BinaryCoP: Binary Neural Network-based COVID-19 Face-Mask Wear and Positioning Predictor on Edge Devices

Nael Fasfous[1], Manoj-Rohit Vemparala[2], Alexander Frickenstein[2], Walter Stechele[1]
[1] Technical University of Munich (<first_name>.<last_name>@tum.de)
[2] BMW Group (<first_name>.<last_name>@bmw.de)

Abstract— **Face masks are used in many industrial settings to protect against the inhalation of hazardous fumes and particles. They also offer an effective solution in healthcare for bi-directional protection against air-borne diseases. Wearing and positioning the mask correctly is essential for its function. Convolutional neural networks (CNNs) offer an excellent solution for face recognition and classification of correct mask wearing and positioning. In the context of the ongoing COVID-19 pandemic, such algorithms can be used at entrances to corporate buildings, airports, shopping areas, and other indoor locations, to mitigate the spread of the virus. These application scenarios impose major challenges to the underlying compute platform. The inference hardware must be cheap, small and energy efficient, while providing sufficient memory and compute power to execute accurate CNNs at a reasonably low latency. To maintain data privacy of the public, all processing must remain on the edge-device, without any communication with cloud servers. To address these challenges, we present BinaryCoP, a low-power binary neural network classifier for correct facial-mask wear and positioning. The classification task is implemented on an embedded FPGA accelerator, performing high-throughput binary operations. Classification can take place at up to ∼6400 frames-per-second and 2W power consumption, easily enabling multi-camera and speed-gate settings. When deployed on a single entrance or gate, the idle power consumption is reduced to 1.65W, improving the battery-life of the device. We achieve an accuracy of up to 98% for four wearing positions of the MaskedFace-Net dataset. To maintain equivalent classification accuracy for all face structures, skin-tones, hair types, and mask types, the algorithms are tested for their ability to generalize the relevant features over a diverse set of examples using the Grad-CAM approach.**

I. Introduction

Convolutional neural networks (CNNs) have been applied to real-world problems since the early days of their conception [1]. In current times, the ongoing COVID-19 pandemic presents new challenges, which can be solved with the help of state-of-the-art computer vision algorithms [2], [3]. One of the most simple ways of mitigating the spread of the COVID-19 disease is wearing a face-mask, which can protect the wearer from direct exposure to the virus through the mouth and nasal passages. A correctly worn mask can also protect other people, in case the wearer is already infected with the disease. This bi-directional protection makes masks highly effective in crowded and/or indoor areas. Although face-masks have become a mandatory requirement in many public areas, it is difficult to ensure the compliance of the general public. More specifically, it is difficult to assert that the masks are worn correctly as intended, *i.e.* completely covering the nose, mouth and chin [4].

CNNs are the current state-of-the-art in face detection applications. Compared to classical computer vision algorithms, CNNs can provide better accuracy on problems with diverse features without having to manually extract said features [5]. This holds true only when the training dataset has a fair distribution of samples. Correctly identifying a mask on a person's face is a relatively simple task for these powerful algorithms. However, a more precise classification of the exact positioning of the mask and identifying the exposed region of the face is more challenging. To maintain equivalent classification accuracy for all face structures, skin-tones, hair types, and mask types, the algorithms must be able to generalize the relevant features over all individuals.

The deployment scenarios for the CNN should also be taken into consideration. A face-mask detector can be set at the entrance of corporate buildings, shopping areas, airport checkpoints, and speed gates. These distributed settings require cheap, battery-powered, edge devices which are limited in memory and compute power. To maintain security and data privacy of the public, all processing must remain on the edge-device without any communication with cloud servers.

Minimizing power and resource utilization while maintaining a high classification accuracy is a design challenge which necessitates hardware-software co-design. In this context, we propose BinaryCoP (Binary COVID-mask Predictor), an efficient binary neural network (BNN) classifier for real-time classification of correct face-mask wear and positioning. The challenges of the described application are tackled through the following contributions:

- Training BNNs on synthetically generated data [6] to cover a wide demographic and generalize relevant task-related features. A high accuracy of ∼98% is achieved for a 4-class problem of mask wear and positioning on the MaskedFace-Net dataset.
- Deploying BNNs on a low-power, real-time embedded FPGA accelerator based on the Xilinx FINN architecture [7]. The accelerator can idle at a low-power of 1.65W on single entrances and gates or operate at high-performance (∼6400 frames-per-second) in crowded multi-gate settings, requiring ∼2W of power.
- The BNNs are analyzed through Gradient-weighted Class Activation Mapping (Grad-CAM) to improve interpretability and study the features being learned.

II. Related Work

A. COVID-19 Face-Mask Wear and Positioning

Correctly worn masks play a pivotal role in mitigating the spread of the COVID-19 disease during the ongoing pandemic [8]. Members of the general public often underestimate the importance of this simple yet effective method of disease prevention and control. Researchers and data scientists in the field of computer vision have collected data to train and deploy algorithms which help in automatically regulating masks in public spaces and indoor locations [9], [10]. Although large-scale natural face datasets exist, the number of real-world masked images is limited [9]. Wang et al. [10] extended their masked-face dataset with a Simulated Masked Face Recognition Dataset (SMFRD), which is synthetically generated by applying virtual masks to existing natural face datasets. Cabani et al. [6] improved the generation of synthetically masked-faces by applying a deformable mask-model onto natural face images with the help of automatically detected facial key-points. The key-points of the deformable mask-model can be matched to the key-points of the face, allowing the application of the mask in a variety of ways. This allows the dataset generation process to further generate examples of incorrectly worn masks, such as chin exposed, nose exposed or nose and mouth exposed.

