ResCoNN: Resource-Efficient FPGA-Accelerated CNN for Traffic Sign Classification

Martin Lechner*, Axel Jantsch*, Sai Manoj Pudukotai Dinakarrao†
*Institute of Computer Technology, TU Wien, Vienna, Austria 1040
†Department of Electrical and Computer Engineering,
George Mason University, Fairfax VA, USA 22030
Email: {e1026059, axel.jantsch}@tuwien.ac.at, spudukot@gmu.edu

Abstract—**Precise classification and detection of (distorted and normal) traffic signs in real-time is one of the non-trivial requirements for safe autonomous driving. The state-of-the-art convolutional neural networks (CNNs) for traffic sign detection though accurate are resource-hungry due to their inherent structure with massive networks with millions of full-precision parameters making them infeasible for low-end FPGA platforms leading to higher implementation costs. The existing works employing low-precision bits and similar techniques though hardware friendly leads to degradation in accuracy. In this work, we propose a resource-efficient CNN (ResCoNN) architecture with a small number of weights (only 60,000 compared to a few million in state-of-the-art CNNs) and employ for traffic sign detection and classification. For increased efficiency, the network takes advantage of binary weights and integer activations, rather than employing complex computations like Batch Normalization and Exponential Linear Units. ResCoNN achieves a classification accuracy of >96% on real-world images at a framerate of 36fps on a Zynq SoC (xc7z020clg484-1) with 90% reduced weights compared to state-of-the-art CNNs.**

I. Introduction

Enhanced capabilities of hardware-software co-design and data processing techniques in embedded systems led to an increased interest in the design of real-time intelligent systems [1], [2]. Advanced driver assistance systems (ADAS) is one of the applications where an intelligent processing of data within a short time and with a better accuracy is crucial. Object detection and classification targeting, among others, traffic signs [3]–[6], road surface signs [7], and pedestrians [8] is gaining attention in such systems to ensure safe driving.

Traffic sign detection and classification is a challenging computer vision problem typically performed based on the canonical structural features like size, shape, and color [9], [10]. The main problems addressed are perspective and illumination variations such as changing lighting conditions (cloudy weather, night, heavy light, and so on). Furthermore, distortions due to human-made or natural activities (e.g., tilted, dirty or bleached boards) make the problem of traffic sign detection and classification even more complex.

With significant progress achieved by deep learning approaches like convolutional neural networks (CNNs) for signboard detection [4], [6], they are widely adopted. Current software implementations of CNNs either lack in performance [11] or demands a lot of computational resources [3], [12] leading to a high power consumption and large packages. Typically, general purpose object detection frameworks can reach between 5 frames per second (fps) [13] and 45 fps [14] on high-end GPUs. Such configurations are not practical and efficient to use for real-time situations, and in embedded systems with limited resources. Specialized hardware accelerators allow massive parallelization and ensure that real-time identification and classification of (distorted) traffic signs is possible. Among various hardware platforms, FPGAs fit best because of its parallelization, reconfigurability and low power utilization properties [15], [16]. As FPGAs have limited resources (area, processing capabilities, and memory), current state-of-the-art classification [17]–[19] and detection [12], [13] networks with millions of full-precision parameters and complex architectures do not fit low-cost FPGA platforms.

The process of developing hardware-based neural networks can be divided into three subtasks. Task one is creating a hardware-friendly optimal architecture. This includes reducing the number of parameters, eliminating the use of computational expensive functions and organizing the network in a strict feed-forward way without any shortcuts to avoid resource-heavy buffering schemes [20]. Second subtask involves minimizing the required memory footprint and trading-off with the accuracy of datatypes, as the memory is one of the bottlenecks in the hardware design. Bit-width of one bit results in binary networks. Finally, the last subtask is to implement the CNN on an FPGA. In this work, we present a resource-efficient convolutional neural network (ResCoNN) which can classify ten categories of traffic signs (including one background class) with only 60,000 weights following the procedure above mentioned. Due to the full convolutional architecture, the network can be used for traffic sign detection on arbitrary sized input images without any changes.

