EXPRESS: CNN EXecution Time PREdiction for DPU DeSign Space Exploration

***Abstract*—Many accelerators, such as the Deep learning Processor Unit (DPU) from Xilinx, have been proposed to improve the execution speed and/or reduce energy consumption of Convolutional Neural Networks (CNNs) on embedded systems. DPUs can execute a variety of CNNs and are highly configurable at design-time in terms of total parallel compute units, bus configuration for data access, and the number of DPU instances. These configuration parameters determine the execution time of the CNNs implemented on DPUs. Our work is motivated by applications that need to deploy a variety of CNNs for different tasks, and their individual performance requirements could change dynamically. In such a setting, an execution time predictor can help "optimize" the DPU configurations to meet the performance requirements of different tasks. We propose EXPRESS, which uses a careful combination of parameters associated with CNNs, DPUs, and their bus configurations, and predicts the execution time of any given CNN on a DPU. As DPU is invoked by a host CPU to process a CNN layer by layer, EXPRESS considers the CPU and the DPU execution time for predicting the end-to-end processing time. It has been developed through executing a variety of CNN workloads on a real FPGA device. Our experiments across different CNNs, DPUs, and bus configurations indicate that EXPRESS has an average prediction error of 2.2% and significantly outperforms state-of-the-art. We also show its usefulness for the design space exploration of an application involving concurrent CNN execution.**

I. Introduction

Convolutional Neural Networks (CNNs) are gaining popularity in embedded systems due to their high accuracy in various classification and detection tasks [1], [2]. Continuous improvements in CNNs has resulted in a large variety of them being available for any given inference task [3]. Multiple CNNs might be required to be executed concurrently to achieve different kinds of tasks in many embedded systems [4]–[6]. As an example, in a traffic monitoring system [2], many cameras are present at every intersection and each of them have to detect and read license plates of vehicles. Therefore, multiple instances of an object detection and digit classification CNN has to execute and process various camera feeds. At peak hours, there will be more number of cars, but traveling at less speed. So, a CNN with high accuracy and low frames-per-second (FPS) is a suitable choice. At non-peak hours, vehicles will move at a very high speed, but the road will be mostly empty. Thus, a CNN with high FPS is a suitable choice. Such requirements exist for many other systems such as advanced driver assistance systems (ADAS) [1], [5], [7] where CNN processing speed as well as accuracy is determined based on whether the vehicle is on a highway or within the city; assistive gadgets [8] where the user's walking speed and location affect the processing rate. Therefore, CNN accelerators are required to support concurrent CNN execution and switching between CNNs at the run-time [5].

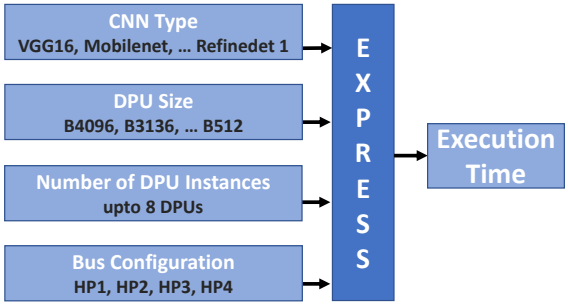

Fig. 1. Different configurable options for DPUs

Among various computation platforms, Field Programmable Gate Arrays (FPGAs) are able to support high performance-per-watt and hence are popularly used in the embedded systems [9], [10]. Xilinx's Deep learning Processor Unit (DPU) [11] is a generic CNN processor (or accelerator) having the following configurable and programmable options (Fig. 1).

i) During the design-time, a DPU can be parameterized for many different sizes that differ in their processing speed and the amount of FPGA resources required to implement them.

ii) The number of DPU instances is also configurable (limited by resources available in the chosen FPGA chip), and multiple DPUs can be active together.

iii) The connection between DPU buses and system-level ports connecting to external memory is also configurable, and different connections lead to different CNN execution times.

iv) At the run-time, a DPU can be programmed to execute any CNN through an instruction file. Note, different CNNs differ in their execution time and energy consumption, and support different inference accuracy.

Given the number of choices for the DPU sizes (n), number of DPU instances (m), count of buses (p), and types of CNNs (k), the total number of design choices becomes very large ($k * p^m * n^m$). Choosing a suitable design point from this vast space requires a quick prediction of execution time for various choices. We propose *EXPRESS (CNN EXecution time PREdiction for DPU deSign Space exploration)* to facilitate exploration of such a large design space. EXPRESS uses machine learning techniques with easy-to-obtain features of CNNs, DPUs, and buses to predict the execution time of various instances of a CNN executing on various DPU design-time configurations (size, count, and bus connections). Additionally, since a DPU is invoked by a host (on-chip) CPU

TABLE I
VARIOUS CNNS AND THEIR SHORT NAMES USED

S.No.	CNN name	Short name	S.No.	CNN name	Short name
1	squeezenet	*sqz*	9	inception_v1	*inc1*
2	resnet18	*res18*	10	inception_v2	*inc2*
3	ssd_traffic	*traf*	11	refinedet_3	*rdet3*
4	ssd_adas	*adas*	12	ssd_pedestrian	*ped*
5	resnet50	*res50*	13	ssd_mobilenetv2	*mossd*
6	refinedet_1	*rdet1*	14	mobilenet_v2	*mob2*
7	vgg16	*vgg*	15	refinedet_2	*rdet2*
8	yolo_v3	*yolo*	16	inception_v3	*inc3*

for processing each CNN layer, EXPRESS also considers the CPU time for predicting the end-to-end CNN processing time. While such models for predicting CNN execution time have been proposed earlier, many of them do not consider concurrent CNN execution [12], [13]. A closely related work, INFER [14], considers a significantly restricted set of DPU configurations. It supports upto only 3 DPU instances and does not consider the performance variation associated with different bus configurations as well as the effect of CPU time in the prediction of total time.

