Mask-Net: A Hardware-efficient Object Detection Network with Masked Region Proposals

Hanqiu Chen
Nanjing University
Nanjing, China
181870014@smail.nju.edu.cn

Cong Hao
Georgia Institute of Technology
Atlanta, GA, USA
callie.hao@ece.gatech.edu

***Abstract*—Object detection on embedded systems is challenging because it is hard to achieve real-time inference with low energy consumption and limited hardware resources. Another challenge for low-latency and energy-efficient object detection on embedded systems is to find hardware-friendly methods to avoid redundant computation. To address these challenges, in this work, we propose Mask-Net, a hardware-efficient object detection network with masked region proposals in regular shapes. First, we propose a hardware-friendly region proposal method to avoid redundant computation as much as possible and as early as possible, with slight or no accuracy loss. Second, we demonstrate that our method is generalizable by applying it to many detection backbones including SkyNet, ResNet-18 and UltraNet. Our method behaves well in different scenarios, including DAC-SDC dataset, UAV123 dataset and OTB100 dataset. We choose SkyNet as our base model to design an accelerator and verify our design on Xilinx ZCU106 FPGA. We observe a speedup of $1.3\times$ and about 30% energy consumption reduction when the FPGA runs at different frequencies from 124 MHz to 214 MHz with only slight accuracy loss. We also conduct a design space exploration and demonstrate that our accelerator can achieve a theoretical speedup of $1.76\times$ with masked region proposals compared with the accelerator without masked region proposals. This is achieved by optimally allocating DSPs to different parts of the accelerator to balance the computations before and after the mask.**

***Keywords*—embedded systems, object detection, DNN, region proposal, mask, FPGA**

I. Introduction

Recent years have witnessed a rapid development of deep neural networks (DNNs) in the area of computer vision. Many deep models have demonstrated their success in computer vision tasks, like AlexNet [1], VGG [2] and ResNet [3]. Empowered by edge computing, researchers are also trying to deploy DNNs on embedded systems for some real-life applications. Since embedded systems are usually resource-constrained, it is hard to deploy well-behaved computation-intensive DNNs on them. Meanwhile, it is also challenging to satisfy the demanding requirements of high inference accuracy and throughput, low energy budget and limited memory capacity at the same time.

To solve these challenges, one possible approach is to do some optimizations on hardware or software or use hardware/software co-design techniques to design hardware-efficient DNN architectures. Intensive studies have been done on them and we have got some promising results:

Hardware Design. Most of the computer vision tasks are deployed on application-specialized devices from one of the four following categories: CPU-based, GPU-based, FPGA-based and ASIC-based. Among them, FPGA and ASIC allow the users to alter the hardware design to meet various design requirements. Thanks to this flexibility, many previous works [4]–[8] have proposed efficient DNN accelerators to improve the DNN inference speed and reduce energy consumption.

Software Design. Designing hardware-efficient DNN models from software is another solution. For instance, designing lightweight DNNs that will be deployed on mobile devices is a new trend. This includes SqueezeNet [9], ShuffleNet family [10] and MobileNet family [11]. Besides, model compression is also proved to be effective in recent literatures. This includes some popular methods, such as pruning [12]–[14], quantization [15]–[17], low-rank approximation and sparsity [18], [19]. Another possible direction is to use adaptive inference. By applying adaptive inference methods from four common dimensions: layer dimensions, channel dimensions, input image resolution dimensions and computation precision dimensions, we can witness a decrease in energy consumption and increase in execution speed with only slight accuracy loss.

Hardware/Software Co-design. Hardware/Software Co-design methodologies are also promising to solve the challenges mentioned above since co-design methods combine the advantages from both hardware and software. For example, SkyNet [20] is proposed by using a bi-directional DNN design approach based on neural architecture search (NAS) to meet hardware constraints. Another good example is FNAS [21], which is a hardware-aware neural network search framework to make a balance between FPGA implementation efficiency and inference accuracy.

