Binary-LoRAX: Low-Latency Runtime Adaptable XNOR Classifier for Semi-Autonomous Grasping with Prosthetic Hands

Nael Fasfous[1], Manoj-Rohit Vemparala[2], Alexander Frickenstein[2], Mohamed Badawy[1], Felix Hundhausen[3], Julian Höfer[3], Naveen-Shankar Nagaraja[2], Christian Unger[2], Hans-Jörg Vögel[2], Jürgen Becker[3], Tamim Asfour[3], Walter Stechele[1]

[1] Technical University of Munich [2] BMW Group [3] Karlsruhe Institute of Technology

***Abstract*— Intelligent, semi-autonomous prostheses take advantage of combining autonomous functions and traditional myoelectric control. With the help of visual and environment sensors, intelligent prostheses achieve a level of autonomy which relieves the user from generating elaborate electromyographic (EMG) signals for grasp type and trajectory. To achieve the desired functionality, the semi-autonomous prosthesis must efficiently process the incoming environmental data at a high rate, with low power and high accuracy. In this paper, we propose Binary-LoRAX, a low-latency runtime adaptable classifier for the semi-autonomous grasping task of prosthetic hands. We offload the classification task to an efficient binary neural network accelerator which performs high-throughput XNOR operations on digital signal processing (DSP) blocks. To tailor the classifier's performance to the current application scenario, we propose a frequency scaling approach which dynamically switches between two modes of operation, high-performance and power-saving. At high-performance, classifications are performed with a low latency of 0.45ms, high-throughput of 4999 FPS and power consumption of ∼2.15 W. This enables functions such as object localization and batch classification. Switching to power-saving mode, a latency of 80 ms is maintained, with up to 19% improved classifier battery-life. Our prototypes achieve a high accuracy of up to 99.82% on a 25 class problem from the YCB graspable object dataset.**

I. Introduction

Computer vision tasks such as image classification [1], object localization [2] and semantic segmentation [3] are fundamental to many applications such as autonomous driving, human-robotic interaction and smart factories. With the abundance of training data and compute resources, deep learning algorithms, such as convolutional neural networks (CNNs), have dominated most computer vision tasks, albeit at the cost of increased memory requirements and compute complexity [4], [5], [6].

The design of low-power, performant intelligent systems emphasizes the importance of an efficient deployment of deep learning algorithms on embedded hardware. Specifically for autonomous applications, including robotic or prosthetic devices, real-time interpretation of sensor data is essential for responsiveness. When visual sensors such as cameras are used, the processing of the high-bandwidth input data is challenging, especially for battery-powered systems. In prosthetic hands, the implementation of semi-autonomous functions is enabled through in-hand visual perception, which requires efficient embedded processing to avoid insecure, high-latency external compute services. The complete control algorithms, including image recognition, must be executed on in-hand embedded processing hardware.

Such contradictory objectives of maximizing performance while minimizing power and resource utilization, assign a decisive role to optimization of deep learning algorithms. One option is to exploit the representational redundancy of neural networks through quantization and binarization [7], [8], [9]. In this work, we propose Binary-LoRAX, an efficient runtime adaptable binary neural network (BNN) classifier for the semi-autonomous grasping task of prosthetic hands. We tackle the challenges arising from the prosthetic hand's application constraints through the following contributions:

- Applying BNNs to the graspable object classification task, enabling the efficient deployment of neural networks on intelligent prostheses with a task-related accuracy of 99.82% on a 25 class problem from the YCB object dataset [10], adding 12 classes compared to existing work [11].
- Achieving low-latency classifications of 0.45 ms, consuming <1% of the optimal controller delay [12] and achieving a 99.7% reduction in latency compared to existing work [11].
- Efficiently executing `XNOR` operations on an FPGA's digital signal processing (DSP) blocks in a vectorized manner, freeing more look-up table (LUT) resources and allowing larger BNNs to fit onto embedded FPGAs.
- Dynamically adapting the frequency of the accelerator, offering high-performance and low-power modes to target different application scenarios (dangerous/delicate objects, batch processing, object localization, low battery, prosthetic movement), improving the classifier's battery-life by up to 19% compared to [13].

II. Related Work

A. Efficient Intelligent Prosthetics

For semi-autonomous control of a prosthetic hand, vision-based perception is an important component which has gained a lot of attention in research lately [14], [15], [16], [17], [18], [19], [20], [21]. The implementation of

This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project Number 146371743 - TRR 89: Invasive Computing.

semi-autonomous hand functions reduces the complexity of required user-control commands typically generated by electromyographic (EMG) signals. Compared to completely manual hand control, the set of required commands can be smaller and invoked with lower cognitive effort and thus lowers the cognitive burden on the user [17], [22]. Moreover, the reduced complexity of commands requires less complex electrode setups that are needed for higher accuracy and more long-term stable EMG-pattern recognition.

