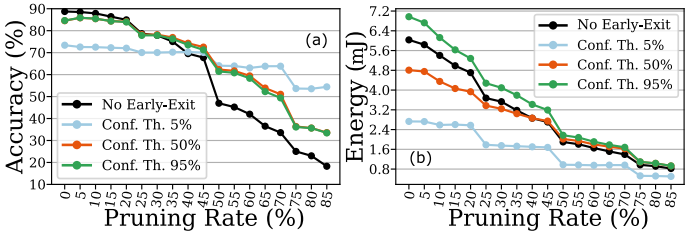

Figure 1. Accuracy (a) and Energy (b) w.r.t Pruning of CNVW2A2 on CIFAR10 with no Early-Exit and with Early-Exit under three Confidence Thresholds.

With the aforementioned challenges in mind, we propose AdaPEx, for Adaptive Pruning of Early-Exit CNNs, a two-step framework that exploits the intrinsic reconfigurable nature of FPGAs. *At design time*, AdaPEx automatically generates a library of pruned early-exit models with different resource and accuracy profiles. *At runtime*, AdaPEx offers a mechanism that dynamically chooses the best pruning rate and confidence threshold to adapt the inference serving, automatically reconfiguring the FPGA as needed. With that, AdaPEx enlarges the design options and increases the number of processed inferences with less energy, according to the workload and accuracy demands at a given moment.

AdaPEx exposes two optimization knobs: the *pruning rate*, which controls how much of the CNN gets pruned away; and the *confidence threshold*, which defines an expected level of certainty in the CNN outputs to guide the early-exit. As will be explained later, lower values of this threshold make more inputs to get classified (i.e. exited) earlier, lowering the processing time. Figure 1(a) shows the new optimization opportunities enabled by AdaPEx. It plots the accuracy (y-axis) over pruning rates varying from 0 to 85% (x-axis) of the CNN with no early-exit and of the same CNN model but with early-exits, considering three different confidence thresholds (5, 50, and 95%). They are all executed on the FPGA accelerator FINN [6]. As we increase the pruned portion of the CNN, accuracy naturally drops. However, for the early-exit models, an interesting behavior can be observed: while the classification under 5% confidence threshold (blue curve) gives the poorest accuracy for lightly pruned CNNs (from 0 to 45% pruning rates), when dealing with heavier pruning rates, the 5% confidence threshold returns the highest accuracy. The extra design space opened by AdaPEX also impacts energy consumption, as observed in Figure 1(b),

which gives the energy per inference (y-axis) over pruning rates ranging from 0 to 85% (x-axis) for the same CNNs. For instance, when setting the early-exit confidence to 50% (orange curve), it ends up saving energy over the CNN with no early-exit for pruning rates of up to 40% only, after that energy consumption is increased. In this context, AdaPEx enables an important trade-off between pruning rate and confidence threshold that allows, for instance, recovering some of the accuracy lost from pruning with early-exit, and saving energy with smaller, pruned, CNNs.

Concretely, this work makes the following contributions:

- Presents a novel optimization approach combining pruning and early-exit in a single framework for exploring the accuracy-performance-energy trade-off;
- Proposes AdaPEx: a framework that, at design time, leverages the design space to build a library that can be used at runtime by reconfiguring the FPGA to match to inference processing to the current edge conditions;
- Evaluates AdaPEx against the state-of-the-art FINN accelerator under an Edge server scenario, increasing the number of processed inferences in up to 1.32× at 2.55× reduced energy-delay product.

II. Background

CNN Optimizations. Several optimizations have been proposed to minimize CNN requirements of millions of weights and multiply-accumulate (MAC) operations (see a sample CNN over the yellow background in Figure 2). Notably, compression methods, like pruning and quantization, have successfully reduced these requirements at small accuracy costs. **Pruning** is especially recommended to reduce the CNN memory footprint *and* computation (i.e., MACs) at inference (see a pruned CNN over the blue background in Figure 2). In particular, filter pruning [4] is the technique used in this work. It removes filters from the CNN weight matrices, creating no sparsity (keeping memory access regular), facilitating the use of the existing hardware infrastructure. Removing filters from a CONV layer also reduces the number of channels of the output feature map.