In this work, we propose **Mask-Net** with a light-weight and hardware-friendly mask generation branch to focus on the detection of important regions only, aiming to avoid redundant computations and data movement waste. We summarize our contributions as follows:

- We demonstrate that the existing region proposal methods using box regression are too complicated and require rich feature extraction before the region proposal, which are not suitable for light-weight embedded object detection.
- We propose a hardware-friendly grid-based mask generation method, named **Mask-Net**, which requires much less information and can skip redundant computations at an early stage. We also investigate the impact of different mask shapes

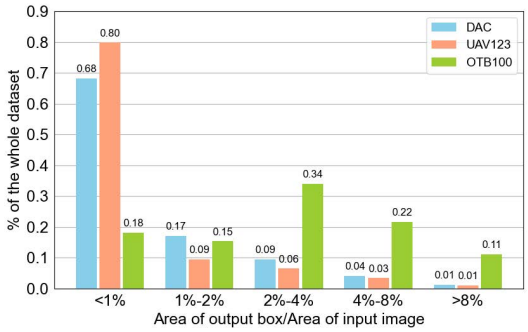

Fig. 1: The distribution of bounding box relative size in DAC-SDC training set, OTB100 dataset and UAV123 dataset. The bounding box relative size equals to the ratio of output bounding box size divided by the input image size.

regarding the computational cost and detection accuracy, demonstrating that our masked region proposal is effective.
- We apply our method on various detection backbones, including SkyNet [20], ResNet-18 [3], and UltraNet [22], to verify the efficiency and robustness of our method. We evaluate Mask-Net on three single object detection and tracking datasets, including DAC-SDC dataset, OTB100 dataset, and UAV123 dataset, to demonstrate that our technique can be scaled to various scenarios.
- We choose SkyNet as a base model to perform a case study and design an accelerator for Mask-Net on Xilinx ZCU106 FPGA. We achieve a speedup of $1.3\times$ with about 30% energy reduction with slight accuracy drop. We also do a design space exploration to generate a theoretically optimal solution to allocate DSPs on FPGA, which will help to maximize the throughput under a fixed number of DSPs. The speedup can achieve $1.76\times$ according to C/RTL co-simulation results.

II. Motivation

Although the methods mentioned in the introduction can help solve the challenges in fast-speed and low-energy computer vision tasks on embedded systems, there is still room for improvement. First, we find that in object detection tasks, a large part of the computational resources of a neural network are wasted on the background. According to our preliminary study on some single object detection and tracking datasets, e.g., DAC-SDC [23], OTB100 [24] and UAV123 [25], about 90% of the region in an image is background, as shown in Fig. 1, and it is unnecessary compute these regions.

Second, we choose a lightweight detection network SkyNet [20] as the backbone to do a contrast experiment. We imitate Faster-RCNN's approach [26] to perform bounding box regression and utilize the generated bounding boxes as region proposals. In the experiment, we find that the location and size of the bounding box are difficult to converge during training, and the generated regions of interest almost make no sense. The reason is that to obtain meaningful regions via regression, much more features are needed to be extracted and thus require more trainable parameters. Given limited resources on embedded systems, the existing regression-based region proposal methods are not applicable.

Motivated by the challenges of low latency and energy-efficient object detection on embedded systems and drawbacks of existing region proposal networks, we propose a lightweight grid-based mask generation method, and we refer to our proposed architecture as Mask-Net as shown in Fig. 2. The most critical part of the Mask-Net is that it introduces a mask generation new branch, which is a two-layer fully convolution network (FCN) and can be trained end-to-end for region proposals.