We perform a detailed experimental evaluation using 16 standard CNNs (Table I), 8 DPU configurations (Table II), upto 8 DPU instances, and various bus configurations, for which EXPRESS predicts the execution time with an average error of only 2.2%. Our experiments also indicate that for the expanded design space considered in our work, our prediction error is significantly lower than state-of-the-art [14]. In summary, following are the key contributions of this paper:

1) For the first time, we extend Xilinx tools to implement more than four DPU instances as well as customize the bus interconnections. This has vastly expanded the design space in terms of DPU instances, bus configurations, DPU sizes, and CNNs which in turn has enabled us to identify different factors that affect the execution time on DPUs.
2) We propose EXPRESS, a framework to predict the inference latency of CNNs executing concurrently on multiple DPUs. EXPRESS is the first work to predict DPU execution time considering various bus configurations, a larger number of DPU instances, and the CPU time needed for invoking a DPU.
3) We validate EXPRESS using real data from ZCU102 board and obtain a low error rate (2.2%), with the model being robust to changes in FPGA frequency and the training data set. We illustrate a design space exploration use-case where the feasible points identified using actual measurements are very close to that using prediction.

II. RELATED WORK

Many FPGA-based accelerators for CNNs have been proposed [10], [15], [16]. Unlike these works which focus on specific types of CNNs, Xilinx DPU [11] and Google TPU [13] are generic CNN accelerators that can execute any given CNN through a simple software compilation. Many research works have proposed prediction of execution time or energy consumption of CNNs on various embedded platforms like CPUs, GPUs, TPUs, and FPGAs [12]–[14], [17]–[20]. PETET framework [13] predicts the execution time of a CNN executing on a Google Edge TPU. They validate their model using only two different types of CNNs namely *vgg* and *mobilenet*. nn-Meter framework [12] predicts CNN execution time for various edge devices like mobile CPU, mobile GPU, and Intel VPU. Qiu et al. [18] use analytical modeling and internal architecture details of a DPU to predict the CNN execution time. A key limitation of these works compared to EXPRESS is that they primarily consider either only a single CNN or use internal design details for prediction. None of them consider the effect of bus/interconnects between the accelerator and external memory which becomes a critical factor when multiple instances execute concurrently. Furthermore, they ignore the time to invoke an accelerator by the host CPU.

f-CNNx [21] is one of the first works to consider multi-CNN execution on FPGA platforms and uses execution time prediction for making design decisions. Due to the complete flexibility and visibility in datapath design, they use a simple prediction model with total execution time being the sum of computation and data access time. In contrast, EXPRESS uses generic CNN accelerators like DPU where resource utilization as well as the amount of overlap between computation and memory access can vary significantly for different CNNs. Moreover, f-CNNx does not consider any bus-based contention which significantly affects DPU execution time.

The closest to our proposed work is the INFER framework [14]. Their research problem is the same as ours, namely "predicting runtime for CNN inferences on Xilinx DPU". Their methodology is also ML based modeling of empirical measurement data for training some CNN and DPU configuration combinations, and prediction for unseen combinations. However, INFER [14] had two serious short-comings that affected prediction accuracy – (a) not considering bus configurations that affect data transfer latency between DPU and external memory, increasing CNN runtime by upto 83% for the same CNN-DPU combination and (b) not considering the host CPU's execution time for invoking the DPU, which again can increase CNN execution time by more than 100%. More importantly, through non-trivial engineering modifications, we extend the Xilinx provided official capability of implementing only 4 concurrent DPUs at a time, to the maximum possible number of upto 8 concurrent DPUs on ZCU102. We also extend our prediction framework to handle this additional concurrency which significantly increases the design space, not addressed in any prior work. We explicitly model bus and CPU execution time in our work, improving prediction accuracy by upto 10.8% (2.1% on average) over INFER [14].

III. DESIGN-TIME CONFIGURATION PARAMETERS OF DPU

A Deep learning Processor Unit (DPU) [11] is a CNN accelerator for Xilinx FPGAs which can execute any CNN. DPUs are configurable in eight different sizes which are referred to as B512, B800, B1024, B1152, B1600, B2304, B3136, and B4096; where the suffix number represents the count of parallel processing Multiply-Accumulate (MAC) units in

TABLE II
MAX. NUMBER OF DPUS OF DIFFERENT SIZES FEASIBLE ON ZCU102

DPU Configuration	Max. number of DPUs instantiated	Notation used
B512	8	B512_8
B800	7	B800_7
B1024	6	B1024_6
B1152	6	B1152_6
B1600	5	B1600_5
B2304	4	B2304_4
B3136	3	B3136_3
B4096	3	B4096_3

the DPU. Due to the difference in the number of concurrent MAC units, different DPUs require different amount of FPGA resources and also consume different execution times for the same CNN. Further, each DPU has multiple buses which can connect to various system-level ports available in FPGA chips in different combinations, and affect the overall data transfer time. Thus, the size of DPUs, number of instances, and the system-level bus configuration constitute various design-time configurable parameters for a DPU-based system and are described in more detail.

A. *Size and count of DPUs*

Smaller DPUs (e.g., B512) use less hardware resources as compared to the larger DPUs (e.g., B4096). Thus, a larger count of smaller DPUs can fit on a given FPGA as compared to larger DPUs. However, existing Xilinx tools support a maximum of only 4 DPUs on any FPGA board. We modified the provided configuration files and the driver code to enable the support for larger number of instances. Table II shows the maximum number of DPUs of different sizes that can be instantiated on our evaluation board, ZCU102 [22], featuring a Xilinx Zynq Ultrascale+ FPGA. While referring to different count of DPU instances, we use a notation BX_Y (e.g., B4096_3, B512_8) where X refers to the parallel MAC operations in the chosen DPU and Y refers to the number of DPU instances.