While pruning is usually statically applied, offering fixed reductions in computation and storage costs, **early-exit** [5, 7] is a dynamic optimization that takes advantage of certain inputs being "easier" to process. For these "easy" inputs, not all layers of the CNN are needed, so it can finish earlier (i.e., in a layer prior to the last) by following these so-called exits or branches (connected to the CNN original layers, called *backbone*). See an example in Figure 2 over the green background with a single earlier exit. In Figure 2, two output vectors are produced that will have length equal to the number of classes of the dataset (e.g., 10 and 43 classes in our evaluated datasets, CIFAR-10 and GTSRB, respectively). From these output vectors (early or not), the probability of the input being member of each class (i.e., the actual classification) is calculated from the softmax function: $\sigma(y)_i = e^{y_i} / \sum_{j=1}^{K} e^{y_j}$ for the output vector y of K classes. When running inferences, the early-exit CNN must decide whether or not to take earlier exits (e.g., accept the early output vector in Figure 2). This decision is usually based on the confidence that the current input has already been correctly classified with the layers processed so far. Precisely, whenever an exit outputs a classification with confidence above a certain value (called *Confidence Threshold*, from 0 to 100%) the output is accepted, and the inference is completed. Using the softmax (same function used to get the classes probabilities) of the exit output vector is one popular way to measure the exit confidence.

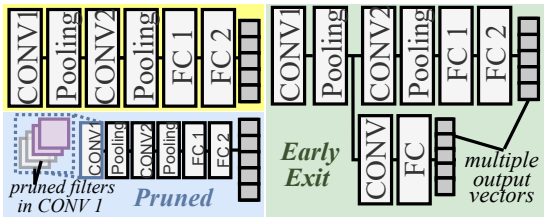

Figure 2. Topology of a sample CNN and its pruned and early-exit versions.

So, high probability values means high confidence. Therefore, by lowering the confidence threshold, more inputs are allowed to get classified earlier since the expected confidence is relaxed.

FPGA-Based CNN Acceleration with FINN. In this work, we adopt FINN [6], a state-of-the-art framework for mapping Deep Neural Networks (e.g., CNN) to FPGA. FINN is a popular open-source tool from Xilinx, which is being heavily used in both academia and industry. FPGA designs synthesized by FINN are *dataflow accelerators*. Dataflow (or streaming) accelerators offer a good compromise between performance and the ability to execute a wide range of more complex CNNs, relying on a pipelined architecture mapping each CNN layer to a hardware module. FINN employs hardware modules implemented as a set of High-Level Synthesis (HLS) template classes configured with parameters (e.g., kernel size, stride, etc) from the CNN layer it is in charge of executing. The main HLS module in the FINN infrastructure is the Matrix-Vector-Threshold Unit (MVTU) that is used to map CONV and fully-connected (FC) layers. FC layers get mapped directly to MVTU modules. CONV layers, on the other hand, need an auxiliary module, the Sliding Window Unit (SWU) that prepares the input feature map before it can be multiplied with the weight matrix at the MVTU. FINN allows the user to tune the accelerator parallelism through a JSON configuration file specifying the number of processing elements (PEs) and SIMD lanes of every MVTU. Therefore, each generated dataflow accelerator is "hard-wired" to its CNN.

III. Related Work

At the Edge, inference servers have been primarily used to process CNN inferences on data offloaded from connected IoT devices. In this context, FPGAs present a good alternative to accelerate these incoming inferences [8–10], while consuming less energy than GPU boards with equivalent accuracy [11], or even superior performance [12], with small accuracy drops.

However, to cope with the demanded efficiency levels, CNN models also need to be optimized. Pruning has been extensively used for adapting the inference processing either on GPUs [2] and FPGAs [10, 13]. Authors in [10] propose a framework that adapts the inference processing by switching the pruned model at runtime on the FPGA. In [13], a toolflow that statically customize the CNN pruning to the underlying FPGA accelerator is proposed. Early-exit has also been used in several works to enable runtime adaptation. At the Edge, [1] and [14] propose frameworks for optimizing and deploying early-exit models on embedded GPUs. For FPGAs, [15] implements an early-exit model by reconfiguring the FPGA at each not-taken exit to load the next set of layers. In contrast, [16] presents a hardware-aware tool for placing early exits in ResNets targeting the best trade-off between accuracy and computational cost.

Wrap-up and Our Contributions. It is known that adapting the inference processing is crucial at the Edge [1–3, 10, 14]. Until now, however, state-of-the-art works have restricted this adaptation to a single optimization method, i.e., pruning only