Our main contribution is a streamlined CNN for traffic sign identification with the following features:

- A simple, straightforward architecture with 8bit unsigned integer activations and ReLU non-linearities.
- Avoidance of computational expensive functions like Batch Normalization (BN)
- The FPGA friendly CNN requires 57% of the available LUTs and 23% of the flip-flops (FFs) on a ZedBoard with 36fps.

TABLE I: OVERVIEW OF DIFFERENT STATE-OF-THE-ART NETWORKS FOR OBJECT AND TRAFFIC SIGN CLASSIFICATION AND DETECTION

General purpose classification networks			
Architecture	**Weight Layers**	**Weights**	**Source**
AlexNet	8 (5 conv, 3 fc)	≈6.2e7	[21]
ZF-Net	8 (5 conv, 3 fc)	≈1e8	[22]
VGG16	16 (13 conv, 3 fc)	≈1.4e8	[18]
GoogLeNet	22 (22 conv, 3 fc)[1]	≈6.8e6	[19]
ResNet	152 (151 conv, 1 fc)	≈1.7e6	[17]
Traffic sign classification networks			
Architecture	**Layers**	**Weights**	**Source**
MCDNN	5 (3 conv, 2 fc)	≈1.5e6	[4]
CNN + bypass	4 (3 conv, 1 fc)	≈3.7e6	[23]
General purpose detection networks			
Architecture	**Layers**	**Weights**	**Source**
OverFeat	8 (5 conv, 3 fc)	≈1.5e8	[11]
YOLO	26 (24 conv, 2 fc)	≈2.7e8	[14]
YOLOv2	19 (19 conv)	≈2e7	[12]
R-CNN (AlexNet)	8 (5 conv, 3 fc)[2]	≈6.2e7	[21], [24]
R-CNN (VGG16)	21 (16 conv, 3 fc)[2]	≈1.4e8	[18], [24]

- State-of-the-art classification accuracy of 96.53% with only ≈60,000 binary weights (60% to 90% less compared to other state-of-the-art networks)

The rest of the paper is organized as follows: Section III describes our network architecture and our training method. The simulation results and the comparisons are presented in section IV with conclusions drawn in section V.

II. RELATED WORK

A. Convolutional Neural Networks

With AlexNet [21] winning the ImageNet competition in 2012, convolutional neural networks began to dominate visual tasks like image classification and object detection. Since then, many other network architectures have been proposed increasing the classification accuracy. During the initial phases of research on CNNs, the CNNs grew in size, from 60 million parameters (AlexNet) up to 140 million in VGG16 [18]. In the past few years, people tried to reduce the network sizes by adding new layer structures or increasing the total depth. GoogLeNet [19] adds inception modules (6.8 million parameters) and ResNet [17] uses 151 convolutional layers based on deep residual learning with shortcut connections (1.7 million parameters). However, all these networks are designed for general purpose classification. But traffic signs are of regular shape with usually a single color next to black and white in each sign. It is non-trivial to use a significantly lower number of parameters to perform traffic sign detection and classification which results in faster processing and a smaller footprint. Also, the accuracy of detection and classification has to be maintained.

Faster R-CNN [13] and YOLOv2 [12] are two promising general purpose object detection architectures. More traditional approaches like OverFeat [11] apply a classification network on many different locations and scales of an image and process the obtained class scores further to predict a bounding box. Efficient sliding window helps to reduce the necessary computations by taking advantage of the inherent share of computations in CNNs in overlapping regions. Faster R-CNN, the latest version of R-CNN [24], uses a region proposal network (RPN) on top of the last feature map i.e., before the affine layers of any classification network applied on the full image. The RPN returns only promising areas of an image to minimize the computational overhead during classification. Again, all these methods are not customized for traffic signs and uses many parameters as shown in table I.