B. Binary Neural Networks

The memory footprint of neural networks and the complexity of their arithmetic operations on inference hardware can be reduced through parameter quantization. In the most extreme case, binarizing neural networks constrains their weights and activations to $\{-1, 1\}$, such that their memory footprint is theoretically reduced by ×32 compared to a float-32 CNN [11]. Additionally, simple `XNOR` and `popcount` operations can be used to implement multiply-accumulate (MAC) operations on inference hardware [12]. Specialized training schemes have been proposed to mitigate the loss in information capacity introduced by the low-bitwidth representation of BNNs [11], [13], [14], [12]. In some cases, the low information capacity due to binarization can have a regularization effect which improves feature generalization [13]. This is helpful in improving the classification performance on real-world data, particularly when training on synthetically generated data [15]. In [13], Courbariaux et al. introduced a scheme to train neural networks with binary weights during forward propagation while maintaining latent full-precision values during back propagation. This ensures proper gradient flow and fine adjustments through the gradients. This approach is later extended by the binarization of activations [11]. Rastegari et al. [12] proposed XNOR-Net, where both weights and activations are binarized such that the convolutions of input feature maps and weights can be approximated by a combination of `XNOR` operations and `popcounts`, followed by a multiplication with scaling factors. The introduction of scaling factors improves the information capacity of the network at the cost of more trainable parameters for each layer. This adds to the computational complexity of XNOR-Net at deployment time. For the task of face-mask detection with a single subject in the frame (e.g. gates and entrance points), more efficient forms of BNNs [11] can be applied.

C. BNN Hardware Accelerators

Several accelerators have been designed to exploit the benefits of BNNs [16], [17], [7], [18]. The Xilinx FINN [7] framework was developed to accelerate BNNs efficiently on FPGA platforms. The framework compiles high level synthesis (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. In this work, we deploy BinaryCoP on FINN-based hardware architectures to achieve an efficient acceleration of the masked-face inference on embedded FPGAs. We parameterize and synthesize accelerators with different hardware requirements, geared towards individual COVID-19 mask recognition (low-power) or multi-camera (multi-gate) classification (high-performance).

III. Method

A. Training and Inference of Binary Neural Networks

The BNN method proposed by Courbariaux et al. [11] serves as our foundation to efficiently approximate weights and activations to single-bit precision at inference time, such that the neural network's arithmetic operations can be executed as simple logic operations. Smooth model training and convergence is ensured by relying on full-precision latent weights W during training time [19]. In detail, the activation tensor $A^{l-1} \in \mathbb{R}^{X_i \times Y_i \times C_i}$, with its dimensions of X_i width, Y_i height, and C_i channels, serves as the input to the convolutional layer $l \in [1, ..., L]$. Here, A^0 and A^L represent the input image and the network's prediction, respectively. The trainable parameters of the 2D-convolutional layers are composed of the latent weight matrix $W \in \mathbb{R}^{K \times K \times C_i \times C_o}$ required for training, with kernel dimension K, input channels C_i, and output channels C_o. As previously stated, the latent weights are mapped to $\{-1, +1\}$ during the forward pass for loss calculation or deployment, resulting in the binarized $b \subset B \in \mathbb{B}^{K \times K \times C_i \times C_o}$. In the hardware implementation, -1 is expressed as a binary 0 to perform multiplications as `XNOR` logic operations. The `sign()` function in Eq. 1 is used to binarize the input feature maps and weights.

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

The derivative of the `sign()` function is almost always zero, resulting in insufficient gradient flow during training and back-propagation. This necessitates gradient flow approximation using a straight-through estimator (STE) [19].

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 to zero mean and unit variance. An advantage of BNNs is that the result of the