The semi-autonomous KIT Prosthetic Hand proposed in [18], employs an in-hand camera to decrease the cognitive burden on the user by automating parts of the grasping process with the help of visual environmental information. For grasping an object with the support of semi-autonomous functions, the first step is to obtain visual object information, such as object class or object dimensions, e.g. width or height. This information is then used to select a suitable grasp from a database, where parameters can include finger trajectories or forces. In [11] a two-step classification system for the KIT Prosthetic Hand is proposed, where an object classification algorithm and an acknowledgment from the user triggers a second segmentation network.

In the first version of the KIT Prosthetic Hand, an ARM Cortex M7-based microprocessor was used. The currently developed design includes a Zynq Z7010-based processing hardware. A photo of the hardware is shown in Fig. 1.

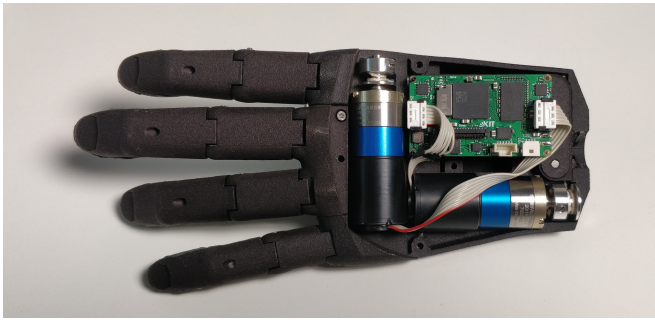

Fig. 1: KIT Prosthetic Hand (50^{th} percentile female) with Zynq Z7010-based processing system

B. Binary Neural Networks

Parameter quantization has a direct impact on a neural network's memory footprint and the complexity of its arithmetic operations on hardware. Binarization is the most aggressive form of quantization, where network weights and activations are constrained to $\{-1, 1\}$ [9]. Theoretically, this leads to a parameter compression of $\times 32$ compared to a float-32 CNN, and allows for an implementation of multiply-accumulate (MAC) operations as simple `XNOR` and `popcount` on inference hardware [23]. As a trade-off, low bitwidth representations have a lower information capacity, losing the precision of the finely adjusted weights achieved by the gradient propagation during training. To address these challenges, specialized training schemes are applied (normalization, straight-through-estimators (STEs), scaling/shifting factors) [24], [25], [23]. Different schemes for binarization have been proposed [9], [26], [27], [28], [23]. Courbariaux et al. [26] introduced the concept of training neural networks with binary weights during the forward pass and maintaining latent full-precision values during back-propagation to allow fine adjustments through the gradients. The authors later augmented this approach with binarized activations [9]. Rastegari et al. [23] introduced XNOR-Net, where the convolutions were approximated by a combination of `XNOR` operations and `popcounts`, followed by a multiplication with scaling factors. The introduction of scaling factors not only increases the number of trainable parameters for each layer, but also adds to the computational complexity of XNOR-Net at deployment time. Although [23], [28], [29] and [30] have focused on adding algorithmic or structural complexity to BNNs to achieve classification performance close to full-precision CNNs on complex tasks, simpler tasks with lower scene complexity can be handled with more efficient forms of BNNs [9], [31].

In the context of semi-autonomous prosthetic hands, the camera input at the instance before the grasp operation takes place is expected to have one central object in the field-of-view. In that regard, the task's complexity resembles that of popular datasets, such as the German Traffic Sign Recognition Benchmark (GTSRB) [32], Street View House Numbers (SVHN) [33] or CIFAR-10 [34], all of which have the object of interest in the forefront of the scene, with minimal, random background complexity when compared to autonomous driving scenes such as Cityscapes [35]. It is important to note that BNNs have shown high accuracy and good generalization on the mentioned datasets [9], [13], making them a worthy candidate for the graspable object classification problem.

C. BNN Hardware Accelerators

Several accelerators have been designed to exploit the benefits of BNNs [36], [37], [13], [38], [39]. FINN [13] is a popular framework for accelerating BNNs on FPGAs. Although the framework is designed for BNNs presented in [9], it also supports 2-bit weights and/or activations. FINN compiles HLS code from a BNN description to create a hardware design for the network. The generated streaming architecture consists of a pipeline of individual hardware components instantiated for each layer of the BNN. OrthrusPE [40] investigates the effectiveness of deploying binary operations onto DSPs as SIMD binary Hadamard product processing units. The authors reconfigure the DSP at runtime to perform either fixed-precision operations or SIMD binary operations, enabling BNNs with scaling factors and multiple bases [28], [23]. In this work, we infuse the FINN architecture with DSPs, by switching them *statically* to a binary operation mode. For LUT constrained devices such as the Z7010, this allows larger and/or faster accelerator designs, by spreading out computations to DSPs. We further enable runtime frequency scaling to achieve different modes of operation, at different latency requirements and power consumption rates, based on the current application scenario.