[1]GoogLeNet is 27 layers deep with a total number of about 100 layers

[2]Network architecture / size only given for underlying classification network

B. Traffic Sign Classification and Detection in Software and Hardware

Before the era of CNNs, traffic sign detection was done using handcrafted, or support vector machine (SVM) trained features like Histogram of Gradients (HOG) [9], Scale-Invariant Feature Transform (SIFT) [25], or combinations of different features. Because of the size of state-of-the-art neural networks, a lot of current hardware implementations of traffic sign classification and detection algorithms still rely on these methods. [26] for example uses trained Haar-like features in a cascaded structure similar to the Viola-Jones face detection method [27]. As a downside, this architecture can only detect a single traffic sign. The focus of [26] is on detecting stop-signs. In contrast, [5] uses a combination of color and color transition features in the YCbCr color space. This architecture in [5] can only detect and classify speed limit signs, but the method can easily be extended to more classes.

Besides using one of the general purpose networks mentioned above, there are also architectures specialized on traffic sign classification. Multi-Column Deep Neural Networks (MCDNN) [4] are an ensemble of multiple CNNs where each network is applied to differently preprocessed input images (histogram equalization, contrast normalization, and so on). A small sized architecture with four weight-layers and bypass connections is presented in [23]. The features computed in the first convolutional layer are not only fed into the second layer but also forwarded to the classifier as high-resolution features. The basic idea behind this approach is to combine global features with more detailed local ones. For CNN based traffic sign detection, implementations of Faster R-CNN are widely used [3], [6]. Current research works also try porting large-scale neural networks on hardware platforms [28], [29]. However, such implementations typically require expensive high-end FPGAs.

In contrast, this work focuses on efficient detection and classification of traffic signs with light weighted CNNs. To achieve this, we take advantage of the geometric structure of traffic signs to reduce the number of required weights compared to general purpose networks and remove performance limiting and resource heavy functions like Batch Normalization.