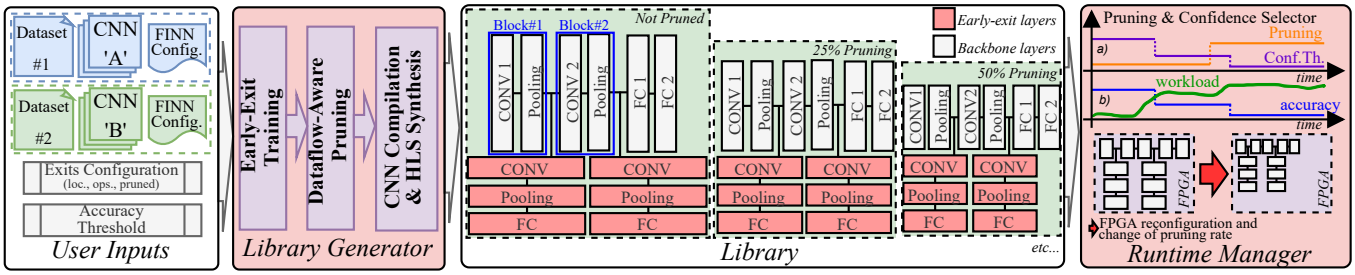

Figure 3. AdaPEx's Workflow.

or early exiting only. One of the reasons behind it is that these optimizations are traditionally applied at different stages of deployment (statically pruning *versus* dynamically exiting). AdaPEx advances the state-of-the-art by (*i*) combining pruning and early-exit on CNNs and (*ii*) exploiting the FPGA reconfigurable capabilities to provide fully adaptive inference processing at the Edge using these new optimization opportunities.

IV. AdaPEx

Figure 3 shows AdaPEx's two-step workflow from the library generation at design-time up to the inference runtime adaptation.

A. AdaPEx at Design-time

AdaPEx design-time consists of the **Library Generator** that creates a library containing multiple pruned early-exit CNNs and their accelerators. Initially, it reads the user's CNNs/datasets and a configuration file specifying the early exits. Then, the generator adds the early exits to the CNN model and train the CNNs ("Early-Exit Training" in Fig. 3). Afterward, AdaPEx varies the pruning rate at fixed steps, gathering multiple pruned versions of each early-exit CNN ("Dataflow-Aware Pruning" in Fig. 3). These versions offer multiple design points on the accuracy-resource-latency trade-off. Once the pruned models have been exported as ONNX files, they can be passed on to the "CNN Compilation & HLS Synthesis" that invokes the modified FINN tool to compile the ONNX to HLS modules so that Vivado can synthesize them. With the generated CNN models and accelerators, the Library is created as a table containing a list of pruned early-exit CNNs (rows) with their accuracy (extracted on the dataset test set) as well as the throughput (in Inferences per Second, IPS) values (extracted during synthesis). Below, we detail the AdaPEx early-exit and pruning optimizations.

1) Early-Exit: A series of modifications to the existing FINN/Brevitas design flow were required to enable inference on CNN models with multiple branches (see backbone and early exit layers in Figure 3 Library). First, early-exit models need to be described in Brevitas, a PyTorch-based tool for quantization-aware training from Xilinx that is part of the FINN infrastructure. Starting from a regular CNN model, AdaPEx can attach the early-exits to any location along the CNN topology. Where to place and how to configure the early-exits is an active research topic in areas like Neural Architecture Search (NAS), Auto-ML, etc. Therefore, the user can specify how AdaPEx adds the early exits (see "Exits Configuration" in Fig. 3), setting the location (*loc.* - i.e., after which backbone layers) and the operations (*ops.* - e.g., CONV, FC, etc.) of the exits, which also enables easy exploration of the design space.

Let us take as example the CNN used as case study in this work, the CNV, a VGG-like CNN available in FINN. We use the CNN block structure (sequence of layers with same configuration, see Fig. 3 Library with two sample blocks) to set in AdaPEx the locations of each exit, making AdaPEx to add two early exits to CNV: one after the second CONV layer (first block) and another exit after the fourth CONV layer (second block). As for configuring the early exits, we set AdaPEx to appended a CONV layer (with the same configuration of the block, number of channels, filter size, etc.) followed by a max-pool layer with kernel size of $k = \lfloor \frac{DIM}{2} \rfloor$, where DIM is the dimension of the block's output feature map, so it significantly reduces the map size, making FPGA synthesis feasible for the two following FC layers (that use the same configuration as the FC layers in the original CNV). The user configuration used as case-study (with CONV, Max-Pool, and FC layers) follows previous works on early-exit [5, 7].

After the model is fully defined with all its exits in Brevitas, the model can be trained. AdaPEx uses a training procedure [5] implemented as a Brevitas script where, instead of optimizing for the traditional loss function (from a single exit), all exits are simultaneously trained by optimizing for a weighted sum of the loss functions of each exit, called Joint Loss Function: $J_{loss} = \sum_{n=1}^{N} w_n L(\hat{y}_{exit_n}, y, \theta)$, where N is the number of exits, w_n the exit's weight, and L the traditional loss function accepting the exit's softmax $\hat{y}_{exit_n}$, ground-truth y, and the model weights θ. Besides, as explained in Section II, AdaPEx takes the softmax on each exit as a measure of their confidence.