Fig. 2 shows the supported processing speed in frames-per-second (FPS) for various CNNs and DPU configurations. We observe a significant variation in the FPS supported by different DPU configurations. For most of the CNNs (e.g., *yolo*, *rdet1*), B4096_3 configuration provides the maximum FPS. However, for CNNs with lower compute/memory requirements (e.g., *mob2*), B1152_6 configuration provides the maximum FPS. This establishes that different CNNs might achieve their highest FPS under different DPU configurations for the same FPGA board. Additionally, depending on the input FPS requirement, it is possible that smaller DPUs or fewer DPU instances, or both, will suffice. Configurable DPU instances and sizes become advantageous in such cases, as they save FPGA resources and reduce power/energy usage. Remaining FPGA resources could be utilized for other tasks.

B. *BUS configurations of DPU*

Each DPU has three distinct buses – an instruction bus (I) and two data buses (D0 and D1). The instruction bus is used to transfer the instructions corresponding to the CNN that needs

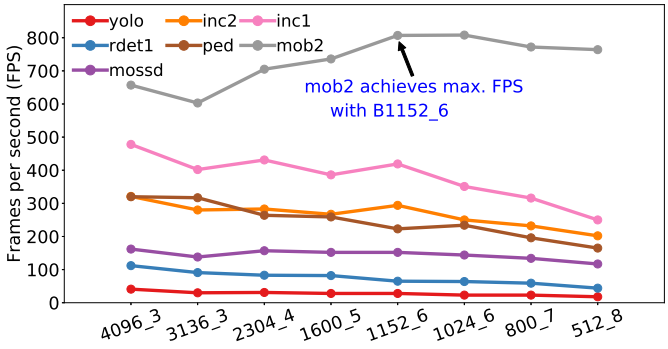

Fig. 2. FPS of different CNNs for different DPU configurations

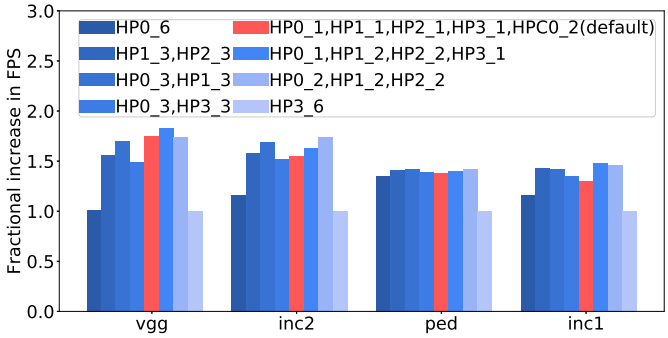

Fig. 3. Fractional increase in FPS of different CNNs for different bus configurations of B4096_3

to be executed on the DPU while the data buses transfer weights, input features, intermediate data, and output data required for CNN execution. Zynq Ultrascale+ FPGA used in ZCU102 board has six different ports named HP0, HP1, HP2, HP3, HPC0, and HPC1 that are available to connect the three DPU buses from each DPU instance. The connection between DPU buses and system-level ports is fully configurable at the design-time. Among the six available ports, HP0–HP3 support high-speed data transfer while HPC0 and HPC1 ports incur coherence overheads and are slower. Since the instruction bus in DPUs requires considerably lower bandwidth than data buses, we combine and connect instruction buses of all DPUs together to one of the HPC ports. The remaining four ports (HP0 – HP3) are connected to the two data buses (D0 and D1) of DPUs and used to access data from external memory (DRAM). For larger count of DPUs, the number of data buses could be much larger than available ports and hence different configurations could arise. For example, B4096_3 has 3 DPUs and thus a total of 6 data buses are required to be connected to four HPx ports. Configurations with smaller DPUs like B512_8 have even larger number of data buses (16), which imply connecting 16 buses to four available ports. These connections could be made in many different ways and they would result in different execution times for the DPUs.

Fig. 3 shows the fractional increase in FPS of various standard CNNs (Table I) for different bus connections for the B4096_3 DPU configuration. Here, we use the notation HPx_y where x indicate the HP port number (0, 1, 2, or 3) and y indicate the count of DPU buses connected to it. For example,

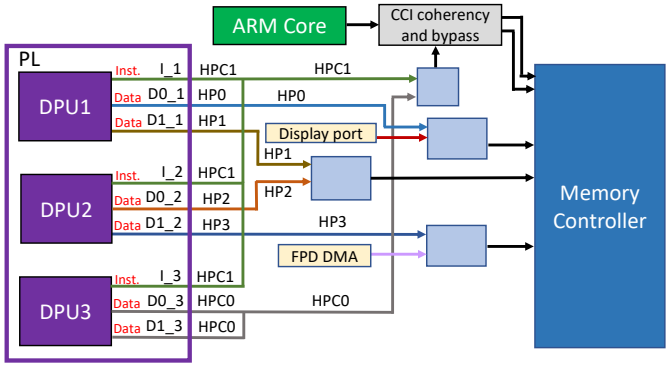

Fig. 4. Default bus configuration for DPU B4096_3

HP0_6 indicates that all the 6 data buses of 3 DPUs are connected to HP0, while {HP0_2, HP1_2, HP2_2} indicates that 2 data buses are connected to HP0, HP1, and HP2 each. The default bus configuration implemented in Xilinx tools is shown in Fig. 4. We observe up to 83% variation in FPS for different bus configurations, which emphasizes that bus configuration should be considered as a part of the design space for estimating execution time. We also observe that the default bus configuration is up to 18% slower than the best configuration. As shown in Fig. 4, the HP3 port gets multiplexed with a Direct Memory Access (DMA) controller and hence supports a lower FPS compared to other ports, with HP3_6 having the lowest FPS. This motivates us to develop an execution time estimation framework considering the size of DPUs, number of DPU instances, bus configuration, and the CNN as configurable parameters so that an appropriate configuration can be chosen at the design-time.