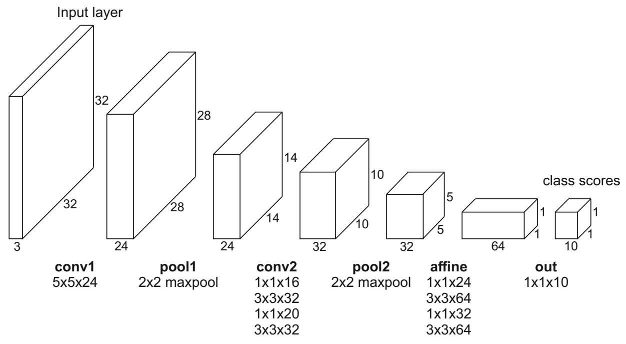

Fig. 1. Architecture of the 50k full convolutional neural network

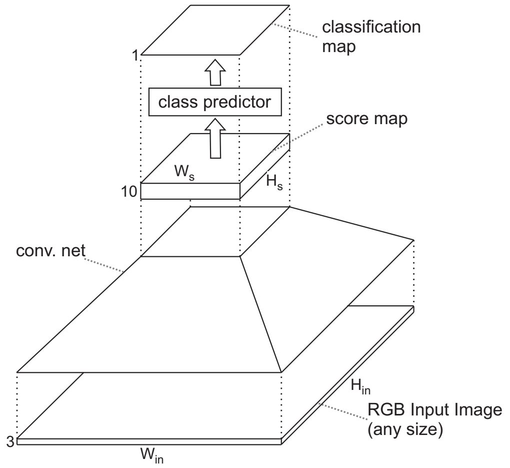

Fig. 2. Setup for traffic sign detection

III. System Architecture and Training Method

A. System Architecture

The images used for training are RGB color images with a resolution of 32×32 pixels and the network produces a $1\times1\times10$ array of class scores. However, at test time, the system (ResCoNN) should be able to process images of arbitrary size at real-time for combined classification and detection. To achieve this, we take advantage of the efficient sliding window technique presented in [11]. As a consequence, the resulting network has to be fully convolutional. Under these constraints we utilize a simple base convolutional structure for our system: $(5\times5\textit{conv} \Rightarrow 2\times2\textit{max-pool})\cdot 2 \Rightarrow 5\times5\textit{conv} \Rightarrow 1\times1\textit{conv}$. In order to address the problem of traffic sign detection and classification to fit on low-end FPGAs, we intend to reduce the number of weights needed for the CNN. Therefor, we replace most 5×5 layers by two consecutive 3×3 layers and insert 1×1 squeezing layers before each convolutional layer (except the first layer) similarly to [30]. Figure 1 gives an overview of the final network architecture with four main layers with weights (conv1, conv2, affine, out) whereas two of them are split up into four sub-layers each. Every convolutional layer is followed by a ReLU non-linearity. Even if exponential linear units (ELUs) can improve the training process [31], they are contrary to our objective to keep the network complexity low. For the same reason, we use neither batch normalization nor ensembles, as both techniques have a negative effect on the performance during the forward pass. For weight initialization, we follow the strategy recommended in [32] for ReLU units. Similar to [32], we initialize the weights of all layers with small Gaussian distributed values with a standard deviation of $\sqrt{2.0/n_l}$ where n_l is the number of inputs of each layer.

The second convolutional layer (conv2) takes 24 input channels and produces 32 output channels. A 5×5 convolutional layer would require 19,200 weights whereas the combination of 3×3 and 1×1 layers optimizes it to 11,400 for the CNN architecture in Figure 1. As such, we save about 40% of the weights in this layer.

For detection, we can, in theory, feed an input image of any size into the network leading to a three-dimensional score map of size $W_s \times H_s \times N_{classes}$ as shown in Figure 2. In practice, W_s and H_s, calculated using eq (1), have to be integer values to get a whole-numbered output map.

$$W_s = \frac{W_{in}}{P} - \sum_{k=1}^{C}\frac{F_k-1}{P_k}, \quad H_s = \frac{H_{in}}{P} - \sum_{k=1}^{C}\frac{F_k-1}{P_k} \tag{1}$$

where W_{in} and H_{in} are the sizes of the input image, P is the total sub-sampling caused by pooling, C is the number of convolutional layers, and F_k is the spatial convolution size of filter k with sub-sampling of P_k behind. For an input image of 1344×768 pixels, the size of test images in the German Traffic Sign Detection (GTSDB) dataset [33], we get a score map of $329\times185\times10$ giving a score resolution of about four pixels.

At test time, we apply the network on multiple scales of the input image similar to [11] to detect the traffic signs within a wide range of sizes inside the image, i.e., a wide range of distances from the camera. The class predictor shown on top of the score map in Figure 2 is responsible for interpreting the class scores to get the final predictions collected in the classification map. There are several possible algorithms to choose from for the prediction function, ranging from a simple max-function, over a minimum confidence level using the softmax function, to more advanced methods taking a small neighborhood and multiple scales into account. Since the softmax function requires computing the sum of exponential functions, we only compare the basic max operation and the neighborhood checking method.

B. Weight Binarization

To binarize the weights, we follow the strategy presented in [34]. When using the binary representation $\{-1,1\}$, the network structure does not have to change. The weights are still stored as full precision variables to allow a gradient-based weight optimization, but for the forward and backward pass, the weights are binarized as follows:

$$w^b = Sign(w) = \begin{cases} +1 \text{ if } w \geq 0 \\ -1 \text{ otherwise} \end{cases} \tag{2}$$

with the binarized weight w^b and the floating point weight w. To keep the output activation in the same range for all layers,

TABLE II: TRAFFIC SIGN CLASSES

class	name	examples
0	Warning Signs	Uneven Road, Road Works
1	Prohibitory Signs	No Overtaking, No Entry
2	Mandatory Signs	Turn Left, Straight Only
3	Informational Signs	Pedestrian Crossing
4	End-of Signs	End of Overtaking Restricions
5	Speed Limits	Speed Limit (50)
6	Give Way	Give Way
7	Stop	Stop
8	Priority Road	(End of) Priority Road
9	Background	-

we scale them down to 9-bit integers using binary shifting operations depending on the number of input channels. The following ReLU units eliminate the negative values resulting in 8-bit unsigned integer outputs.