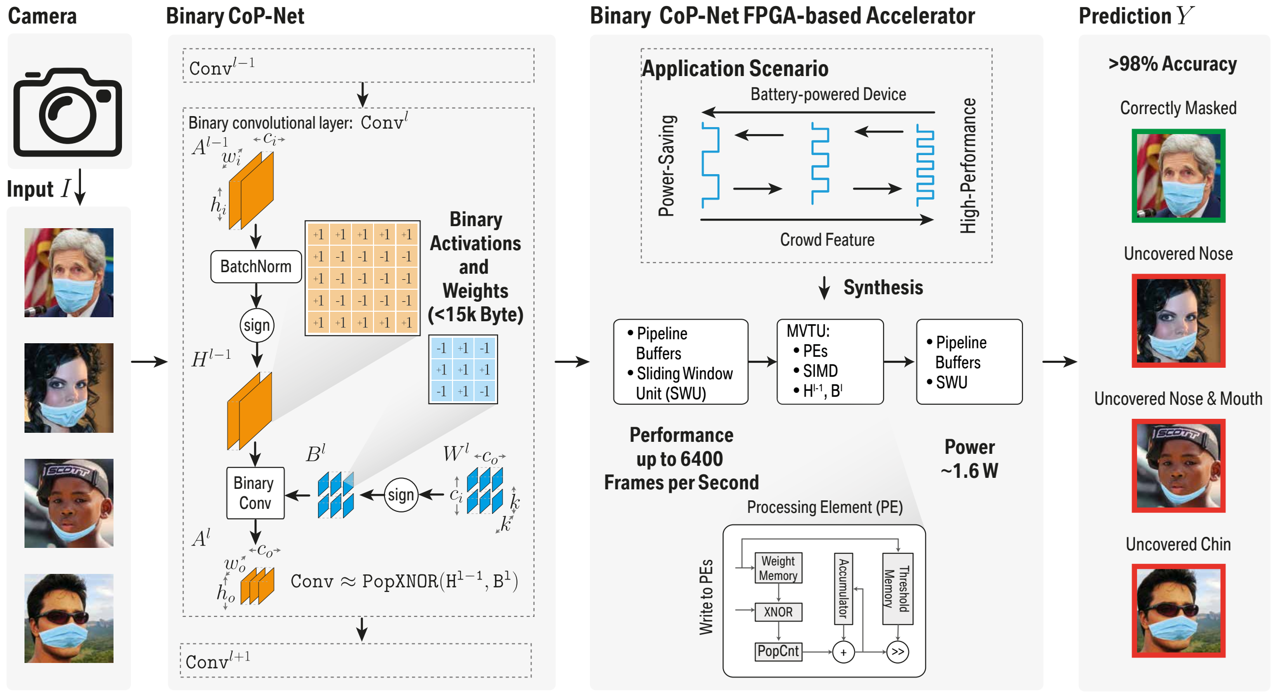


Fig. 1: Schematic representation of the BinaryCoP accelerator. A camera captures images to be classified by the neural network. The BNN accelerator is tailored for the application scenario (single or multi-gate prediction). Binary tensors are processed in the PEs of the FPGA-based accelerator using efficient XNOR operations.

batch-norm operation is followed by $\mathtt{sign}()$ (see Fig. 1). 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 [7]. 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:

$$H^{l-1} = \mathtt{sign}(\mathtt{BatchNorm}(A^{l-1})); B^l = \mathtt{sign}(W^l) \quad (2)$$

$$A^l = \mathtt{BinConv}(H^{l-1}, B^l) = \mathtt{PopCnt}(\mathtt{XNOR}(H^{l-1}, B^l)), \quad (3)$$

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

B. Hardware Architecture

The trained BNNs are conditioned for deployment on the Xilinx FINN framework [7]. The pipelined architecture offers several advantages on embedded devices, most importantly, the reduction in on-chip to off-chip memory transfers of the BNN parameters B^l and intermediate activations A^l and H^l. This is mainly feasible due to the binary format, which results in highly compact neural networks that can fit on the on-chip memory units of embedded devices. The number of processing elements (PEs), single-instruction-multiple-data (SIMD)-lanes, and other parameters can be optimized by the designer to suit the acceleration of the trained BNN. The final design is synthesized and implemented on an embedded FPGA.

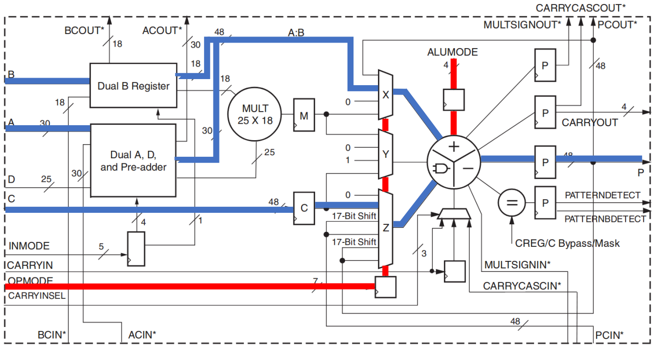
For each convolutional or fully-connected layer in the BNN, a matrix-vector-threshold unit (MVTU) is instantiated, which executes the `XNOR`, `popcount` and threshold operations mentioned in Sec. III-A. Each MVTU in the pipeline can be dimensioned for the number of PEs and SIMD lanes, which have a significant impact on hardware resource utilization, latency and the effective throughput of the pipeline. Based on the compute complexity of each layer, the available hardware resources need to be distributed over the corresponding MVTUs, such that all parts of the pipeline have a matched-throughput. A single under-dimensioned MVTU could throttle the entire pipeline, resulting in sub-optimal throughput. A single MVTU of the pipeline is shown in Fig. 1, and a corresponding PE is detailed.

For convolutional layers, an additional sliding-window unit (SWU) reshapes the binarized activation maps to create a single, wide input feature map memory, which can efficiently be accessed by the corresponding MVTU. 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.

C. SIMD Binary Products on DSP-blocks

The available hardware on resource constrained platforms 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 present a good alternative to LUT resources for executing the parallel `XNOR` operations of the accelerator.

The DSP48E1[1] slice is presented in Fig. 2. 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. 2) represents the input to one (or multiple) of the MVTU's PEs and the respective SIMD lanes). Note that the top blue path in Fig. 2 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. 2), the DSP is transformed into a SIMD binary product module. This low-level FPGA primitive reprogramming is difficult to integrate at 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 XNORs 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.



*These signals are dedicated routing paths internal to the DSP48E1 column. They are not accessible via fabric routing resources.

Fig. 2: The DSP48E1 Slice [20]. Appended blue path indicates the operands inside the DSP, red path indicates signals that are needed to achieve the desired binary mode.

D. BNN Interpretability with Grad-CAM

The output of the convolutional layers in a CNN contains localized information of the input image, without any prior bias on the location of objects and features during training. This information can be captured using Class Activation Mapping (CAM) [21] and Gradient-weighted Class Activation Mapping (Grad-CAM) [22] techniques. To apply CAM, the model must end with a global average pooling layer followed by a fully-connected layer, providing the logits of a particular input. The BNN models investigated in this work operate on a small input resolution of 32×32, and achieve a high reduction of spatial information without incorporating a global average pooling layer. For this reason, the Grad-CAM approach is better-suited to obtain visual interpretations of BinaryCoP's attention and determine the important regions for its predictions of different inputs and classes.

To obtain the class-discriminative localization map, we consider the activations and gradients for the output of the Conv2_2 layer (see Tab. I), which has spatial dimensions of 5×5. We use average pooling for the corresponding gradients and reduce the channels by performing Einstein summation as specified in [22]. With this approach the base networks do not need any modifications or retraining. Due to the synthetically generated dataset used for training, we expect BinaryCoP models to generalize well against domain shifts.