From the FPGA point of view, a new HLS module had to be developed and included in the FINN design flow, along with a new transformation step to manage it during FPGA mapping. This new HLS *branch* module performs the branches between the CNN backbone and the early exits. FINN connects MVTU modules (the HLS module in charge of executing each CONV and FC layer) with AXI stream interfaces (implemented as FIFOs, in the input/output ports of each MVTU). In that way, the output of every HLS module corresponds to the CNN layer's output feature map and is fed to the next module in the dataflow. The new branch module leverages this implementation style by duplicating the incoming stream into two independent streams (one feeding the backbone, the other feeding the early exit). Therefore, some FPGA resource overhead will be observed (mainly in terms of BRAMs), but neither backbone nor exit throughput is undermined, and there is no risk of pipeline stalls.

2) Pruning: To prune out parts of a CNN that will be later executed on the FPGA, AdaPEx implements in Brevitas a pruning mechanism based on [10], called Dataflow-Aware Pruning, which, besides the CNN model, takes into account properties of the dataflow accelerator. Respecting such properties guarantees that the pruned CNN models get synthesized to the accelerators configured by the user. Notably, two properties in FINN have to be met to ensure correct feeding and synchronization of all PEs and SIMD lanes of every MVTU: the number of PEs must divide the number of CONV filters, and the number of SIMD lanes must divide the number of input channels.

For each pruned model, the Dataflow-Aware Pruning takes an initial CNN model, a user-defined FINN configuration file (containing parameters like number of PE/SIMD in JSON format, "FINN Config." in Figure 3), and a pruning rate (percentage specifying how many filters to prune). Then, for every convolutional layer, the procedure prunes r_i filters in such a way that it respects the $({ch_{out}}^i - r_i)\ mod\ (PE_i) = 0$ and $({ch_{out}}^i - r_i) mod (SIMD_{i+1}) = 0$ constraints, where PE_i and $SIMD_{i+1}$ give the MVTU's number of PEs and SIMD lanes of current i and next layer $i+1$, respectively. ${ch_{out}}^i$ gives the not-pruned number of channels for that layer (from the initial CNN). If the constraints are not met, the procedure iteratively decreases r_i until they are met. Then, r_i filters are pruned based on their ranking of the sum, from the floating-point representation, of its absolute weight values (ℓ_1-norm) [4].

Two approaches can be taken when pruning an early-exit model: pruning only the backbone layers or pruning the backbone and the CONV layers inside each early-exit (see backbone/exit layers in Figure 3). While the latter approach will speed up the early exits as well as the backbone layers, it may significantly decrease the exit's accuracy since these exits can be less resilient to pruning. On the other hand, the former approach, which leaves the exits CONV layers untouched, offers an exciting trade-off between performance and accuracy when comparing the early to the last exits. AdaPEx supports both approaches (which the user can set to either one or both with the *pruned* flag in the Exits Configuration, see Figure 3). These approaches will be compared in the Section VI. After pruning the early-exit CNN, it is retrained and exported as an ONNX file to be compiled by FINN.

B. AdaPEx at Runtime

AdaPEx's Library is used by the **Runtime Manager** during the second step (Figure 3), performing runtime adaptation of the inference processing. Whenever a change in the workload is flagged (possible with performance monitors added to the software in charge of the incoming inferences), the runtime manager searches in the library for pruning rate and confidence threshold that are most adequate to the current workload and the user's accuracy threshold. Thus, it can either change the confidence threshold, changing the acceptability of the earlier exits (as seen above, lowering the threshold cause more inputs to get classified earlier); or, change the pruning rate. Unlike the confidence threshold, changing the pruning rate requires reconfiguring the FPGA to switch the running accelerator, since, as already mentioned, each dataflow accelerator is synthesized (i.e. hardwired) to a particular pruned CNN. The Runtime Manager runs along the FINN host code (executing on the board GPP connected to the FPGA that manages the inferences, sending new inputs and collecting results).

The search on the Library takes as input the user's *accuracy threshold* (see Fig. 3), configured before deployment, and information on the current workload (i.e., incoming inferences per second, IPS), sampled at runtime. The Runtime Manager will select the models with accuracy above the threshold and with sufficient throughput for the incoming workload (recall that accuracy and throughput values were gathered at design-time and are stored in the Library). Whenever there is more than one model above threshold, the Runtime Manager will select the one with the highest accuracy (averaged on all exits).