C. Training Method

The network is trained using the *RMSprop* strategy with a slight modification. Instead of just monitoring the relative size of an update step r, we use it to modify the learning rate. In other words, we reduce the learning rate for large steps and increase it for tiny steps according to the following rule:

$$\eta = \begin{cases} \eta \cdot \eta_{-}, & \text{if } r > \alpha_u \\ \eta \cdot \eta_{+}, & \text{if } r < \alpha_l \\ \eta, & \text{otherwise} \end{cases} \tag{3}$$

where η_{-} and η_{+} are the factors to increase or decrease the learning rate with $0 < \eta_{-} < 1 < \eta_{+}$, and α_u and α_l being the upper and the lower threshold, respectively. Additionally, both threshold values can be decreased over time similar to the learning rate. In our experiments, we initially set $\alpha_u = 2.5e^{-3}$, $\alpha_l = 2.5e^{-4}$, $\eta_{-} = 0.75$ and $\eta_{+} = 1.25$. This method is very robust to variations of the parameters as long as the thresholds are separated by a factor of at least 10.

For weight initialization, we follow the strategy recommended in [32] for ReLU units. Similar to [32] we initialize the weights of all layers with small Gaussian distributed values with a standard deviation of $\sqrt{2.0/n_l}$ where n_l is the number of inputs of each layer.

We also use a small amount of dropout [35] for the affine layer only. While a small dropout with $p = 0.85$ in this layer improves the accuracy a bit, a higher setting has a negative impact. Also dropout applied in the convolutional layers reduces the overall performance. Since we use a fully convolutional network, it is better to drop complete feature vectors instead of single activations (spatial dropout [36]).

IV. RESULTS

A. Dataset

For our experiments, we us a modified version of the German Traffic Sign Recognition (GTSRB) dataset [37], [38] for training and the German Traffic Sign Detection (GTSDB) dataset [33] for testing. First, we included suitable images from the Belgium Traffic Sign dataset [39] and added background images to accomplish the detection task. After rescaling all images to 32×32 pixels, we increased the size of the dataset by duplicating all training images and adding slight variations to the duplicates inspired by [23]. These are:

- Rotations between $-20°$ and $20°$
- Brightness shifts by adding random numbers between -30 and 30 to all three channels equally to take different lighting conditions into account.
- Color shifts by adding random numbers between -30 and 30 to all three channels individually to consider different illumination conditions, i.e. more blueish or reddish light
- Gaussian noise with zero mean and a maximum variance of $1.25e^{-3}$

We also added another set of labels to the dataset grouping similar traffic signs together as shown in Table II.

Usually, the validation set is subsampled from the training set at random. However, the GTSRB dataset contains multiple images of the same traffic sign recorded when getting closer. Thus subsampling would cause the validation set to be not independent of the training set. We overcome this issue by manually selecting some sequences for the validation set. For the classification test set, we extended the GTSRB test set by adding background images. In total, our dataset consists of 140,360 training images of which 73,128 images contain background information, 7,822 validation images (4,460 background images) and 20,631 test images (10,137 background images). To satisfy the traffic sign detection requirement, about the half of each set contains background information.

TABLE III: NETWORK CONFIGURATIONS AND BEST VALIDATION ACCURACY ACHIEVED

	40k	50k	50k_v2	60k	100k
conv 1	5x5x24	5x5x32	5x5x24	5x5x40	5x5x32
conv 2	1x1x12 3x3x32 1x1x16 3x3x32	1x1x16 3x3x32 1x1x20 3x3x32	1x1x12 3x3x32 1x1x16 3x3x32	1x1x20 3x3x48 1x1x24 3x3x48	1x1x16 3x3x48 1x1x24 3x3x48
affine	1x1x16 3x3x64 1x1x32 3x3x64	1x1x24 3x3x64 1x1x32 3x3x64	1x1x16 3x3x64 1x1x40 3x3x80	1x1x24 3x3x64 1x1x32 3x3x64	1x1x32 3x3x96 1x1x48 3x3x96
out	1x1x10	1x1x10	1x1x10	1x1x10	1x1x10
accuracy (%)	95.62	97.53	96.61	97.77	97.79

B. Classification Results

In our experiments, we evaluated different scales of our architecture presented in Section III. Table III shows the layer configuration for each of the four networks named corresponding to the approximate number of parameters. All networks are trained using the same training method and hyperparameter settings. The accuracies reported in the last row of Table III are obtained by taking the best validation accuracy of three independently trained networks for each configuration. Although the *100k* network achieves the highest accuracy, we choose the *60k* version for further experiments, as it performs nearly the same while having 40% less weights.