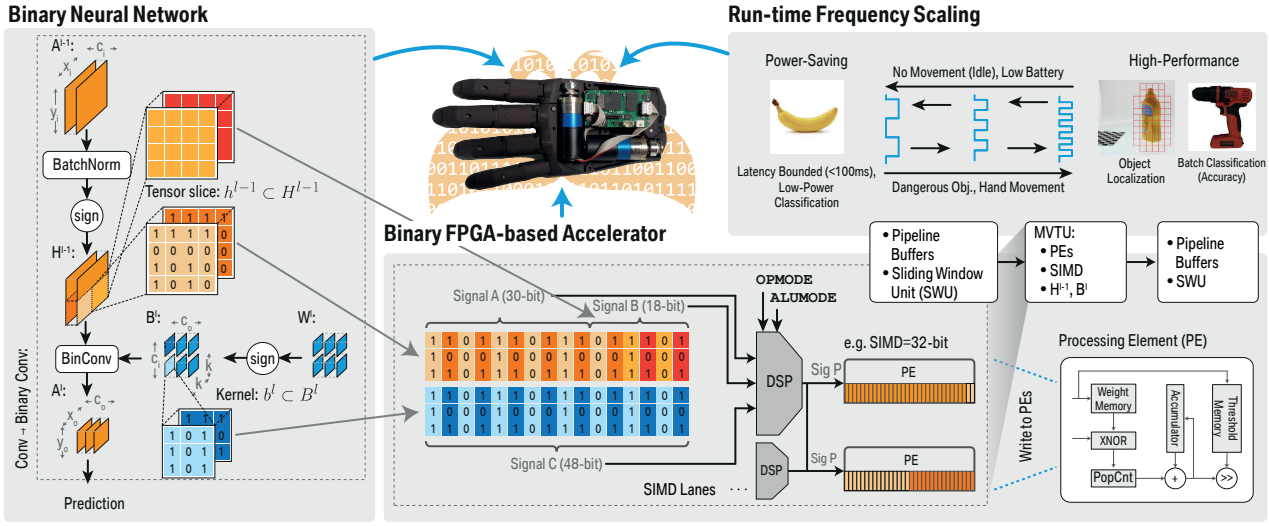

Fig. 2: Overview of Binary-LoRAX: BNN tensor slices are fed into DSPs which perform high-throughput XNOR operations. DSP results are forwarded to the PEs of an MVTU. A single MVTU of the pipeline is shown for compactness. Runtime frequency scaling allows high-performance functions, or power-saving mode.

III. Method

A. Training and Inference of BNNs

For efficient approximation of weights and activations to single-bit precision, the BNN method by Courbariaux et al. [9] is used. At training time, the network parameters are represented by full-precision latent weights W allowing for a smoother convergence of the model [24]. It is important to note that the input and output layers in this implementation are not binarized, to avoid a drop in classification accuracy.

Without loss of generality, the activation feature map $A^{l-1} \in \mathbb{R}^{X_i \times Y_i \times C_i}$ is considered as the input to a convolutional layer $l \in [1, ..., L]$, where X_i, Y_i and C_i describe the dimensions of width, height and input channels. A^0 and A^L are the input image and the prediction of the BNN, respectively. The latent weight matrix $W \in \mathbb{R}^{K \times K \times C_i \times C_o}$ is composed of the trainable parameters of the individual 2D-convolutional layers, where K and C_o are the kernel dimensions and the number of output channels. During the forward-pass for loss calculation or deployment, the weights $w \in W$ are transformed into the binary domain $b \subset B \in \mathbb{B}^{K \times K \times C_i \times C_o}$, where $\mathbb{B} = \{-1, 1\}$. In the hardware implementation, the -1 is represented as 0 to perform multiplications as `XNOR` logic operations. The weight and input feature maps are binarized by the $sign()$ function

$$b = sign(w) = \begin{cases} 1 & \text{if } w \geq 0, \\ -1 & \text{otherwise} \end{cases}. \tag{1}$$

The $sign()$ function blocks the flow of gradients during training due to its derivative, which is zero almost everywhere. To overcome the gradient flow problem, the $sign()$ function is approximated during back-propagation by the straight-through estimator (STE) [24]. In the simplest case, the estimated gradient g_b could be obtained by replacing the derivative of $sign()$ with the hard $tanh$, which is equivalent to the condition $g_w = g_b$ when $|w| \leq 1$ [9].

Particularly for BNNs, it is of crucial importance to adjust the input elements $a^{l-1} \subset A^{l-1}$, before the approximation into the binary representation $h^{l-1} \subset H^{l-1} \in \mathbb{B}^{X_i \times Y_i \times C_i}$ by means of batch normalization. An advantage of BNNs is that the result of the batch-norm operation will be followed by $sign()$ (see Fig. 2). Since the result after applying both functions is simply $\{-1, 1\}$, the precise calculation of the batch-norm is wasteful on embedded hardware. Based on the batch-norm statistics collected at training time, a *threshold* point τ is defined, wherein an activation value $a^{l-1} \geq \tau$ results in 1, otherwise -1 [13]. This allows the implementation of the typically costly batch-norm operation as a simple magnitude comparison operation on hardware. Next, the binary convolution follows as

$$A^l = \texttt{BinConv}(H^{l-1}, B^l) = \texttt{PopCnt}(\texttt{XNOR}(H^{l-1}, B^l)), \tag{2}$$

which results in the output feature map $A^l \in \mathbb{R}^{X_o \times Y_o \times C_o}$.