IV. Results and Design Space Exploration

A. Experimental Setup

BinaryCoP is able to detect the presence of a mask, as well as its position and correctness. This level of classification detail is possible through the more detailed split of the MaskedFace-Net dataset [6] from 2 classes, namely Correctly Masked Face Dataset (CMFD) and Incorrectly Masked Face Dataset (IMFD), to 4 classes of CMFD, IMFD Nose, IMFD Chin, and IMFD Nose and Mouth. The dataset suffers from high imbalance in the number of samples per class. From the total 133,783 samples, roughly 5% of the samples are IMFD Chin, and another 5% samples are IMFD Nose and Mouth. CMFD samples make up 51% of the total dataset while IMFD Nose makes up 39%. The dataset in its raw distribution would heavily bias the training towards the two dominant classes. To counter this, we randomly sample the larger classes CMFD and IMFD Nose to collect a comparable number of examples to the two remaining classes, IMFD Chin and IMFD Nose and Mouth. The evenly balanced dataset is then randomly augmented with a varying combination of contrast, brightness, gaussian noise, flip and rotate operations. The final size of the balanced dataset is 110K train and validation examples and 28K test samples. The images are resized to 32×32 pixels, similar to the CIFAR-10 [23] dataset. The BNNs are trained up to 300 epochs, unless learning saturates earlier. The full-precision (FP32)

TABLE I: Network architectures and hardware dimensioning.

Network	**CNV**	**n-CNV**	**μ-CNV**
Arch. $L \mid [C_i, C_o]$ $K = 3$ ∀ Conv	Conv1_1 | [3, 64] Conv1_2 | [64, 64] Conv2_1 | [64, 128] Conv2_2 | [128, 128] Conv3_1 | [128, 256] Conv3_2 | [256, 256] FC1 | [512] FC2 | [512] FC3 | [4]	Conv1_1 | [3, 16] Conv1_2 | [16, 16] Conv2_1 | [16, 32] Conv2_2 | [32, 32] Conv3_1 | [32, 64] Conv3_2 | [64, 64] FC1 | [128] FC2 | [128] FC3 | [4]	Conv1_1 | [3,16] Conv1_2 | [16, 16] Conv2_1 | [16, 32] Conv2_2 | [32, 32] Conv3_1 | [32, 64] FC1 | [128] FC2 | [4]
PE Count	16, 32, 16, 16, 4, 1, 1, 1, 4	16, 16, 16, 16, 4, 1, 1, 1, 1	4, 4, 4, 4, 1, 1, 1
SIMD lanes	3, 32, 32, 32, 32, 32, 4, 8, 1	3, 16, 16, 32, 32, 32, 4, 8, 1	3, 16, 16, 32, 32, 16, 1

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

TABLE II: Hardware results of design space exploration.

Prototype	LUT	BRAM	DSP	Power [W]		Thr.put	Latency	Acc.
				Idle	Inf.	[FPS]	[ms]	[%]
CNV	26060	124	24	1.65*	2.212	3049	1.58	**98.10**
n-CNV	20425	**10.5**	14		2.122	**6460**	0.31	93.94
μ-CNV	**11738**	14	27		2.028	1646	0.81	93.78

*Required by the board and ARM-Cortex A9 processor. Accelerator is idle.

variant used for the Grad-CAM comparison is trained for 175 epochs due to early learning saturation (98.6% final test accuracy). 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) platform for the experiments is the Xilinx XC7Z020 (Z7020) chip on the PYNQ-Z1 board. The μ-CNV design can also be synthesized for the more constrained XC7Z010 (Z7010) chip, when `XNOR` operations are offloaded to the DSP blocks as described in [24]. Power and throughput measurements are taken directly on a running system. The power is measured at the power supply of the board (includes both the processing system and programmable logic). The throughput reported is the classification rate when the accelerator's pipeline is full, while latency is measured end-to-end for a single image entering and exiting the pipeline. It should be noted that due to the pipelined architecture, throughput is not simply the reciprocal of latency. When the pipeline is full, each MVTU (layer) ideally operates on a different image of the input batch, *i.e.* concurrently processing L images at different stages of the accelerator.

B. Design Space Exploration

We evaluate three BinaryCoP prototypes, namely CNV, n-CNV and μ-CNV. The CNV network is based on the architecture in [7] inspired by VGG-16 [25] and BinaryNet [11]. n-CNV is a downsized version for a smaller memory footprint, and μ-CNV has fewer layers to reduce the size of the synthesized design. All designs are synthesized with a target clock frequency of 100MHz.

Referring back to Tab. I, the PE counts and SIMD-lanes for each layer (MVTU) are shown in sequence. For BinaryCoP-n-CNV, the most complex layer is `Conv1_2` with 3.6M `XNOR` and `popcount` operations. In Fig. 3, we mark this layer as the throughput setter, due to its heavy influence on the final throughput of the accelerator. Allocating more PEs for this layer's MVTU increases the overall throughput of the pipeline, so long as no other layer becomes the bottleneck. We allocate enough resources for `Conv1_1` to roughly match `Conv1_2`'s latency. The FINN architecture employs a weight-stationary dataflow, since each PE has its own pre-loaded weight memory. When the total number of parameters of a given layer increases, it becomes important to map these parameters to BRAM (Block RAM) units instead of logic. The deeper layers have several orders of magnitude fewer OPs, but more parameters. For these layers, increasing the number of PEs fragments the total weight memory, leading to worse BRAM utilization and no benefit in terms of throughput. Here, choosing fewer PEs, with larger unified weight memories, leads to improved memory allocation, while maintaining rate-matching with the shallow layers (see Fig. 3), leaving the throughput gains from the initial PEs unhindered. The CNV architecture in [7] follows the same reasoning for PE and SIMD allocation. For μ-CNV, we choose fewer PEs for the throughput-setters, as this prototype is meant to fit on embedded FPGAs with less emphasis on high frame rates.