On the test set, our best network reaches a classification accuracy of 96.53%. While most per-class accuracies are above 98.5%, two classes (prohibitory signs and background information) show a significant drop. For the background images, the discrepancy is easy to explain as they contain a lot of other not traffic related signboards and various company logos that might look similar to traffic signs. Regarding the prohibitory signs, the most noticeable error sources are warning signs (class 0), speed limits (class 5) and priority road signs (class 8). Prohibitory signs and speed limits have a very similar appearance which can explain the confusion but the missclassifications with priority road signs are not obvious as they feature a different shape and color.

C. Detection Results

In the context of autonomous driving, it is not important to locate traffic signs precisely in an image, i.e., a nicely fitting bounding box is not required. Hence our detection network returns only the classes of signs present in the image and not their location. For testing the detection accuracy, we directly applied our trained classification network on six scale versions (original image and five downscaled images) of full-scale images of the GTSDB test set. While our network was able to detect 92.1% of the traffic signs correctly, it also produced a high rate of false positives. The main reason is that the classification accuracy for background images is only 89.6% in combination with the size of the score map of 329x185. Thus, the score map contains 60,865 individual scores, and even with 94.8%, there will be an average of roughly 6,330 misclassifications on a single image when performing the classifications based on the maximum score only. With a more advanced approach of checking the neighborhood, i.e. a detected traffic sign in the score map is only taken as a valid detection if the 3-by-3 neighborhood contains at least one additional detection of the same class, we can reduce the missclassifications to approximately 6 per image. A further improvement can be achieved by clever filtering and post-processing schemes as used in [11]. However, such methods conflict with our goal of building a simple architecture to be used on FPGAs.

D. Hardware Results

As a hardware platform, the Zynq SoC (xc7z020clg484-1) is used, mounted on the ZedBoard, and operating at 100MHz. It offers 53,200 Look-Up Tables (LUTs), 106,400 Flip-Flops (FFs), 280 units of 18k block RAM (BRAM) and 220 DSP48E units as a part of FPGA.

To save resources, the weights are stored using the $\{0, 1\}$ binary representation although the architecture presented in Section III uses -1 and 1. This modification results in more efficient computations since it does not require a 2-bit signed datatype for the weights and removes the need of multiplying the weights and the activations. Because each convolutional layer (except the out layer) is followed by a ReLU layer, the ReLU functionality is included in the convolutional layer. A benefit of this combination is that the output - which has to be buffered for the following layer - fits in an unsigned integer variable of 8bit instead of a 9bit signed integer.

TABLE IV: SYNTHESIS RESULTS FOR THE 3×3 CONVOLUTIONAL LAYERS OF THE 60K ARCHITECTURE

Channels	Filters	LUT	FF	BRAM_18K
20	48	4,418 (8%)	3,722 (3%)	40 (14%)
24	48	5,223 (10%)	4,558 (4%)	48 (17%)
24	64	6,790 (13%)	5,779 (5%)	48 (17%)
32	64	9,025 (17%)	7,942 (7%)	64 (23%)

TABLE V: SYNTHESIS RESULTS FOR THE 1x1 CONVOLUTIONAL LAYERS OF THE 60K ARCHITECTURE. ROWS ONE TO THREE ARE SQUEEZING LAYERS AND ROW FOUR IS THE *out* LAYER

Channels	Filters	LUT	FF	BRAM_18K
40	20	504 (0.9%)	234 (0.2%)	0 (0%)
48	24	690 (1.3%)	324 (0.3%)	0 (0%)
64	32	1,207 (2.3%)	537 (0.5%)	0 (0%)
64	10	465 (0.9%)	226 (0.2%)	0 (0%)