C. Host CPU time for invoking DPU

CNN execution on a DPU is invoked by the host CPU. The host CPU prepares the data (input image frame as well as the CNN information) for processing and then signals the DPU to start the computation. We refer to this processing time as *Interframe time* as it is incurred whenever a new input frame is to be processed. While the data preparation occurs only once, the DPU needs to be triggered for each layer. Upon an interrupt from DPU on completion of a layer, the host CPU triggers the DPU for the next layer until all layers have finished execution. This handshake between the DPU and CPU for each layer consumes a small amount of time, which we refer to as *Interlayer time* and the total Interlayer time for a CNN is the sum of Interlayer time of all its layers. We consider the sum of Interframe time and total Interlayer time as the total CPU time, and the overall CNN execution time is the sum of total CPU time and DPU time. Fig. 5 shows the CPU and DPU execution times for different CNNs for the B4096_1 DPU configuration. While the CPU execution time for larger CNNs like *vgg* and *rdet1* is very small compared to their DPU execution time, it is a considerable fraction of the DPU execution time for smaller CNNs like *sqz* and *inc1* and cannot be ignored. This motivates that CPU execution time must be considered when predicting the execution time of a CNN on a DPU-based system.

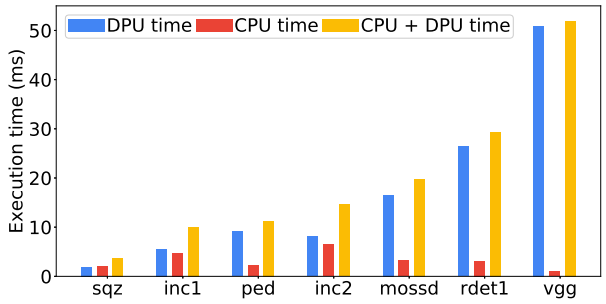

Fig. 5. CPU and DPU execution time for different CNNs

IV. Proposed Methodology for CNN Execution Time Prediction

The CNN execution time is computed as a sum of DPU time and the CPU time, which are individually explained further.

A. DPU time prediction

EXPRESS predicts the execution time (FPS) for four different configurable parameters in a DPU-based system i.e., CNN being executed, DPU size, DPU count, and bus configuration. In order to support these parameters during prediction, we consider the following three categories of features. We use subscript i and j in the notation where i represents the CNN and j represents the DPU configuration.

1) CNN related features: We consider that the execution time ($Time(i,j)$) of a given CNN (i) on a single DPU (j) is already known, which could be either based on measurement on the board or predicted using prior works like INFER [14]. $Time(i,j)$ corresponds to the {HP0_1, HP1_1} bus configuration where D0 and D1 buses of DPU are connected to HP0 and HP1 ports respectively. This bus configuration is chosen as it provides the lowest execution time for all CNNs. We use $Time(i,j)$, $MBpl(i)$, $BW(i,j)$, and $N\_layers(i)$ as features which are representative of these characteristics of CNNs. $MBpl(i)$ is the amount of data required per layer of a CNN. $BW(i,j)$ is the bandwidth requirement of a CNN running on a single DPU. $MBpl(i)$ and $BW(i,j)$ are calculated as:

$$MBpl(i) = \frac{Data(i)}{N\_layers(i)} \qquad BW(i,j) = \frac{Data(i)}{Time(i,j)}$$

Here, *Data(i)* is the total data required for computation of a CNN. $N\_layers(i)$ is the total number of layers present in the given CNN. We also use $N\_layers(i)$ as a separate feature for our predictor. We observe that using it as a separate feature reduces the mean error from 2.6% to 2.4% and the maximum error reduces from 37.7% to 33.6%.

2) DPU related features: As discussed in Section III-A, DPUs can be configured for their size and count at the design-time. We consider the following DPU related features in our framework: (i) N_DPU indicating the count of DPUs used, (ii) $DPU\_Comp(j)$ indicating the number of concurrent MAC operations supported in the DPU j, and (iii) $DPU\_Mem(j)$ indicating the amount of on-chip memory present in the DPU j. Previous works [14] assumed $N\_DPU = 3$ and thus supported limited configurations while we support larger values for N_DPU.

TABLE III
ERROR (%) FOR DIFFERENT APPROACHES FOR PREDICTION OF DPU EXECUTION TIME

Approaches	Mean	Max	90th	75th
Approach-1	24.7	136.7	49.3	34.0
Approach-2	11.5	91.7	26.7	15.8
Approach-3	3.2	64.8	8.5	3.4
Approach-4	**2.4**	**33.6**	**6.3**	**2.7**

TABLE IV
CORRELATION OF FEATURES CONSIDERED FOR PREDICTION OF INTERFRAME CPU TIME

Features correlated	Correlation factor
Number of layers of CNN	**0.99**
MAC operations	0.28
Total data required for computation	0.19
Data required per layer	-0.25
MAC operations per layer	-0.20

3) Bus related features: Since buses in different DPU configurations can connect in many different ways to the four system-level ports (HP0–HP3) which affect their overall execution time, we consider four different features corresponding to each of these ports. We use $HP0\_N$, $HP1\_N$, $HP2\_N$, and $HP3\_N$ as the four features which indicate the count of DPU buses connected to HP0, HP1, HP2, and HP3 ports respectively. Rather than considering the total count of buses for each port, we could have alternatively considered more detailed features indicating the connection of each bus of the DPU. But our experiments suggest that the proposed set of features are easier to obtain and provide similar accuracy. Previous works [14] did not consider bus related features and suffer from higher inaccuracies (Section V-B2).

Using these features, we explored the following four different approaches for developing the prediction model:

- Approach-1: We predict the execution time for multiple instances of CNNs running on multiple DPUs using various features without considering the measured $Time(i,j)$ as an input feature.
- Approach-2: We consider that the single CNN execution time ($Time(i,j)$) is available as an input feature and used along with other features to predict the execution time for multiple instances of a CNN running on multiple DPUs.
- Approach-3: Building up on approach-2, rather than predicting the absolute execution time for multiple instances, we predict the increase in execution time compared to a single DPU, and then add it to the single CNN execution time to obtain the execution time for multiple instances.
- **Approach-4:** Modifying approach-3, instead of predicting the absolute increase in the execution time, we predict the percentage increase in execution time over the single CNN/DPU scenario and use it to predict the execution time for multiple instances.