Figure 3 right side illustrates the Runtime Manager at work. Plot *a* shows the currently selected pruning rate (orange curve) and confidence threshold (purple curve) and *b* shows a workload

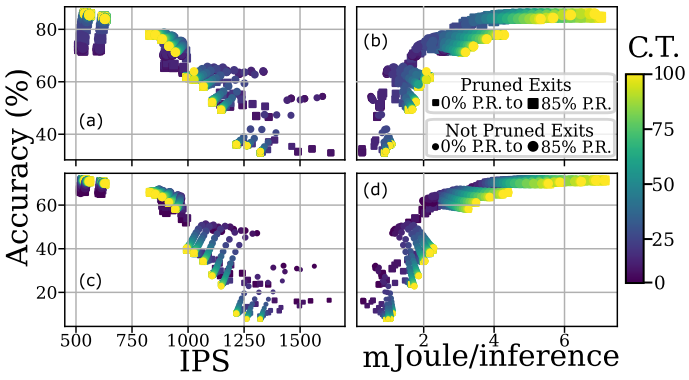

Figure 4. AdaPEx design space for CNVW2A2 on IPS and Joule per inference on CIFAR-10 dataset (plots a and b) and GTSRB (plots c and d).

curve (green) and the delivered accuracy (blue). The initial relatively low workload allows for a low pruning rate and a high confidence threshold, resulting in high accuracy levels. After this initial phase, an increase in workload is observed, which the Runtime Manager counters by lowering the confidence threshold (enabling faster inferences). Workload keeps rising until the Runtime Manager switches the pruning rate to a higher value, with an accelerator of a smaller area footprint and faster processing, but also with lower accuracy levels.

V. Methodology

Accelerators used across our experiments were synthesized within the Xilinx's FINN design flow [6] with Vivado targeting a Xilinx Zynq Ultrascale+ MPSoC ZCU104 board (XCZU7EV) at 100MHz. We used Xilinx Vivado for resource usage and power extraction and Verilator RTL simulations for performance.

We adopted the CNV CNN from FINN for evaluation with 2-bits quantization (CNVW2A2). Models were adapted to the CIFAR-10 and the German Traffic Sign Recognition Benchmark (GTSRB) datasets. AdaPEx generates 18 models for each initial early-exit CNN with pruning rates from 0% (not-pruned) to 85% (5% steps). Each model generates a specific FINN accelerator. On each pruned model, the confidence threshold can vary from 0 to 100% at 5% steps. All images consider CIFAR-10's image resolution (3x32x32). Accuracy results are reported on Brevitas TOP-1 test accuracy. The early-exit training procedure follows [5], weighting the first exit at 1.0 and the remaining at 0.3. Early-exit models are pruned and retrained for 40 epochs [4], with standard data augmentation and learning rate of 0.001 with decay of 0.1. Training was performed on Intel Xeon E5-2640 with NVIDIA Tesla K20m GPU.

We base our evaluation on a typical IoT application, smart video surveillance, with numerous cameras requesting frames to be inferred at a local Edge server. To keep the discussion general, we will refer to those requests as inference requests. For that, we model 20 cameras requesting inferences at the rate of 30 Inferences per Second (IPS) for 25 seconds. Due to factors like IPS fluctuation, network congestion, or the variable number of connected cameras, the rate of incoming inference requests (workload) changes over time [17], represented as 30% random workload deviation every 5 seconds. Experiments are executed 100 times, and average values are reported. We have set the maximum accuracy loss (AdaPEx's accuracy threshold) to 10%. Next, we evaluate performance, Quality of Experience (here defined as the product of accuracy by the percentage of processed frames), power, and energy of AdaPEx over the following baselines:

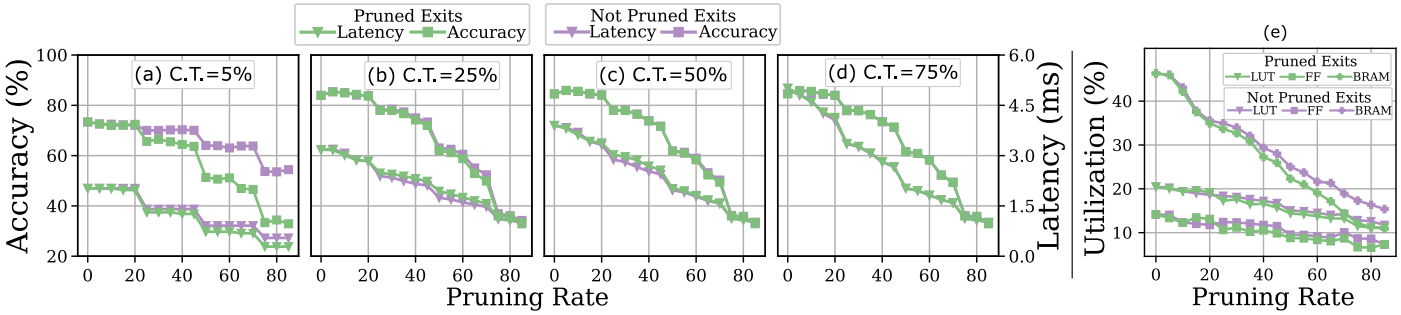

Figure 5. Accuracy (left y-axes) and latency (right y-axes) over pruned CNNs with Confidence Thresholds (C.T.) of 5, 25, 50, and 75% on plots from (a) to (d). On plot (e), resource usage on a XCZU7EV board for the Early-Exit CNVW2WA on the CIFAR-10 dataset.

- Original **FINN** accelerator synthesized to off-the-shelf CNN models;
- Pruning-Only, called **PR-Only**, that uses the runtime selection presented in Sec. IV-B, but with a single (no early) exit to evaluate the pruning benefits;
- Confidence-Only, a not-pruned early-exit model, called **CT-Only** that also uses the runtime selection but adapts the Confidence Threshold only.

VI. Results

This section presents the design space enabled by combining pruning and early-exit. It then discusses evaluation results of AdaPEx on the aforementioned application, showing the benefits of exploiting the new design space at runtime.

A. AdaPEx's Design Space

Newly Created Design Space. Figure 4 presents the design space enabled by the AdaPEx combination of pruning and early-exit for the two evaluated datasets: CIFAR-10 (upper plots) and GTSRB (lower plots). Figures (a) and (c) plots throughput (in IPS) versus accuracy (y-axis) while Figures (b) and (d) give the energy per inference versus accuracy. Design points are generated by varying the pruning rate (P.R.) from 0 to 85% (indicated by the size of each point) and the confidence threshold (C.T.) from 0 to 100% (indicated by the color scale) for both pruned exits (squares) and not pruned exits (circles). From plots (a) and (c), we see that CNN models (and accelerators) of lower accuracy that run faster are required to cope with high workload levels. A similar behavior is observed on the energy plots (b and d), but with a noticeable plateau from around 4mJ onwards for both datasets. Beyond 4mJ, the extra energy consumed by targeting inferences of higher accuracy is wasted. As can be observed, it is crucial to match the CNN model, by setting P.R. or C.T., to the current workload at runtime. This way, no accuracy is unnecessarily lost while meeting throughput constraints with minimal energy.

Pruned Early-Exits. One important design decision of pruning early-exit CNNs is how to handle the exit layers, i.e., whether or not to prune layers in the early exits. As shown in Figure 4, having these two options naturally enlarges the design space, but what is more interesting is that it may also recover some of the accuracy lost from pruning the original CNN layers (backbone). Figure 5(a)–(d) present the average accuracy (left y-axes) and latency (right y-axes) versus pruning rate (x-axes) on the CIFAR-10 dataset under four Confidence Thresholds (C.T. = 5%, 25%, 50% and 75%). Figure 5(e) plots the FPGA resource utilization. In *Pruned Exits* across all plots, the additional convolutional layers for the early-exits are pruned at the same rate of the backbone. In *Not Pruned Exits*, the exits are not pruned when added to the CNN backbone.

In terms of accuracy, not pruning early exits recovers some of the accuracy for the more heavily pruned models especially at lower confidence thresholds (5 and 25% thresholds in plots (a) and (b)). The reason for this is twofold. First, note that the layers in the pruned backbone are larger than the not pruned exit layers. As the backbone layers get more heavily pruned, their accuracy drops quicker than that of the early exits. Second, the confidence threshold also plays an important role in such scenarios. As the confidence threshold is lowered, more inputs get classified at earlier exits, increasing their impact in the overall accuracy/latency when the full test set is considered. When considering latency, similar reasoning holds for combinations of high pruning rates and low confidence thresholds. For example, from around 50% pruning onwards in plot (a), where the low confidence threshold causes more inputs to exit early, we see faster inferences.

In summary, in scenarios where a high pruning rate is needed to, for example, achieve high throughput, not pruning early exits may help recover some of the accuracy lost in the backbone. In high-accuracy scenarios, in turn, lightly pruned CNNs can be used with a high confidence threshold, causing the last (backbone) exit to be in charge of classifying most inputs. In this case, early exits can be pruned more aggressively, reducing resource usage without significant costs in accuracy.