B. Hardware Architecture

The baseline hardware architecture is provided by the Xilinx FINN framework [13]. The hardware design space has many degrees of freedom for compute resources, pipeline structure, number of processing elements (PEs) and single-instruction-multiple-data (SIMD)-lanes, among other parameters. The streaming architecture is composed of a series of matrix-vector-threshold units (MVTUs) to perform the `XNOR`, `popcount` and threshold operations mentioned in Sec. III-A. In Fig. 2, a single MVTU is shown in detail, containing two PEs with 32 SIMD-lanes each. A detailed view of a single PE is also provided (bottom-right). For convolutional layers, a sliding-window unit (SWU) reshapes the binarized activation maps $H^{l-1} \in \mathbb{B}^{X_i, Y_i, C_i}$ into interleaved channels of $h^{l-1} \subset H^{l-1}$, to create a single wide input feature map memory, that can efficiently be accessed by the subsequent MVTU and operated upon in a parallel manner. Max-pool layers are implemented as Boolean `OR` operations, since a

single binary "1" value suffices to make the entire pool window output equal to 1.

A single MVTU is solely responsible for a single layer in the BNN, and is composed of single or multiple PEs, each having their own SIMD lanes. The SIMD lanes determine the throughput of each PE for the `XNOR` operation. The choice of PEs and SIMD lanes determines the latency and hardware resource utilization of each layer (i.e. MVTU) on the hardware architecture. A layer's poorly dimensioned MVTU can result in an inefficient pipeline, leading to poor overall throughput. Throughput in a streaming architecture is heavily influenced by the slowest MVTU of the accelerator, as it throttles the rate at which results are produced when the pipeline is full. On the other hand, latency is dependent on the time taken by all the MVTUs of the architecture as well as the intermediate components between them (e.g. SWU, pooling unit, etc.).

Choosing the correct number of PEs and SIMD lanes for each layer becomes a design problem of balancing the FPGA's resources, the pipeline's efficiency (throughput and latency), and potentially the choice of layers in the BNN (i.e. task-related accuracy). The number of resources on the FPGA is limited, especially in the context of low-power prosthetics, making these aspects important in planning the deployment with a HW-BNN codesign approach.

C. Runtime Dynamic Frequency Scaling

In the previous section, the importance of defining the number of layers (BNN design) and PE/SIMD lanes per MVTU (HW design) was outlined. To enable efficient performance of the semi-autonomous prosthesis, a further aspect must be considered next to resource utilization and latency, namely the power consumption of the classifier. Prosthetic devices are meant to be used on a day-to-day basis, making high power consumption a prohibitive aspect to their practicality. Here, we further append the classifier with the ability to change its operating frequency dynamically at runtime. The purpose in this case, is not having the classifier continually run at its full capacity, but rather scale down its performance (in terms of latency) for more efficient use of the available energy supply. Dynamic power in CMOS scales roughly with frequency following $P_{dyn} \approx \alpha f \cdot C V_{dd}^2$, where α is the switching activity, f is the frequency, C the effective capacitance and V_{dd} the supply voltage.

In case of our target Xilinx Zynq System-on-Chip boards, the programmable logic (PL) on which the hardware acceleration is implemented, is clocked through phased-locked-loops (PLLs) controlled by a CPU-based processing system (PS). The PS can manipulate the PL's clock by writing into special registers, whose values act as frequency dividers to the PLLs. As an example, the motion of the prosthetic hand can be captured through simple sensors which are monitored by the PS. Based on this motion, the PS can drive up the frequency of the classifier and prepare for a low-latency, high accuracy classification (based on a mean classification of a batch of frames). In case of a fragile or perilous object, the lower risk of a false classification can reduce the chances of an improper grasp. The PS can also trigger the object localization task by splitting the view into multiple small images and classifying them with high throughput. This is elaborated in Sec. IV-C. These high-performance features may extend the use of Binary-LoRAX to other semi-autonomous prostheses and/or applications. Conversely, the PS may monitor the remaining battery power or system temperature and switch the classifier to low-power mode.

D. SIMD Binary Products on DSP-blocks

In resource constrained platforms, the available hardware must be used effectively. Smaller FPGAs that have a few thousands of look-up-tables (LUTs) can easily run into synthesis issues, even with small network architectures. Since digital signal processing (DSP) blocks are not heavily utilized when synthesizing our BNN accelerator designs, they presented a good alternative to LUT resources for executing the parallel `XNOR` operations of the accelerator.