We evaluate these approaches using various regression based prediction techniques like random forest, polynomial, and decision tree. As shown in Table III, Approach-4 provides the lowest error among these and hence adopted as the final approach in EXPRESS for prediction of DPU time.

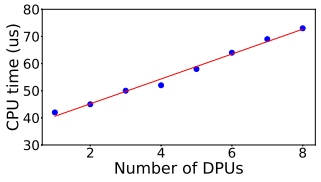

Fig. 6. Interlayer CPU time versus number of DPUs

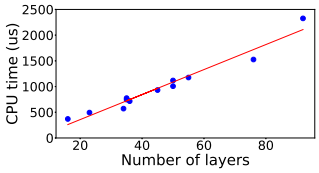

Fig. 7. Interframe CPU time versus number of CNN layers

B. CPU time prediction

The total CPU time comprises of Interframe and Interlayer time, which are determined as follows.

1) Interlayer CPU time : Using a large number of measurements for different CNNs, and DPU/bus configurations on the hardware board, we observe that the Interlayer CPU time remains almost constant for various combinations of CNN, DPU size, and bus configuration and varies only with the count of DPUs used. As shown in Fig. 6, Interlayer time has an almost linear relationship to the number of DPUs used and thus, we use a simple linear regression model with number of DPUs as an input feature to predict the Interlayer CPU time.

2) Interframe CPU time : We measure Interframe CPU time for various CNNs, and DPU/bus configurations and observe that the Interframe CPU time is not affected by the number of DPU cores, DPU type, or bus configuration but depends primarily on the CNN to be executed. We further evaluate different CNN related features for their effect on the Interframe CPU time (Table IV) and observe that the number of layers in a CNN have a very high correlation factor with the Interframe CPU time. Further, as Fig. 7 shows, Interframe CPU time has a linear relationship with the number of CNN layers. Therefore, we use a simple linear regression model with the number of CNN layers as an input feature to predict the Interframe CPU time.

When number of DPUs significantly exceed the number of CPUs, we see a rise in tail latency in Interframe and Interlayer CPU times, as more DPUs get assigned to the same CPU core. This might adversely affect some real time applications. We will investigate this further in future, possibly designing our own CPU scheduler to reduce unpredictable interference.

V. EXPERIMENTS AND RESULTS

A. Experimental Setup

1) Setup: We use Xilinx Zynq Ultrascale+ FPGA board (ZCU102) [22], Vitis AI v1.3 framework [23], and DPU v3.3 [11] for our experimentation. We consider eight different DPU sizes and upto eight DPU instances, as listed in Table II. Further, as discussed earlier in Section III-B, four HPx ports can be used in various different ways to connect to the data bus of each DPU instance, leading to many different bus configurations. We generate a large number of configurations in the form of bitstreams corresponding to various bus configurations, for each of the DPU configurations. We evaluate EXPRESS using 16 standard CNNs (Table I) differing in their compute and memory requirements. Table V shows the train and test data used in EXPRESS. We choose 8 CNNs for training data

TABLE V
TRAIN AND TEST DATA FOR EXPRESS

DPUs → / **CNNs ↓**	DPU config.	No. of DPUs	DPU config.	No. of DPUs
	B512	1-8	B800	1-7
	B1024	1-6	B1152	1-6
	B2304	1-4	B1600	1-5
	B4096	1-3	B3136	1-3
sqz, ped, mob2, rdet1, inc1, yolo, vgg, res50	**Quadrant-1** **Train data**		**Quadrant-2** **Test data**	
res18, traf, adas, rdet2, rdet3, inc2, inc3, mossd	**Quadrant-3** **Test data**		**Quadrant-4** **Test data**	

and the remaining 8 CNNs for test data. We choose the largest (B4096), smallest (B512), and two medium sized DPUs (B1024 and B2304) for training, while the remaining four DPU sizes are in the test dataset. We include all possible bus configurations for the corresponding DPUs in both train and test dataset. This creates a rich dataset with train and test data size of 1175 and 3765 points respectively.

2) Measurement methodology: We use the *profile* mode execution of DPUs to measure the execution time of a single CNN executing on a single DPU, averaged over 100 different images. When executing multiple CNNs on multiple DPUs, we use *dpuSetTaskAffinity* API to ensure that each CNN is executed on a fixed DPU and different layers of a CNN are not assigned to different DPU instances. We consider that multiple copies of the same CNN is executing on different DPUs. While the CNN and DPU size are same for various instances, use of different bus connections for different DPUs could result in different execution time for each CNN. We follow an approach to first extract the execution time of each CNN on their assigned DPUs, out of which we consider the lowest execution time (T_{low}). Then, we calculate the number of image frames (F_i) processed by each DPU (i) within T_{low}. The average execution time of the CNN per image (T_{avg}) is thus obtained as:

$$T_{avg} = \frac{T_{low}}{\sum_i F_i}$$

We extract the Interlayer and Interframe CPU time by processing the raw profile file created during DPU execution.

3) Prediction model: We explore three widely used prediction techniques, namely random forest, decision tree, and polynomial regression (first and second order), for training the model and predicting the execution time. Random forest provides the lowest percentage error (mean, median, and maximum). This is also in-line with our expectations as random forest is a non-linear predictor composed of several uncorrelated decision trees that outperforms any individual tree [24]. A similar observation about random forest predictor is also reported by prior works [13], [14], and thus, we use random forest based prediction in EXPRESS.