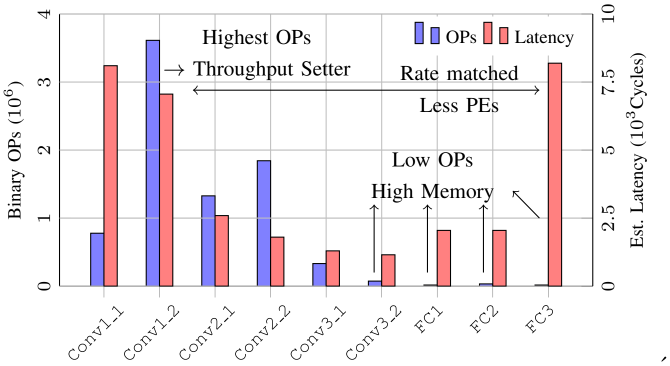
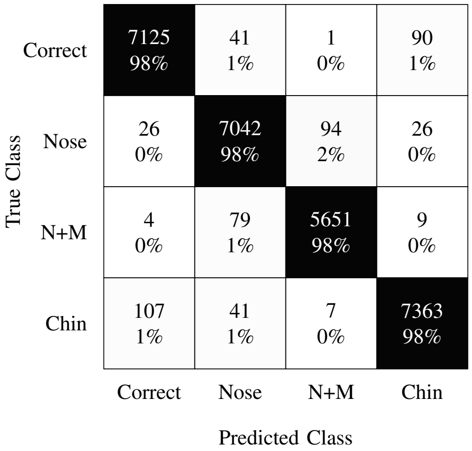


Fig. 3: Binary operations and layer-wise latency estimates based on PE/SIMD choices for BinaryCoP-n-CNV.

In Tab. II, the hardware utilization for the BinaryCoP prototypes is provided. With μ-CNV, a significant reduction in LUTs is achieved, which makes the design synthesizable on the heavily constrained Z7010 SoC. The trade-off is a slight increase in the memory footprint of the BNN, as the shallower network has a larger spatial dimension before the fully-connected layers, increasing the total number of parameters after the last convolutional layer. The choice of PE count and SIMD lanes for the n-CNV prototype allow it to reach a maximum throughput of $\sim$6400 classifications per second when its pipeline is full. This high-performance can be used to classify images from multiple cameras in multi-gate settings. The inference power values reported in Tab. II show a total power requirement of around 2W for all prototypes. For single entrance/gate classifications, all prototypes have an idle power of around 1.65W. In this setting, a classification needs to be triggered only when a subject is attempting to pass through the entrance where BinaryCoP is deployed. The idle power is required mostly by the processor (ARM-Cortex A9) on the SoC and the board (PYNQ-Z1). This can be reduced further by choosing a smaller processor to pair with the proposed hardware accelerator. Although the PYNQ-Z1 board has no PMBus to isolate the power measurements of the FPGA from the rest of the components, we can infer that the hardware accelerator requires roughly 0.4W for the inference task from the two measured power values in Tab. II. The current design is still dependent on the processor for pre- and post-processing, therefore we report the joint power for fairness.



True Class \ Predicted Class	Correct	Nose	N+M	Chin
Correct	7125 98%	41 1%	1 0%	90 1%
Nose	26 0%	7042 98%	94 2%	26 0%
N+M	4 0%	79 1%	5651 98%	9 0%
Chin	107 1%	41 1%	7 0%	7363 98%

Fig. 4: Confusion matrix of BinaryCoP-CNV on the test set.

C. Grad-CAM and Confusion Matrix Analysis

The confusion matrix in Fig. 4 shows the generalization of BinaryCoP-CNV on all classes after balancing the dataset. As expected, it is extremely rare to mistake nose+mouth exposed with a correctly worn mask. Less critically, nose and nose+mouth have a slight misclassification overlap, still at only 2% of the total samples given for each class. Finally, the chin exposed and the correct class have some sample misclassifications ($\leq$1%), which could be attributed to the chin area being small in some images and hard to detect at low-resolution.

We further analyze the output heat maps generated by Grad-CAM to interpret the predictions of our BNNs with respect to the diverse attributes of the MaskedFace-Net dataset. In Fig. 5 and Fig. 6 - Fig. 8, column 1 and 2 indicate the label and input image respectively. Columns 3, 4 and 5 highlight the heat maps obtained from the Grad-CAM output of BinaryCoP-CNV, BinaryCoP-n-CNV and a full-precision version of CNV with float-32 parameters (FP32). The heat maps are overlaid on the raw input images for better visualization. All raw images chosen have been classified correctly by all the networks, for fair interpretation of feature-to-prediction correlation.