The DSP48E1[1] slice is presented in Fig. 3. We exploit the internal concatenation of signals `A` and `B` to fit part of the tensor slice h^{l-1} and select it through the `X` multiplexer, forming a 48-bit wide signal. Signals `A` and `B` are asymmetric, having 30 bits and 18 bits respectively. These signals are aligned before entering the DSP, such that their concatenated value `A:B` (top blue line in Fig. 3) represents the input to one (or multiple) of the MVTU's PEs and the respective SIMD lanes (Fig. 2). Note that the top blue path in Fig. 3 skips over the "MULT" inside the DSP, allowing us to clock-gate the multiplier for further power savings. The tensor slice b^l of binarized weights is wired to input `C` of the DSP, and made internally accessible at multiplexers `Y` and `Z` (as well as `W` on DSP48E2). The order in which h^{l-1} is aligned in signals `A:B` must match their element-wise multiplicand in b^l in signal `C` to get the correct 48-bit wide output as signal `P`. By setting the `ALUMODE` signal and activating the correct multiplexers through the `OPMODE` signal (marked red in Fig. 3), the DSP is transformed into a SIMD binary product module. This low-level FPGA primitive reprogramming is not possible through High Level Synthesis (HLS), which is used to describe the overall accelerator. A script was developed to parse through the Hardware Description Language (HDL) files generated by HLS, to find all the signals corresponding to `XNOR`s in the accelerator. The connections between the operand signals and the output registers are removed, then primitive DSP modules are instantiated with the correct wiring to operate in the binary mode described earlier. The operand signals of h^{l-1} and b^l are arranged into the aforementioned `A:B` and `C` signals and connected into the DSP(s). The wide output `P` signal is then split and passed back into the next stages of the PE.

IV. Results and Design Space Exploration

A. Experimental Setup

We evaluate Binary-LoRAX on 25 objects from the YCB dataset [10], improving upon previous work by 12 ob-

[1]Can also be applied to all 7-series, Ultrascale and Ultrascale+ FPGAs (DSP48E1 and DSP48E2), as well as the Versal DSP58.

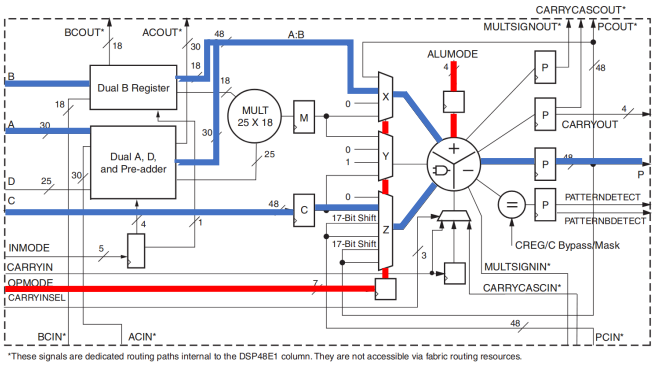

Fig. 3: The DSP48E1 Slice [41]. Appended blue path indicates the operands inside the DSP, red path indicates signals that are needed to achieve the desired binary mode.

jects [11]. The dataset is augmented through scale, crop, flip, rotate and contrast operations. The masks provided with the dataset are used to augment the background with random Gaussian noise. The dataset is expanded to 105K images for the 25 classes. The images are resized to 32×32 pixels similar to the CIFAR-10 [34] dataset. The BNNs are trained up to 300 epochs, unless learning saturates earlier. Evaluation is performed on a 17.5K test set. We trained the BNN architectures shown in Tab. I according to the method described in Sec.III-A. Each convolutional (Conv) and fully-connected (FC) layer is followed by batch-norm and activation layers except for the final layer. Conv groups 1 and 2 are followed by a max-pool layer. The target System-on-Chip (SoC) platforms for the experiments are the XC7Z020 (Z7020) for v-CNV and m-CNV prototypes, and XC7Z010 (Z7010) for μ-CNV. All prototypes are finally deployed on the Z7020 SoC. Power, latency and throughput measurements are taken directly on a running system. The power is measured at the power supply of the board (includes both PS and PL). Latency measurements are performed end-to-end on the accelerator covering the classifier's total time for an inference, while throughput is the classification rate when the accelerator's pipeline is full. Note that throughput is higher than the latency rate due to the streaming architecture working on *multiple images* concurrently in different parts of its pipeline when it is full.

TABLE I: Network architectures and hardware dimensioning.

Network	**(v)-CNV**	**m-CNV**	**μ-CNV**
Arch. L | $[C_i, C_o]$ $K = 3$ $\forall$ Conv	Conv_1_1 | [3, 64] Conv_1_2 | [64, 64] Conv_2_1 | [64, 128] Conv_2_2 | [128, 128] Conv_3_1 | [128, 256] Conv_3_2 | [256, 256] FC_1 | [512] FC_2 | [512] FC_3 | [25]	Conv_1_1 | [3, 32] Conv_1_2 | [32, 32] Conv_2_1 | [32, 64] Conv_2_2 | [64, 64] Conv_3_1 | [64, 128] Conv_3_2 | [128, 128] FC_1 | [256] FC_2 | [256] FC_3 | [25]	Conv_1_1 | [3,16] Conv_1_2 | [16, 16] Conv_2_1 | [16, 32] Conv_2_2 | [32, 32] Conv_3_1 | [32, 64] FC_1 | [128] FC_2 | [25]
PE Count **SIMD lanes**	16, 32, 16, 16, 4, 1, 1, 1, 4 3, 32, 32, 32, 32, 32, 4, 8, 1		4, 4, 4, 4, 1, 1, 1 3, 16, 16, 32, 32, 16, 1
YCB-Objects	`mug, banana, toy_airplane, chips_can, tomato_soup_can, windex_bottle, apple, scissors, sugar_box, master_chef_can, mustard_bottle, orange, pudding_box, lemon, plate, pitcher_base, potted_meat_can, mini_soccer_ball, gelatin_box, large_clamp, power_drill, tennis_ball, cracker_box, adjustable_wrench, knife`		

TABLE II: Hardware results of design space exploration. Power is averaged over a period of 100 seconds of operation.