TABLE VI
DPU TIME PREDICTION ERROR (%) FOR DIFFERENT DPU CONFIGURATIONS

DPU config.	Mean	90th	95th	99th	Max
B512	**0.9**	2.0	2.9	9.2	13.7
B800	**2.0**	4.6	6.9	16.8	29.1
B1024	**2.7**	6.1	12.9	28.1	33.6
B1152	**2.6**	6.9	9.7	19.5	33.1
B1600	**3.1**	8.5	13.0	20.8	25.5
B2304	**4.0**	10.8	13.8	21.9	33.1
B3136	**2.8**	6.2	8.9	21.1	26.1
B4096	**3.7**	7.4	12.9	23.5	24.3
All DPUs	**2.4**	6.3	9.9	20.6	33.6

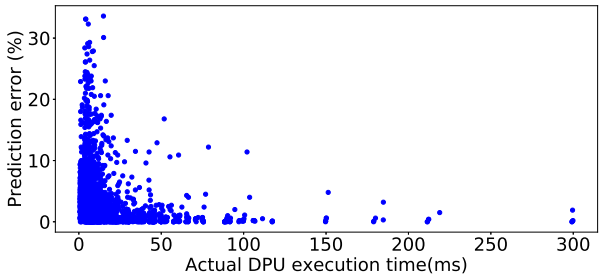

Fig. 8. Prediction error versus the actual DPU execution time for various test points

B. Results

1) Prediction errors: We present the prediction accuracy of EXPRESS for DPU time followed by the total time (CPU+DPU). The test dataset (Quadrants-2, 3, and 4) used for evaluation is shown in Table V. Table VI shows the percentage estimation error of DPU time in comparison to the measured execution time on ZCU102 board for various DPU sizes and various number of DPU instances. The mean error for various DPU sizes is within 4.0%, which is quite low. When considering the entire test dataset together, the overall mean, 90th percentile, and 95th percentile errors are 2.4%, 6.3% and 9.9% respectively. We also observe a significant difference for the 99th percentile and maximum error in comparison to the 95th percentile, which indicates that most of the cases are predicted with low error rate barring a few. The maximum error goes as high as 33.6% due to a few extreme corner cases. For example, if all the DPU buses are connected to the same HP3 port, the execution time will be very high specifically for B512 which can have upto 8 DPU instances. While such cases are very unlikely to occur as the designer might also want to use the other three HPx ports, EXPRESS shows high percentage prediction errors for such cases.

To illustrate further, Fig. 8 shows the prediction error for all points in our test dataset (Table V). The x-axis shows the average execution time of DPU per image, measured and averaged for a batch of 1000 images whereas the y-axis shows the prediction error corresponding to these execution times. We observe that high percentage errors correspond to small actual execution times. When actual execution time is small, a slight absolute error causes a high percentage error. In our dataset, the execution time for various CNNs, DPUs and bus configurations ranges from 0.7 ms to 299.8 ms while the abso-

TABLE VII
PREDICTION ERROR (%) FOR DPU AND CPU EXECUTION TIMES

Different prediction cases	Mean	90th	95th	99th	Max
DPU time only	2.4	6.3	9.9	20.6	33.6
Total time (with act. DPU time)	0.5	1.2	1.7	3.0	5.7
Total time (CPU + DPU)	2.2	5.5	8.8	17.6	32.2

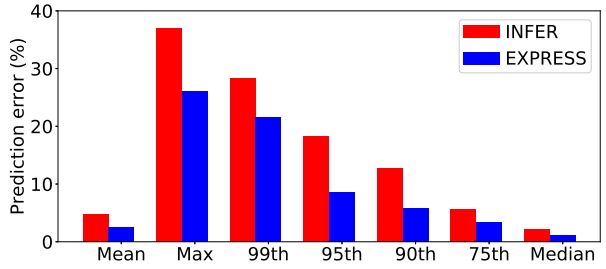

Fig. 9. Prediction error (%) of EXPRESS and INFER [14]

lute prediction error for most of them is below 2.0 ms. These results indicate that EXPRESS predicts execution time for various DPU configurations, CNNs, and bus configurations, with high accuracy.

We now present the prediction error for the total execution time by considering two different scenarios. In one case, we consider the total execution time with CPU time being predicted from our model but combined with actual DPU time obtained from measurements. In the second case, we consider the total time as the sum of predicted times for CPU and DPU. The error rate for these scenarios are shown in row-2 and row-3 of Table VII. Overall, we observe that the total prediction error is slightly lower than predicting only the DPU time, which happens as the denominator is larger when considering the total time. The overall mean, 90th percentile, and 95th percentile errors are 2.2%, 5.5% and 8.8% respectively.

2) Comparison with state-of-the-art: We consider a recent work, INFER [14], which predicts execution time of CNNs on DPUs for a fixed number of DPUs (three) and does not consider bus configuration. It does not take CPU time into account while making predictions and just predicts the DPU time. For a fair comparison between EXPRESS and INFER, we modify our prediction model so that it predicts only the DPU time, considers only the features used by INFER, and uses only 3 DPUs when calculating the error. Further, we also used the measured execution time of a single DPU as an input feature for both EXPRESS and INFER. Fig. 9 shows that EXPRESS significantly outperforms INFER. The mean reduces from 4.7% to 2.5% and the 95th percentile error reduces from 18.2% to 8.5%. This is because INFER performs poorly on non-default bus configurations.

3) Prediction errors with varying train dataset: We conduct a separate experiment to show the effect of varying the train data size on the overall accuracy. In our previous experiments, we considered only the Quadrant-1 from Table V for training which uses only four DPU configurations. Now, we increase the number of DPU configurations used in training set from 2 to 8 and measure the error for a common test dataset (only Quadrant-4 from Table V). As shown in Fig. 10, increasing