With the modifications above, the binary weight network is synthesized using Vivado HLS and verified using the integrated RTL/C co-simulation tool. Assuming a hardware-friendly C/C++ code, HLS allows a simple and fast hardware synthesis. Because the design is pipelined and embedded in a streaming environment, it can process one input at each clock cycle. One input can contain multiple variables depending on the number of input channels. This would allow a frame rate of 325fps. However, such a frame rate is not required for traffic sign detection and additionally, a good amount of resources can be saved by reducing the parallel computations. Thus, each of the 9 elements of the 3×3 filters are computed one after another resulting in a speed of 36fps. The synthesis results of all 3-by-3 convolutional layers are given in Table IV and the results for the 1-by-1 squeezing layers in Table V. The results show, that the BRAM_18K memory resources are the main limitation. Computing a 3×3 filter requires to buffer two lines of each input channel. Thus, the number of required BRAM_18K units is always two times the input channels. Because the BRAM_18K units are organized as $2048 \times 9bit$ (8bit + 1 parity bit), the results are valid for input images up to a with of 2048 pixels.

For an approximation of the hardware size of the whole 60k architecture, two additional layers have to be analyzed. These are the 5-by-5 convolutional input layer and the 2-by-2 max-pooling layers. Synthesizing the convolutional layer results in an usage of 1,422 (3%) LUTs, 1,040 (1%) FFs, and 12 (4%) BRAM_18Ks under the same constraints as used for the 3×3 layers. The pooling layers are simple to analyze. The hardware required is minimal, but 2-by-2 pooling requires buffering one line of each input channel. Because splitting up the maximum operation, does not change the result, half of the necessary BRAM units can be saved. In more detail, taking the maximum of the two values to be buffered reduces the required memory as well as the memory accesses by the factor of two since a new value only has to be stored every second clock cycle. As a result, each 2-by-2 max-pooling layer needs a number of BRAM units equal to half of the input channels of this layer. That is 20 for the first and 24 for the second pooling layer.

TABLE VI: ESTIMATION OF THE REQUIRED RESOURCES FOR THE 60K NETWORK ON A ZEDBOARD

LUT	FF	DSP48E	BRAM_18K
30,434 (57%)	24,686 (23%)	0 (0%)	256 (91%)

The overall synthesis outcomes are given in Table VI. As explained, the memory required for buffering the input channels for each convolutional layer and the pooling layers is the main limitation for the depth of the network. Due to binarization, the weights require only 7.5kB of memory. On the other hand, there are enough LUT and FF resources available to increase the width of the network, i.e. the number of weights in each layer, or to integrate other functionalities directly onto the FPGA.

V. CONCLUSION

Computer vision with the aid of convolutional neural networks is widely employed in ADAS and other similar applications. The existing works primarily focus on optimization of weights, and approximation of CNNs. However, a large number of weight values lead to a large memory footprint and processing overheads. In contrast, we have presented a resource- and performance efficient CNN based traffic sign detection architecture towards an efficient hardware accelerated implementation. The proposed ResCoNN employs simple unsigned integer activations and ReLU non-linearities instead of complex functions such as batch normalization. The results show that high accuracy classification can be done with only 60k binary weights (which is 60% to 90% compared to other traffic sign classification networks), a simple architecture with 8bit integer activations while omitting computational expensive functions in the forward pass. We deployed the resulting CNN on a ZedBoard where we could realize a frame rate of 36fps. However, the memory required for buffering the activations between the individual layers is the main limitation which we will address in a future work.