Configuration (W,A)-bits|BNN	**Freq.** MHz	**LUT**	**BRAM**	**DSP**	**Power** [W]	**Latency** [ms]	**Throughput** [FPS]	**Acc.** [%]
(8,8) - [11]*	400	-	-	-	0.446	115	9	96.51*
(2,2) - CNV**	100	35718	140	32	2.217	4.87	860	99.91
(1,2) - CNV	100	40328	131.5	26	2.241	1.63	3049	99.89
(1,1) - CNV [13]	100	26060	124	24	2.212	1.58	3049	99.82
Binary-LoRAX: DSP XNOR + Frequency Scaling:								
(1,1) - v-CNV	$\updownarrow \frac{2}{111}$	23675	124	72	$\updownarrow \frac{1.857}{2.172}$	78.93 1.42	61 3388	**99.82**
(1,1) - m-CNV	$\updownarrow \frac{0.7}{125}$	21972	44.5	66	$\updownarrow \frac{1.879}{2.157}$	80.22 **0.45**	28 **4999**	98.99
(1,1) - μ-CNV	$\updownarrow \frac{1}{100}$	**11738**	**14**	**27**	$\updownarrow \frac{1.824}{2.028}$	80.64 0.81	16 1646	90.58

*: Running on ARM Cortex M7 (CPU frequency reported), accuracy for 13 classes, 72×72 input
**: Less PEs and SIMD lanes to fit the SoC

B. Design Space Exploration

Considering two embedded SoC platforms, the Z7020 and the more constrained Z7010, three Binary-LoRAX prototypes were investigated: v-CNV, m-CNV and μ-CNV. The CNV network is based on the architecture in [13] inspired by VGG-16 [42] and BinaryNet [9]. m-CNV and μ-CNV have a similar architecture, with fewer channels, for faster inference and to fit the Z7010 respectively. For the prosthetic hand, latency is more critical than throughput. On the Z7010, the number of PEs and SIMD lanes were chosen to minimize end-to-end latency accordingly.

In Tab. II, we report the details of the CNV network with (1,2) and (2,2) bits for weights and activations respectively. The fully binarized CNV (1,1) network achieved a comparable accuracy of 99.82% on the YCB graspable object dataset, showing the effectiveness of BNNs for this task, and the potential to add more classes in future work.

In the bottom half of Tab. II, the hardware utilization for the Binary-LoRAX prototypes is provided. For the v-CNV network, a reduction of 2386 (9%) LUTs can be observed from the regular CNV [13]. For the constrained Z7010, such reductions can make a previously non-synthesizable design realizable after moving `XNOR` operations to DSPs. The increase in DSP usage can be justified as they are not the bottleneck for synthesizable designs in our case. It is important to note that μ-CNV was synthesizable on the Z7010 *only after* moving the `XNOR` operations to the DSPs, as proposed in Sec. III-D.

C. Runtime Dynamic Frequency Scaling

Prosthetic devices used on a daily basis must offer high performance for safe and convenient use, while minimizing power dissipation to increase the continuous usage time before charging. Referring back to Tab. II, we report two values ($\updownarrow$) for power, latency and throughput per Binary-LoRAX prototype, for high-performance and power-saving modes. At 2 MHz, Binary-LoRAX's v-CNV achieves a reduction of up to 16% in power consumption with runtime frequency scaling compared to standard CNV [13]. This translates to an improvement in battery-life of up to 19%. In high-performance mode, a latency of only 0.45 ms is consumed by the m-CNV network at 125 MHz. This reduces latency by 99.7% compared to the work in [11].

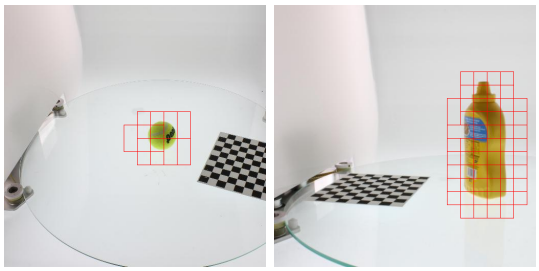

Fig. 4: The large input image is sliced into smaller images and reclassified. High confidence classifications are bounded.