FPGA Resource Utilization. Figure 5(e) plots the BRAM, LUT, and FF resource usage w.r.t. pruning rates for "Pruned" and "Not Pruned" early-exit CNNs. Remember that the confidence threshold is simply used to accept output vectors and thus it does not change any hardware configuration. For this reason, the resource plot is valid for all confidence thresholds. As can be seen, there is no significant difference in resource usage when comparing pruned and not-pruned early exits for lightly pruned CNNs (from 0 up to 20% pruning rates). This is due to the reduced contribution of early-exits to the total resource usage of such large models. For example, for the not-pruned early-exit CNN (0% pruning rate), exits correspond to 15.25%, 22.58%, and 30% of the allocated BRAMs, LUTs, and FFs, respectively. On the other hand, for the CNN pruned at 85% the exits represent 45%, 28.38%, and 30.82% of the accelerator BRAM, LUT, and FFs. Meaning that as the pruning rate increases, the exits impact on the total resource usage rises and the cost of the "Not Pruned" (purple curves) exits become clearer in contrast with their pruned counterparts (green curves). This is most noticeable for the BRAM usage as this type of resource is primarily used to implement FIFOs to store intermediate data (i.e., feature maps), which are larger within the not pruned exits.

Wrap-Up. We see that combining pruning and early-exit leads to highly heterogeneous design space. From Figures 4 and 5, it is clear that there are many different combinations of pruning rate and confidence threshold, resulting in different

Table I
AVERAGED INFERENCE LOSS, ACCURACY, LATENCY, AND POWER OVER THE FULL 25 SECONDS RUN.

	Dataset	Infer. Loss[%]	Accuracy[%]	Power[W]	Latency[ms]
AdaPEx	CIFAR-10	0.00	80.15	1.26	3.52
	GTSRB	0.00	68.80	1.31	3.04
PR-Only	CIFAR-10	11.82	85.72	1.13	4.37
	GTSRB	0.00	65.38	1.09	3.79
CT-Only	CIFAR-10	12.58	86.57	1.35	4.38
	GTSRB	14.01	66.09	1.37	3.63
FINN	CIFAR-10	22.80	88.74	1.16	5.19
	GTSRB	23.60	70.04	1.14	5.21

accuracy, performance, and energy profiles. The challenge is how to tap into this potential with an effective dynamic approach at runtime. In the following section, we evaluate how AdaPEx achieves this.

B. AdaPEx at the Edge

Table I summarizes the evaluation on the CIFAR-10 and GTSRB datasets. It presents the rate of lost inference requests (Infer. Loss), accuracy, power, and latency results averaged over all executions (25s each). Results show that AdaPEx delivers the best performance across both datasets, reporting no inference loss, meaning 1.31× and 1.32× increase in the number of processed inferences over the original FINN accelerator on CIFAR-10 and GTSRB datasets, respectively. AdaPEx also processes inference requests at latency 1.48× and 1.72× lower than FINN on CIFAR-10 and GTSRB. The increased performance comes at a moderate accuracy cost. AdaPEx processed inferences with accuracy 8.59% below the original CNN (running on FINN) on CIFAR-10 and only 1.24% below on the GTSRB dataset. Recall that this cost is controlled by the user through the accuracy threshold, which was set to 10% in our evaluations. It is also worth noting that early-exit brings power costs due to the extra circuitry required by the additional layers, see AdaPEx or CT-only power in Table I. For example, when comparing the two baselines running not-pruned CNNs (FINN and CT-Only), we see that adding the early exits incurs in 16.09% and 19.63% power overhead, for CIFAR-10 and GTSRB datasets, respectively. Nevertheless, by smartly selecting pruning rates and confidence thresholds, AdaPEx is able to leverage such larger accelerators to deliver inferences of higher quality and energy efficiency as discussed next.

Figure 6 plots the Quality of Experience (QoE curves) and the averaged Energy-Delay Product (EDP) w.r.t original FINN accelerator (bars) averaged over all executions. With QoE (defined as the product of accuracy by the percentage of processed frames) we can assess the overall inference serving quality by measuring the performance-accuracy trade-off. AdaPEx achieves the highest QoE levels among all baselines, increasing QoE over FINN by 11.72% and 15.27% on the CIFAR-10 and GTSRB datasets, respectively. This is because the accuracy loss due to pruning is in part compensated by tuning the confidence threshold (as discussed in Subsection VI-A).

For example, considering the first run on the GTSRB dataset, AdaPEx changed the pruning rate four times (between 5, 20, and 30% pruning rates, requiring four FPGA reconfigurations that took 580 ms in total). With these pruning rates, four confidence thresholds (30, 40, 55, and 60%) were used. Such trade-off can only be exploited by AdaPEx since it simultaneously searches for the best match between pruning rate and confidence threshold.