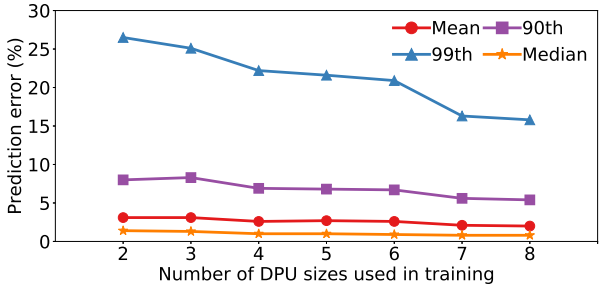

Fig. 10. Prediction error (%) for increasing the number of DPUs in the training

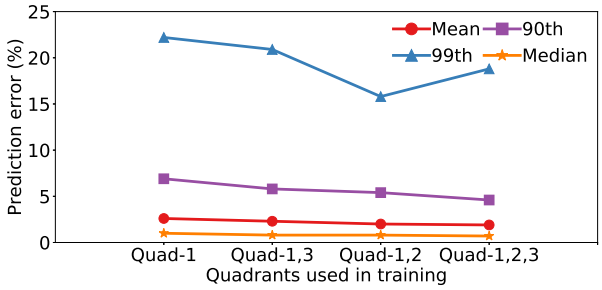

Fig. 11. Prediction error (%) by changing the quadrants used in training

the number of DPU configurations in the training set improves the prediction accuracy of EXPRESS. We now include more quadrants in the training set but consider a common test set (Quadrant-4). Again, as shown in Fig. 11, prediction error decreases as we include more quadrants in the training set. From both these experiments, we observe that our choice of using only Quadrant-1 for training set achieves almost similar 90th percentile error as compared to using three quadrants. The 99th percentile error does show larger improvements with larger training set, but a lot of these cases are not practically useful as mentioned earlier. While one could improve the accuracy by considering a larger training dataset, an estimation model trained on a restricted dataset is a more viable choice given the rapid advancement of CNNs and their accelerators.

4) Cross-validation: In order to show the performance of our framework on unseen data, we perform a 16-fold cross validation of EXPRESS for 16 standard CNNs listed in Table I. We choose 15 CNNs for training and the remaining one as the test set. This experiment is repeated 16 times for each CNN and Fig. 12 shows the prediction error for individual CNNs. We observe a low mean error for all CNNs except *vgg*. This confirms that EXPRESS performs well on unseen data.

vgg shows a higher error because it has a relatively larger fully connected layers, not present in other CNNs. As *vgg* is not considered in training set during cross-validation for *vgg*, such aspects are missed out and the error is higher. However, in the actual training set of EXPRESS captured in Table V, *vgg* is indeed a part of the training set and avoids this situation.

5) Prediction error for different clock frequencies: One key change that can happen when the DPUs get implemented on different FPGA devices is that the frequency of operation may vary. Since we had physical access to only ZCU102 board,

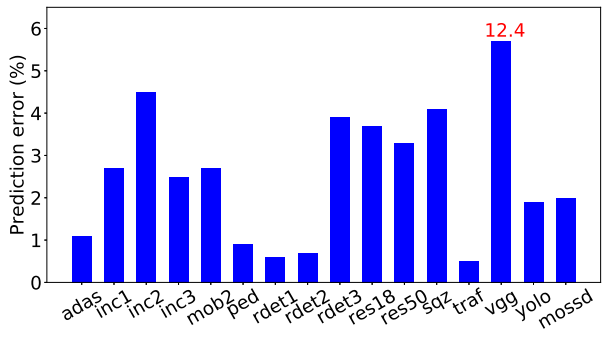

Fig. 12. Prediction error (%) for 16-fold cross validation of EXPRESS

TABLE VIII
FEASIBLE DESIGN POINTS USING ACTUAL (ACT.) AND ESTIMATED (EST.) EXECUTION TIMES

Time limit (ms) –>	**5**	**10**	**20**	**50**	**80**	**100**
Only DPU time (act.)	1349	2850	3942	4603	4783	4845
Only DPU time (est.)	1358	2860	3943	4604	4781	4845
CPU+DPU time (act.)	43	491	1267	3389	4191	4442
CPU+DPU time (est.)	36	482	1295	3407	4208	4437

in order to emulate different FPGA boards, we evaluate EXPRESS to predict CNN execution time on DPUs by changing the clock frequency. We scale the base frequency of our FPGA by 1.0x, 0.93x, 0.75x, and 0.60x and measure the execution time for various CNNs and DPU configurations, for a subset of the bus configurations. We identify that the execution time of different CNNs increase in different proportions as the memory access time does not scale with changing the FPGA frequency. This behavior is indeed a closer match to using another FPGA device and by using multiple frequency scaling factors, we could emulate many devices. We are able to achieve a low prediction error (within 2%) despite such frequency variations and without any retraining of the model because of the use of single DPU execution time as an input feature in EXPRESS. This experiment demonstrates that EXPRESS is robust and can easily be extended for different FPGAs without any need for retraining.

VI. USE OF EXPRESS FOR DESIGN SPACE EXPLORATION

As mentioned in Section I, one of the primary use for a prediction framework like EXPRESS is to enable a fast analysis and exploration of various design choices for a DPU-based system, without the need for actually generating a bitstream. Prior works [5] that address design space exploration (DSE) for DPU-based systems performed extensive measurements to obtain the execution time data. However, it becomes practically prohibitive to perform such measurements due to the many-fold increase in design space due to consideration of bus connections and variable number of DPU instances.

Similar to prior works [5], [6] and as per the needs of targeted applications like drones, ADAS, traffic monitoring, etc. that require CNN execution, we consider that a DPU-based system would execute multiple computer-vision applications concurrently, with a specified periodicity. One of the important aspect of DSE in such cases is to identify whether a given design choice would meet the specific period or not. For example, if an application needs to support a frame rate of 30 FPS, then we could prune out design choices having an execution time larger than 33.33 ms. During such pruning, classifying a design point correctly as feasible or infeasible is more important than the accuracy of the predicted execution time. A design point whose execution time is either very high or very low from the allowed period, might still get correctly classified as feasible/infeasible despite high prediction errors in the underlying model.

We conduct an experiment by considering various values of allowed period in the range of 5 ms to 100 ms and count the number of feasible design points from our entire test dataset, as per actual and predicted execution times. We consider using only DPU time and the total time (CPU+DPU) for checking the feasibility. As shown in Table VIII, the count of feasible points reported by EXPRESS is very close to that identified using actual measurements. When we consider total time (CPU+DPU) rather than just DPU time, the number of feasible design points is significantly reduced, specifically for shorter time periods (e.g., the feasible points reduce from 1349 to 43 for 5 ms period). Such a reduction illustrates the importance of considering CPU time during prediction so as to obtain more realistic design points.