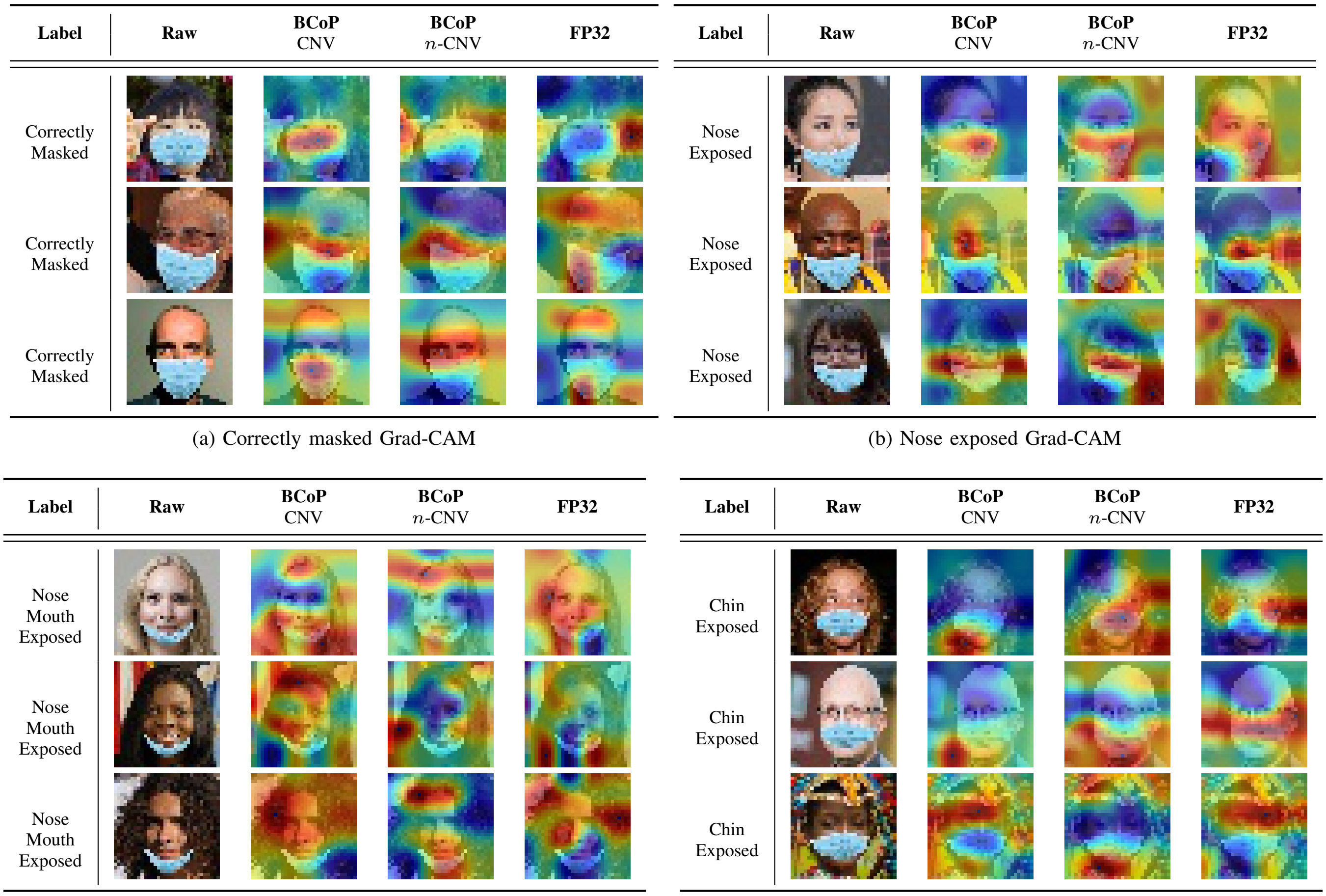
In Fig. 5(a), we analyze the Region of Interest (RoI) for the *correctly masked* class. BinaryCoP's learning capacity allows it to focus on key facial lineaments of the human wearing the mask, rather than the mask itself. This potentially helps in generalizing on other mask types. For the child example shown in the first row, the focus of BinaryCoP lies on the nose area, asserting that it is fully covered to result in a correctly masked prediction. Similarly, for the adult in row 2, BinaryCoP-CNV focuses on the upper edge of the mask, to predict its coverage of the face. This also holds for our small version of BinaryCoP, with significantly reduced learning capacity. The RoI curves finely above the mask, tracing the exposed region of the face. In the third row example, BinaryCoP-CNV falls back to focusing on the mask, whereas BinaryCoP-n-CNV continues to focus on the exposed features. Both models achieve the same prediction by focusing on different parts of the raw image. In contrast to the BinaryCoP variants, the full-precision FP32 model seems to focus on a combination of several different features on all three examples. This can be attributed to its larger learning capacity and possible overfitting.

In Fig. 5(b), we analyze the Grad-CAM output of the *uncovered nose* class. BinaryCoP-CNV and BinaryCoP-n-CNV focus specifically on two regions, namely the nose and the straight upper edge of the mask. These clear characteristics cannot be observed with the oversized FP32 CNN. In Fig 5(c), the results show the RoI for predicting the *exposed mouth and nose* class. All models seem to distribute their attention onto several exposed features of the face. Fig. 5(d) shows Grad-CAM results for *chin exposed* predictions. Although the top region of the mask points upwards, similar to the correctly worn mask, the BNNs pay less attention to this region and instead focus on the neck and chin. With the full-precision FP32 model, it is difficult to interpret the reason for the correct classification, as little to no focus is given to the chin region, again hinting at possible overfitting.

Beyond studying the BNNs' behavior on different class predictions, we can use the attention heat maps to understand the generalization behavior of the classifier. In Fig. 6 - Fig. 8, we test BinaryCoP's generalization over ages, hair colors and head gear, as well as complete face manipulation with double-masks, face paint and sunglasses. In Fig 6, we see that the smaller eyes of infants and elderly do not hinder BinaryCoP's ability to focus on the top region of the correctly worn masks. In Fig. 7, BinaryCoP-CNV shows resilience to differently colored hair and head-gear, even when having a similar light-blue color as the face-masks (row 2 and 3). In contrast, the FP32 model's attention seems to shift towards the hair and head-gear for these cases. Finally, in Fig. 8, both BinaryCoP variants focus on relevant features of the corresponding label, irrespective of the obscured or manipulated faces. This empirically shows that the complex training of BNNs, along with their lower information capacity, constrains them to focus on a smaller set of relevant features, thereby generalizing well for unprecedented cases.