Regarding energy efficiency, AdaPEx shows also a highly positive impact. Figure 6 bars show that AdaPEx reduces the average EDP in 2× on CIFAR-10 and 2.55× on GTSRB w.r.t original FINN accelerator. In general lines, by selecting pruned

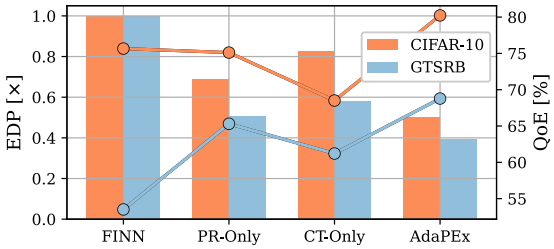

Figure 6. Average EDP normalized w.r.t original FINN accelerator (bars) and QoE (curves) for CIFAR-10 and GTSRB datasets.

early-exit CNNs, AdaPEx gets the best from both optimizations, resulting in a overall efficiency higher than the baselines pruning only (PR-Only that cannot exploit easy images to lower the latency) or early exiting only (CT-Only that cannot alleviate some the power cost of large models).

VII. CONCLUSIONS

We showed that combining pruning and early-exit enlarges the design space; and, that with a simple, and yet effective, adaptation mechanism these optimizations can be used dynamically. The approach grants AdaPEx inferences of higher quality (up to 15% higher QoE) and energy efficiency (up to 2.55× lower EDP) compared to a state-of-the-art CNN accelerator.

ACKNOWLEDGMENTS

This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), São Paulo Research Foundation (FAPESP) grant #2021/06825-8, FAPERGS, CNPq, and by the AI competence center ScaDS.AI Dresden/Leipzig in Germany (01IS18026A-D).

REFERENCES

[1] B. Fang *et al.*, "FlexDNN: Input-Adaptive On-Device Deep Learning for Efficient Mobile Vision," in *SEC*. IEEE, 2020, pp. 84–95.
[2] Z. Xu *et al.*, "Reform: Static and dynamic resource-aware DNN reconfiguration framework for mobile device," in *DAC*. ACM, 2019.
[3] J. Kim, R. M. Bradford, and Z. Shao, "Anytimenet: Controlling time-quality tradeoffs in deep neural network architectures," in *DATE*. IEEE, 2020, pp. 945–950.
[4] H. Li *et al.*, "Pruning filters for efficient convnets," in *ICLR*. OpenReview.net, 2017.
[5] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *ICPR*. IEEE, 2016, pp. 2464–2469.
[6] M. Blott *et al.*, "Finn-*R*: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM TRETS*, vol. 11, no. 3, pp. 16:1–16:23, 2018.
[7] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *DATE*. IEEE, 2016, pp. 475–480.
[8] H. Ting *et al.*, "Dynamic sharing in multi-accelerators of neural networks on an FPGA edge device," in *ASAP*. IEEE, 2020.
[9] G. Korol *et al.*, "Synergistically exploiting cnn pruning and hls versioning for adaptive inference on multi-fpgas at the edge," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, 2021.
[10] ——, "Adaflow: A framework for adaptive dataflow CNN acceleration on fpgas," in *DATE*. IEEE, 2022, pp. 244–249.
[11] C. Hao *et al.*, "FPGA/DNN co-design: An efficient design methodology for iot intelligence on the edge," in *DAC*, 2019.
[12] Y. Chen *et al.*, "Cloud-dnn: An open framework for mapping DNN models to cloud fpgas," in *FPGA*. ACM, 2019, pp. 73–82.
[13] J. Faraone *et al.*, "Customizing low-precision deep neural networks for fpgas," in *FPL*, 2018.
[14] S. Laskaridis, S. I. Venieris *et al.*, "HAPI: hardware-aware progressive inference," in *ICCAD*. IEEE, 2020, pp. 91:1–91:9.
[15] M. Farhadi, M. Ghasemi, and Y. Yang, "A novel design of adaptive and hierarchical convolutional neural networks using partial reconfiguration on FPGA," in *HPEC*. IEEE, 2019, pp. 1–7.
[16] M. Wang *et al.*, "Dynexit: A dynamic early-exit strategy for deep residual networks," in *SiPS*. IEEE, 2019, pp. 178–183.
[17] V. J. Reddi, C. Cheng, D. Kanter *et al.*, "Mlperf inference benchmark," in *ISCA*. IEEE, 2020, pp. 446–459.