In a similar manner, EXPRESS can also be used to make decisions to switch CNNs dynamically during the run-time. Lei et al. [6] propose to use heterogeneous DPUs to improve system efficiency during concurrent CNN execution and use measured execution time in their analysis. Such a work can significantly benefit from an execution time prediction framework like EXPRESS.

VII. CONCLUSION AND FUTURE WORK

We extend the existing tools to implement larger number of DPU instances and study the effect of various bus configurations on execution time of CNNs. Using these insights, we developed EXPRESS framework to predict the execution time of multiple instances of a CNN executing on DPUs, by considering various design-time configurations. This is the first work to consider variable number of DPU instances, bus connections between DPU and external memory, and host CPU time in such a prediction model. EXPRESS requires easy-to-derive features of CNNs, DPUs, and buses for prediction and hence the model is practically viable. We validate EXPRESS using 16 standard CNNs, 8 DPU sizes, upto 8 DPU instances, and many different bus configurations on a ZCU102 board and achieve a low prediction error of 2.2% on an average. Our detailed experimentation also indicate that EXPRESS has significantly lower error rate compared to state-of-the-art, it is robust to changes in FPGA frequency as well as the training dataset, and it is highly effective for exploring various design choices during system design.

In the future, we plan to support heterogeneous CNNs executing on multiple DPUs. We would also augment our framework to predict power and energy consumption alongside execution time.

REFERENCES

[1] F. Restuccia and A. Biondi, "Time-predictable acceleration of deep neural networks on FPGA SoC platforms," in *2021 IEEE Real-Time Systems Symposium (RTSS)*, 2021.

[2] M. S. Chauhan, A. Singh, M. Khemka, A. Prateek, and R. Sen, "Embedded CNN based vehicle classification and counting in non-laned road traffic," in *International Conference on Information and Communication Technologies and Development (ICTD)*, 2019. [Online]. Available: https://doi.org/10.1145/3287098.3287118

[3] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, 2017.

[4] P. Subedi, J. Hao, I. K. Kim, and L. Ramaswamy, "AI multi-tenancy on edge: Concurrent deep learning model executions and dynamic model placements on edge devices," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, 2021.

[5] R. Kedia, S. Goel, M. Balakrishnan, K. Paul, and R. Sen, "Design space exploration of FPGA-based system with multiple DNN accelerators," *IEEE Embedded Systems Letters*, 2021.

[6] Y. Lei, Q. Deng, S. Long, S. Liu, and S. Oh, "An effective design to improve the efficiency of DPUs on FPGA," in *International Conference on Parallel and Distributed Systems (ICPADS)*, 2020, pp. 206–213.

[7] J. Peng, L. Tian, X. Jia, H. Guo, Y. Xu, D. Xie, H. Luo, Y. Shan, and Y. Wang, "Multi-task ADAS system on FPGA," in *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, March 2019.

[8] R. Kedia, A. Sobti, M. Rungta, S. Chandoliya, A. Soni, A. K. Meena, C. M. Lobo, R. Verma, M. Balakrishnan, and C. Arora, "MAVI: Mobility assistant for visually impaired with optional use of local and cloud resources," in *International Conference on VLSI Design and International Conference on Embedded Systems (VLSID)*, 2019.

[9] M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, "Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels," in *2019 IEEE international conference on embedded software and systems (ICESS)*. IEEE, 2019.

[10] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA-based neural network inference accelerators," *ACM TRETS*, 2019.

[11] "DPU for CNN v3.3," 2019. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_3/pg338-dpu.pdf

[12] L. L. Zhang, S. Han, J. Wei, N. Zheng, T. Cao, Y. Yang, and Y. Liu, "Nn-Meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices," ser. MobiSys, 2021. [Online]. Available: https://doi.org/10.1145/3458864.3467882

[13] Y. Ni, Y. Kim, T. Rosing, and M. Imani, "Online performance and power prediction for edge TPU via comprehensive characterization," in *Proceedings of the 25th Conference on Design, Automation and Test in Europe*, ser. DATE, 2022.

[14] S. Goel, R. Kedia, M. Balakrishnan, and R. Sen, "INFER: Interference-aware estimation of runtime for concurrent CNN execution on DPUs," in *2020 International Conference on Field-Programmable Technology (ICFPT)*, 2020.

[15] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh, "From high-level deep neural models to FPGAs," in *International Symposium on Microarchitecture (MICRO)*, 2016.

[16] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2017. [Online]. Available: https://doi.org/10.1145/3020078.3021744

[17] M. Ferianc, H. Fan, R. S. Chu, J. Stano, and W. Luk, "Improving performance estimation for FPGA-based accelerators for convolutional neural networks," in *ARC*, 2020.

[18] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, and et al., "Going deeper with embedded FPGA platform for convolutional neural network," in *International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2016. [Online]. Available: https://doi.org/10.1145/2847263.2847265

[19] L. Mei, H. Liu, T. Wu, H. E. Sumbul, M. Verhelst, and E. Beigne, "A uniform latency model for DNN accelerators with diverse architectures and dataflows," in *Proceedings of the 25th Conference on Design, Automation and Test in Europe*, ser. DATE, 2022.

[20] G. Zhong, A. Prakash, Y. Liang, T. Mitra, and S. Niar, "Lin-analyzer: A high-level performance analysis tool for FPGA-based accelerators," in *Design Automation Conference (DAC)*, 2016, pp. 1–6.

[21] S. I. Venieris and C. Bouganis, "f-CNNx: A toolflow for mapping multiple convolutional neural networks on FPGAs," in *International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2018, pp. 381–3817.

[22] Xilinx, "Zynq UltraScale+ MPSoC ZCU102 evaluation kit." [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html

[23] "Vitis for CNN v1.3," 2021. [Online]. Available: https://www.xilinx.com/cgi-bin/docs/rdoc?t=vitis_ai;v=1.3;d=ug1414-vitis-ai.pdf

[24] L. Breiman, "Random forests," *Mach. Learn.*, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324