D. Discussion and Comparison with Other Works

As mentioned in Sec II-A, detection of masks has piqued the interest of many researchers in the computer vision domain due to its relevance in the context of the ongoing COVID-19 pandemic. NVIDIA proposed mask recognition using object detection models [26]. These models require INT8 or Float-16 numerical precision, with ResNet-18 as a backbone for input images of 960×544. The complexity is orders of magnitude higher than the models we propose in this paper. A head-to-head comparison is difficult to make due to differences in the training approach, the CNN model, the datasets used and the application requirements. The networks are trained to predict only two classes (mask, no mask), which is a simpler problem compared to the exact positioning supported by BinaryCoP. However, the localization and higher resolution makes it a more complex task overall. With the NVIDIA Jetson Nano hardware, which typically requires $\sim$10W of power on intensive workloads, a frame-



(a) Correctly masked Grad-CAM

(b) Nose exposed Grad-CAM

(c) Mouth + nose exposed Grad-CAM

(d) Chin exposed Grad-CAM

Fig. 5: Grad-CAM output of two BinaryCoP variants and a float-32 (FP32) CNN. Results are collected for all four wearing positions on a diverse set of individuals. Binarized models show distinct regions of interest which are focused on the exposed part of the face rather than the mask. The FP32 model is difficult to interpret in some cases.

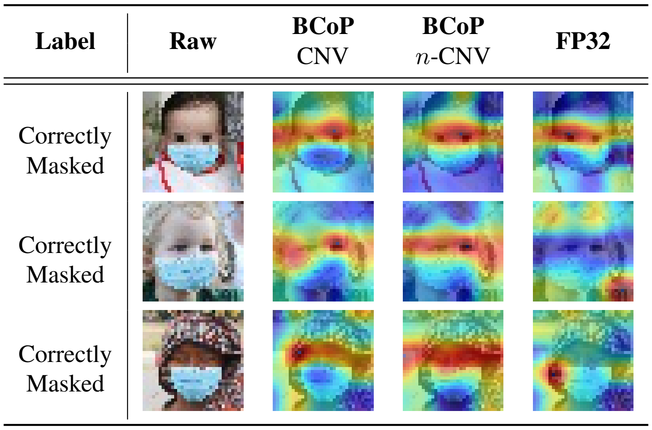


Fig. 6: Grad-CAM results for age generalization.

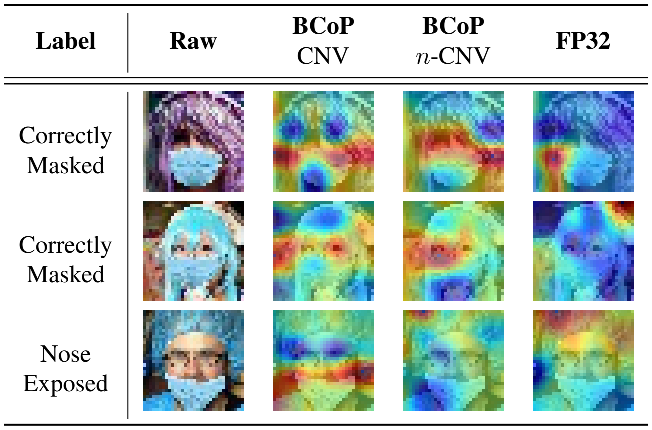


Fig. 7: Grad-CAM results for hair/headgear generalization.

rate of 21 FPS is achieved. The more powerful 25W Jetson AGX Xavier can achieve up to 508 FPS. Compared to the NVIDIA approach [26], BinaryCoP is targeted at low-power, embedded applications with peak inference power of ∼2W and high classification rates of up to ∼6400 FPS on smaller resolution input images. It is worth noting that BinaryCoP can also classify high resolution images containing multiple individuals, by slicing the input into many 32×32 frames and batch processing them. This application makes use of the high-throughput results presented in Tab. II. Another approach proposed by Agarwal et al. [27] achieves the task of detecting a range of personal protective equipment (PPE).

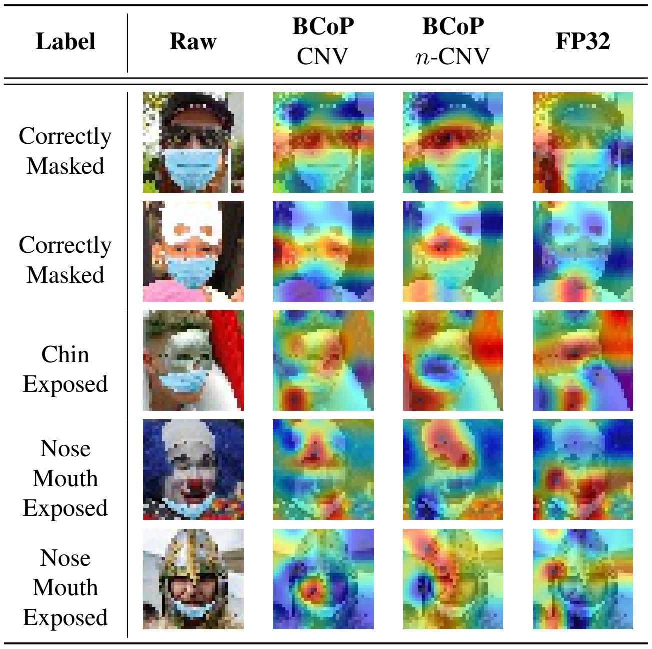


Fig. 8: Grad-CAM results for face manipulation with double-masks, face paint and sunglasses.

Processing takes place on cloud servers, which could raise privacy and data safety concerns in public settings. Wang et al. [28] propose an in-browser server-less edge computing method, with object detection models. The browser-enabled device must support the WebAssembly instruction format. The authors benchmarked their approach on an iPad Pro (A9X), an iPhone 11 (A13) and a MacBook pro (Intel i7-9750H), achieving 5, 10 and 20 FPS respectively. Needless to say, these devices (or similar) are expensive and cannot be placed in abundance in public areas. Similarly, [29] offers an Android application solution, which is suitable for users self-checking their masks. In this case, low-power, edge-hardware and continuous surveillance are not emphasized.