Considering the performance/Watt efficiency metric, Binary-LoRAX's m-CNV achieves 2318 frames/Watt compared to 20 frames/Watt in [11]. With an optimal controller delay for myoelectric prostheses of 125 ms [12], all our classifiers consume $<$1% of the total time, leaving more slack for post-processing, actuators and other parts of the system. In power-saving mode, the Binary-LoRAX prototypes run at 0.7-2 MHz and achieve an $\sim$80 ms latency, still leaving more than 36% of the allocated delay for the controller. It is important to note that in all the reported power measurements, roughly 1.65W of power is consumed by the Z7020's ARM-Cortex A9 processor (PS) and the board. This leaves the isolated accelerator's power at roughly 0.2W in power-saving mode for all configurations, making it very energy efficient. However, we report the overall power since the accelerator is still dependent on processor calls and preprocessing. In future work, the PS power consumption can also be optimized to further reduce the classifier's overall power requirement.

In addition to the low latency of the high-performance mode, the high throughput of up to 4999 FPS can be used to improve the quality of the application. Instead of providing a single classification, the accelerator can pipeline the inference of many images (potentially from different sensors) and perform batch-classification. The batch classification result will represent the highest class over all classifications, which in practice compose of slightly different angles, lighting and distance to the object, improving the chances of a correct classification. Multi-camera prosthetics proposed in [43] can benefit from the high throughput, as more data is gathered through the multiple camera setup.

Another use of the high-performance mode is object localization in multi-object scenes. A large input image can be sliced into several smaller images and reclassified [13]. The image can be reconstructed with bounded high confidence classifications. Fig. 4 demonstrates the described function on Binary-LoRAX. This can help the prosthesis predetermine the location of different objects in a *far* scene, when the hand is not yet close to the graspable object. The approach also fits our training scheme, as the BNNs are trained on up-close images of the object (soon before the grasp), while far scenes with no central object would be unrecognizable to the BNN. The individual slices of a far scene are similar to the up-close train images.

In Fig. 5, we perform a frequency sweep on the v-CNV prototype, identifying different points of operation for different application requirements. The low-power region is considered to be below 1.90 W, while localization would require classification rates of above 2250 FPS for an input resolution of 320×240. Batch classification can be triggered in critical scenarios where a latency of $<$10 ms is needed.

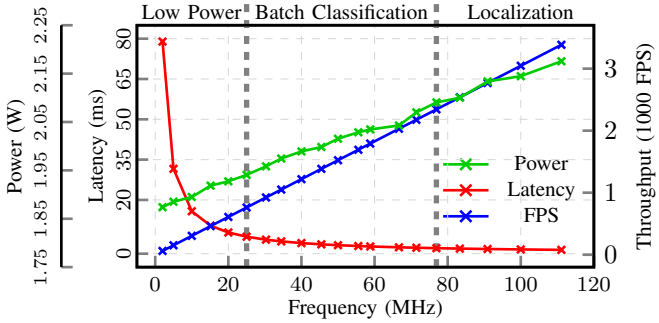

Fig. 5: Runtime frequency scaling ranging from 2MHz to 111MHz for the v-CNV prototype

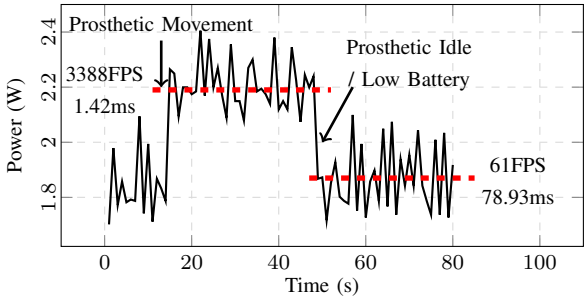

Fig. 6: Run-time change in operation mode based on application scenario, e.g. motion, delicate object or low battery

We demonstrate the application of runtime frequency scaling in Fig. 6. The total power of the chip is measured for a duration of 80 seconds. At time t=15, we introduce a stimulus representing a dangerous object or similarly a signal from a motion sensor on the hand. The event triggers the classifier to high-performance mode for an observation period of 35 seconds. If no further event occurs, the classifier winds down to low-power mode at t=50. Naturally, the intermediate frequencies shown in Fig. 5 can all be triggered for other scenarios or operating modes.

V. Conclusion

A daily-used device, such as a prosthetic hand, must operate in different modes to suit daily application scenarios. In this paper, we present a low-latency runtime adaptable XNOR classifier for semi-autonomous prosthetic hands. We enable high-performance features and power-saving modes through runtime adaptable frequency scaling. Our Binary-LoRAX prototypes achieved over $\sim$99% accuracy on a 25 class problem from the YCB dataset, and a maximum of 4999 FPS and latency of 0.45 ms. The low-power mode can potentially improve the battery-life of the classifier by 19% compared to an equivalent accelerator running continuously at full-power. This work demonstrates that BNNs have the potential to bring cutting-edge classification performance to the field of semi-autonomous prosthetics.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[2] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6517–6525.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3431–3440.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," vol. 115, no. 3, 2015, pp. 211–252.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 630–645.

[6] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[7] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, pp. 187:1–187:30, 2017.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2015.

[9] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 4107–4115. [Online]. Available: http://papers.nips.cc/paper/6573-binarized-neural-networks.pdf

[10] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 International Conference on Advanced Robotics (ICAR)*, 2015, pp. 510–517.