REFERENCES

[1] M. Wess *et al.*, "Weighted quantization-regularization in DNNs for weight memory minimization towards HW implementation," *IEEE Trans. on Computer Aided Systems of Integrated Circuits and Sys.*, 2018.
[2] S. Pagani *et al.*, "Machine learning for power, energy, and thermal management on multi-core processors: A survey," *IEEE Tran. on Computer Aided Systems of Integrated Circuits and Sys.*, 2018.
[3] X. Changzhen *et al.*, "A traffic sign detection algorithm based on deep convolutional neural network," in *IEEE ICSIP*, 2016.
[4] D. Cirean *et al.*, "Multi-column deep neural network for traffic sign classification," vol. 32, pp. 333–8, 02 2012.
[5] F. Schwiegelshohn *et al.*, "FPGA based traffic sign detection for automotive camera systems," in *2015 10th ReCoSoC*, June 2015, pp. 1–6.
[6] Z. Zuo *et al.*, "Traffic signs detection based on faster r-cnn," in *2017 IEEE 37th ICDCSW*, June 2017, pp. 286–288.
[7] R. Qian *et al.*, "Road surface traffic sign detection with hybrid region proposal and fast r-cnn," in *ICNC-FSKD*, 2016.
[8] D. Ribeiro *et al.*, "A real-time pedestrian detector using deep learning for human-aware navigation," *CoRR*, vol. abs/1607.04441, 2016.
[9] I. M. Creusen *et al.*, "Color exploitation in hog-based traffic sign detection," in *2010 IEEE ICIP*, Sept 2010, pp. 2669–2672.
[10] M. Liang *et al.*, "Traffic sign detection by roi extraction and histogram features-based recognition," in *IJCNN 2013*, Aug 2013, pp. 1–8.
[11] P. Sermanet *et al.*, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013.
[12] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016.
[13] S. Ren *et al.*, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.
[14] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.
[15] J. Fowers *et al.*, "A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications," in *ACM/SIGDA Int. Symp. on FPGAs*, 2012.
[16] M. I. AlAli *et al.*, "Implementing image processing algorithms in FPGA hardware," in *2013 IEEE AEECT*, Dec 2013, pp. 1–5.
[17] K. He *et al.*, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.
[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
[19] C. Szegedy *et al.*, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.
[20] M. Wess *et al.*, "Neural network based ECG anomaly detection on FPGA and trade-off analysis," in *Int. Symp. on Circuits and Systems (ISCAS)*, 2017.
[21] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105.
[22] M. D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks*. Cham: Springer International Publishing, 2014, ch. Learning and Inference, pp. 818–833.
[23] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *IJCNN 2011*, July 2011, pp. 2809–2813.
[24] R. Girshick *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE CVPR*, 2014.
[25] M. C. Kus *et al.*, "Traffic sign recognition using scale invariant feature transform and color classification," in *ISCIS*, 2008.
[26] W. Shi *et al.*, "An FPGA-based hardware accelerator for traffic sign detection," *IEEE Transactions on VLSI Systems*, vol. 25, no. 4, pp. 1362–1372, April 2017.
[27] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE CVPR*, vol. 1, 2001, pp. I–511–I–518 vol.1.
[28] H. Li *et al.*, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *FPL 2016*, Aug 2016, pp. 1–9.
[29] Embedded vision alliance - "Caffe to Zynq: State-of-the-art machine learning inference performance in less than 5 watts,". Available at https://www.embedded-vision.com/platinum-members/xilinx/embedded-vision-training/videos/pages/may-2017-embedded-vision-summit-kathail (7/2017).
[30] F. N. Iandola *et al.*, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016.
[31] D. Clevert *et al.*, "Fast and accurate deep network learning by exponential linear units (elus)," *CoRR*, vol. abs/1511.07289, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289
[32] K. He *et al.*, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015.
[33] S. Houben *et al.*, "Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks*, no. 1288, 2013.
[34] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016.
[35] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, pp. 1929–1958, 2014.
[36] J. Tompson *et al.*, "Efficient object localization using convolutional networks," *CoRR*, vol. abs/1411.4280, 2014.
[37] J. Stallkamp *et al.*, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *IJCNN 2011*, 2011, pp. 1453–1460.
[38] J. Stallkamp *et al.*, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, 2012.
[39] R. Timofte. Belgiumts dataset. Available at http://btsd.ethz.ch/shareddata/ (5/2017).