Our approach offers a unique, low-power, high-throughput solution, which is applicable to cheap, embedded FPGAs. Moreover, the BinaryCoP solution is not constrained to FPGA platforms. Software-based inference of BinaryCoP is also possible on other low-power microcontrollers, with binary instructions. Training on synthetic data allows us to generate more samples with different mask colors, shapes, and sizes [30], further improving the generalizability of the BNNs, while keeping real-world data available for fine-tuning stages.

V. Conclusion

In this paper, we apply binary neural networks to the task of classifying the correctness of face-mask wear and positioning. In the context of the ongoing COVID-19 pandemic, such algorithms can be used at entrances to corporate buildings, airports, shopping areas, and other indoor locations to mitigate the spread of the virus. Applying BNNs to this application solves several challenges such as (1) maintaining data privacy of the public by processing data on the edge-device, (2) deploying the classifier on an efficient XNOR-based accelerator to achieve low-power computation, and (3) minimizing the neural network's memory footprint by representing all parameters in the binary domain, enabling deployment on low-cost, embedded hardware. The accelerator requires only ∼1.65W of power when idling on single gates/entrances. Alternatively, high-performance is possible, providing fast batch classification on multiple gates and entrances with multiple cameras, at ∼6400 frames-per-second and 2W of power. We achieve an accuracy of up to 98% for four wearing positions of the MaskedFace-Net dataset. The Grad-CAM approach is used to study the features learned by the proposed BinaryCoP classifier. The results show the classifier's high generalization ability, allowing it to perform well on different face structures, skin-tones, hair types, and age groups.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] L. Wang, Z. Q. Lin, and A. Wong, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images," *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.

[3] A. I. Khan, J. L. Shah, and M. M. Bhat, "Coronet: A deep neural network for detection and diagnosis of covid-19 from chest x-ray images," *Computer Methods and Programs in Biomedicine*, vol. 196, p. 105581, Nov 2020. [Online]. Available: http://dx.doi.org/10.1016/j.cmpb.2020.105581

[4] "When and how to use masks." [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public/when-and-how-to-use-masks

[5] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision*, K. Arai and S. Kapoor, Eds. Cham: Springer International Publishing, 2020, pp. 128–144.

[6] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "Maskedface-net – a dataset of correctly/incorrectly masked face images in the context of covid-19," *Smart Health*, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352648320300362

[7] 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

[8] T. Mitze, R. Kosfeld, J. Rode, and K. Wälde, "Face masks considerably reduce covid-19 cases in germany," *Proceedings of the National Academy of Sciences*, vol. 117, no. 51, pp. 32 293–32 301, 2020. [Online]. Available: https://www.pnas.org/content/117/51/32293

[9] S. Ge, J. Li, Q. Ye, and Z. Luo, "Detecting masked faces in the wild with lle-cnns," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 426–434.

[10] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Pei, H. Chen, Y. Miao, Z. Huang, and J. Liang, "Masked Face Recognition Dataset and Application," *arXiv e-prints*, p. arXiv:2003.09093, Mar. 2020.

[11] 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

[12] 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.

[13] 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.

[14] 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

[15] A. Frickenstein, M. Rohit Vemparala, J. Mayr, N. Shankar Nagaraja, C. Unger, F. Tombari, and W. Stechele, “Binary DAD-Net: Binarized Driveable Area Detection Network for Autonomous Driving,” *arXiv e-prints*, p. arXiv:2006.08178, June 2020.

[16] 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.

[17] 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.

[18] C. Fu, S. Zhu, H. Su, C.-E. Lee, and J. Zhao, “Towards fast and energy-efficient binarized neural network inference on fpga,” in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 306. [Online]. Available: https://doi.org/10.1145/3289602.3293990

[19] 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

[20] “XILINX 7 series dsp48e1 slice,” no. UG479. Xilinx, Inc, 3 2018, v1.10.

[21] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2921–2929.

[22] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[23] A. Krizhevsky, “Learning multiple layers of features from tiny images,” *University of Toronto*, 2009.

[24] 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.

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

[26] A. Kulkarni, A. Vishwanath, and C. Shah, “Implementing a real-time, ai-based, face mask detector application for covid-19,” Feb 2021. [Online]. Available: https://developer.nvidia.com/blog/implementing-a-real-time-ai-based-face-mask-detector-application-for-covid-19/

[27] T. Agrawal, K. Imran, M. Figus, and C. Kirkpatrick, “Automatically detecting personal protective equipment on persons in images using amazon rekognition,” Oct 2020. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/automatically-detecting-personal-protective-equipment-on-persons-in-images-using-amazon-rekognition/

[28] Z. Wang, P. Wang, P. C. Louis, L. E. Wheless, and Y. Huo, “WearMask: Fast In-browser Face Mask Detection with Serverless Edge Computing for COVID-19,” *arXiv e-prints*, p. arXiv:2101.00784, Jan. 2021.

[29] K. Hammoudi, A. Cabani, H. Benhabiles, and M. Melkemi, “Validating the correct wearing of protection mask by taking a selfie: Design of a mobile application “checkyourmask” to limit the spread of covid-19,” *Computer Modeling in Engineering & Sciences*, vol. 124, no. 3, pp. 1049–1059, 2020. [Online]. Available: http://www.techscience.com/CMES/v124n3/39927

[30] A. Anwar and A. Raychowdhury, “Masked Face Recognition for Secure Authentication,” *arXiv e-prints*, p. arXiv:2008.11104, Aug. 2020.