[11] F. Hundhausen, D. Megerle, and T. Asfour, "Resource-aware object classification and segmentation for semi-autonomous grasping with prosthetic hands," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 215–221.

[12] T. R. Farrell and R. F. Weir, "The optimal controller delay for myoelectric prostheses," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 1, pp. 111–118, 2007.

[13] 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 *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: ACM, 2017, pp. 65–74. [Online]. Available: http://doi.acm.org/10.1145/3020078.3021744

[14] S. Došen, C. Cipriani, M. Kostić, M. Controzzi, M. C. Carrozza, and D. B. Popović, "Cognitive vision system for control of dexterous prosthetic hands: experimental evaluation," *Journal of neuroengineering and rehabilitation*, vol. 7, no. 1, p. 42, 2010.

[15] M. Markovic, S. Dosen, C. Cipriani, D. Popovic, and D. Farina, "Stereovision and augmented reality for closed-loop control of grasping in hand prostheses," *Journal of neural engineering*, vol. 11, no. 4, p. 046001, 2014.

[16] G. Ghazaei, A. Alameer, P. Degenaar, G. Morgan, and K. Nazarpour, "An exploratory study on the use of convolutional neural networks for object grasp classification," 2015.

[17] J. DeGol, A. Akhtar, B. Manja, and T. Bretl, "Automatic grasp selection using a camera in a hand prosthesis," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2016, pp. 431–434.

[18] P. Weiner, J. Starke, F. Hundhausen, J. Beil, and T. Asfour, "The kit prosthetic hand: design and control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3328–3334.

[19] M. Esponda and T. M. Howard, "Adaptive grasp control through multimodal interactions for assistive prosthetic devices," *arXiv preprint arXiv:1810.07899*, 2018.

[20] Y. He, R. Shima, O. Fukuda, N. Bu, N. Yamaguchi, and H. Okumura, "Development of distributed control system for vision-based myoelectric prosthetic hand," *IEEE Access*, vol. 7, pp. 54 542–54 549, 2019.

[21] C. Shi, D. Yang, J. Zhao, and H. Liu, "Computer vision-based grasp pattern recognition with application to myoelectric control of dexterous hand prosthesis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 9, pp. 2090–2099, 2020.

[22] N. Bu, Y. Bandou, O. Fukuda, H. Okumura, and K. Arai, "A semi-automatic control method for myoelectric prosthetic hand based on image information of objects," in *2017 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*. IEEE, 2017, pp. 23–28.

[23] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *The European Conference on Computer Vision (ECCV)*. Cham: Springer International Publishing, 2016, pp. 525–542.

[24] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, 2013. [Online]. Available: http://arxiv.org/abs/1308.3432

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ACM International Conference on International Conference on Machine Learning (ICML)*, ser. ICML15. JMLR.org, 2015, p. 448456.

[26] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems (NeurIPS)*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3123–3131.

[27] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "BNN+: improved binary network training," *CoRR*, vol. abs/1812.11800, 2018. [Online]. Available: http://arxiv.org/abs/1812.11800

[28] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 345–353. [Online]. Available: http://papers.nips.cc/paper/6638-towards-accurate-binary-convolutional-neural-network.pdf

[29] A. Frickenstein, M.-R. Vemparala, J. Mayr, N.-S. Nagaraja, C. Unger, F. Tombari, and W. Stechele, "Binary DAD-Net: Binarized Drivable Area Detection Network for Autonomous Driving," in *International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020.

[30] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Structured binary neural networks for accurate image classification and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 413–422.

[31] N. Fasfous, M. R. Vemparala, A. Frickenstein, L. Frickenstein, M. Badawy, and W. Stechele, "Binarycop: Binary neural network-based covid-19 face-mask wear and positioning predictor on edge devices," in *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2021.

[32] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: A multi-class classification competition," in *The 2011 International Joint Conference on Neural Networks*, 2011, pp. 1453–1460.

[33] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[34] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[35] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[36] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "Yodann: An architecture for ultralow power binary-weight cnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 48–60, Jan 2018.

[37] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, T. Kuroda, and M. Motomura, "Brein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 tops at 0.6

w," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, April 2018.

[38] P. Guo, H. Ma, R. Chen, P. Li, S. Xie, and D. Wang, "Fbna: A fully binarized neural network accelerator," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, 2018, pp. 51–513.

[39] M. R. Vemparala, A. Frickenstein, and W. Stechele, "An efficient fpga accelerator design for optimized cnns using opencl," in *Architecture for Computing Systems (ARCS)*, 2019, pp. 236–249.

[40] N. Fasfous, M. R. Vemparala, A. Frickenstein, and W. Stechele, "Orthruspe: Runtime reconfigurable processing elements for binary neural networks," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 1662–1667.

[41] "XILINX 7 series dsp48e1 slice," no. UG479. Xilinx, Inc, 3 2018, v1.10.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[43] F. Hundhausen, J. Starke, and T. Asfour, "A soft humanoid hand with in-finger visual perception," 2020.