Mask-Net has some promising features as follows:

- **Small Overhead**. Since Mask-Net utilizes convolution layers in the object detection backbone to extract object's features, only a few additional convolution layers are needed to generate masked regions of interest. According to C/RTL co-simulation results, only 6% of the total inference time allows the network to correctly distinguish between objects and background.
- **Hardware Friendly**. The mask generation branch can reuse convolution modules in the backbone. As a result, it can share hardware resources with the backbone. Moreover, we introduce mask shape regularization to change non-rectangular regions of interest in a mask into rectangular ones to avoid complex control logic on hardware. Besides, channel shuffle [10] in group convolution layers of the mask generation branch helps to reduce data movement between DRAM and on-chip memory.
- **Generalizable**. Our region proposal method can be applied to various object detection or tracking backbones conveniently with little modification on the parameters of the mask generation branch. After finishing training the new branch specifically for the detection task and the detection backbone, the detection backbone only needs to be fine-tuned to cooperate with the mask generation branch better. Besides, our techniques work well in different scenarios. We get promising results when evaluating our techniques on different single object detection or tracking datasets, such as DAC-SDC [23], UAV123 [25] and OTB100 [24].

III. Related Work

A. Background of Region Proposal

Sometimes objects are relatively small compared with the whole image in object detection tasks. Initially, people use the sliding window algorithm [27] to compute every pixel in an image to get the exact location of objects. However, this method is computation expensive and will cause high latency. Region proposal is an effective method to avoid unnecessary computations. It is used to generate regions of interest beforehand to avoid computation on inexpensive features. According to a recent survey [28], there are two ways to generate region proposals: grouping methods (such as Selective Search [29], [30]) and window scoring methods (such as Objectness [31], [32]). Among them, Selective Search has

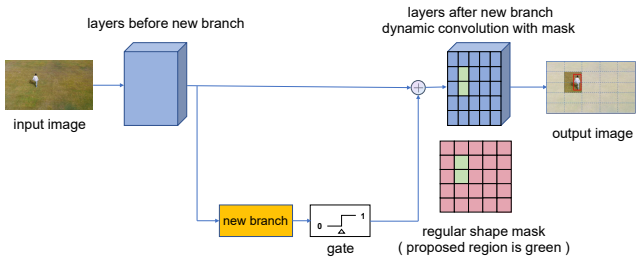

Fig. 2: The architecture of Mask-Net. The mask will divide feature maps after the new branch into small patches and only regions of interest are needed to be computed.

demonstrated its success in some emerging two-stage object detectors like R-CNN [33], SPP-net [34] and Fast-RCNN [35]. The innovation of Selective Search is to calculate the similarity between all adjacent regions and then stitch regions with similar features to generate regions of interest.

Nevertheless, Selective Search still consumes so much time compared with the object detection backbone. Region Proposal Network (RPN) in Faster R-CNN [26] enables nearly cost-free region proposals by sharing full-image convolutional features with the detection backbone and thus enables nearly cost-free region proposals. However, using RPN with present anchor boxes is computationally expensive, which needs deep convolution layers to extract enough features. Mask R-CNN [36] has an extra branch for object mask prediction, which will be used for instance segmentation. This method, however, does not generate rectangular regions and thus is not beneficial for hardware acceleration.

B. Dynamic Neural Network

The dynamic neural network is attracting more attention in deep learning. One of the biggest advantages of the dynamic neural network is that it can better adapt to different inputs through adjustable parameters, structures and computation graphs, leading to higher computational efficiency, accuracy, representation power and adaptiveness. In computer vision tasks, usually the training and inference process of a neural network is spatial-related. This feature suggests that spatial-adaptive dynamic computation approaches are promising and can reduce redundant computation. According to a recent review [37], there are three common categories of spatial-wise dynamic neural networks:

- **Pixel-level.** In a pixel-level dynamic neural network, each pixel in the input image is treated adaptively for improved flexibility of feature representation. For instance, precision gating (PG) [38] is a dynamic dual-precision quantization technique. It uses a binary decision mask with an adaptive threshold to select relatively important pixels in an image and compute them with full precision. Channel gating neural network (CGNet) [39] is proposed from the DNN channel dimension to skip the computation of ineffective pixels on the subset of input channels. However, pixel-level dynamic neural network needs higher levels of computation, which will cause real acceleration slower on hardware.
- **Region-level.** Region-level dynamic neural networks usually perform inference using regions or patches cropped from the input image to avoid the challenges of accelerating pixel-level dynamic neural networks on hardware. Glance and Focus Network (GFNet) [40] is a good example. It first extracts a quick global representation of the input image at a low-resolution level and then selects salient regions to learn more detailed features. Another good example is Recurrent attention CNN (RA-CNN) [41]. It can learn the feature representations for image classification from cropped patches of the input image and generate the attention map for the next scale at the same time. Yet, one of the disadvantages of the region-level dynamic neural network is that cropping the image into small parts will degrade the practical efficiency and accuracy.
- **Resolution-level.** Dynamic neural networks can also treat the image as a whole in adaptive resolutions for lower computational cost. Since it is sufficient for neural networks to recognize easy samples under a relatively low resolution, it is unnecessary to consider the whole image under the same resolution as traditional DNNs. For instance, Scale-aware Face Detection [42] uses Scale Proposal Network (SPN) to generate fine-grained scale proposals before the face detection task to effectively reduce redundant computation. ELASTIC [43] can learn a scaling policy in which images are processed at different resolutions in different layers by just adding a parallel scaling proposal new branch with little computation overhead.

IV. Proposed Approaches and Innovations

In this section, we first introduce our new architecture: Mask-Net. Then we will discuss our innovative approaches for both algorithm and hardware in the following section.

A. New Architecture

The main idea is to add a lightweight branch to generate a rectangular mask, which can help to separate the object from background as early as possible, so that the accelerator can skip the computation for background. The new architecture is shown in Fig. 2. The input of Mask-Net is an image with an object to be detected, and the output is the location, height and width of object's bounding box. The output of the new branch is a confidence matrix, which is considered as the confidence mask, and each patch in the mask represents a patch at the corresponding position in the feature map of the backbone. The gate function here selects proposed regions based on the value of each pixel in the mask. Let y_l^* and y_l represent the feature map of the l_{th} layer with masked region proposals and the original feature map of the l_{th} layer, respectively. After applying the mask to all the layers after the new branch, the new feature map can be written as follows:

$$y_l^* = y_l \odot upsample(mask) \tag{1}$$

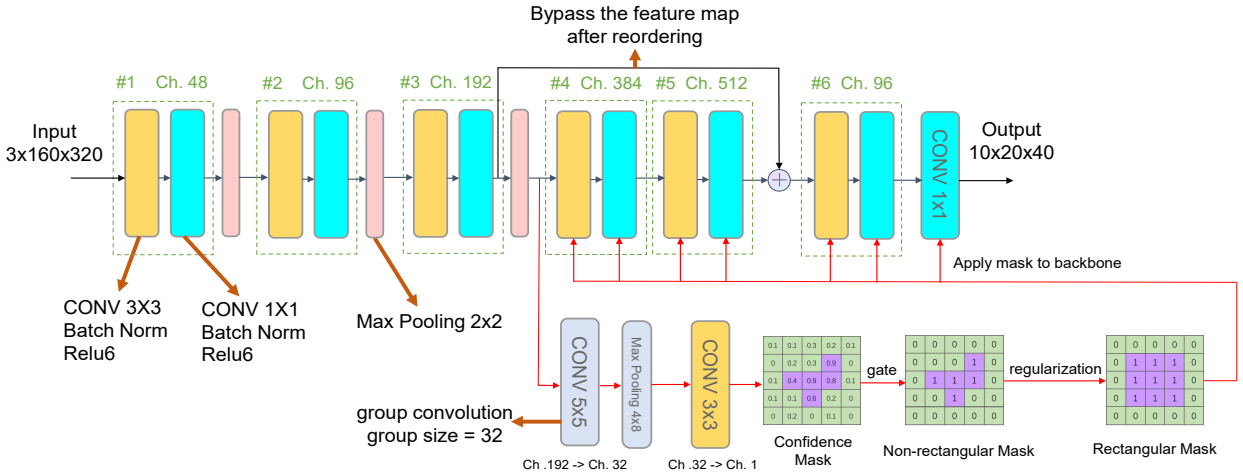

Fig. 3: The architecture of Mask-SkyNet. The new branch includes a two-layer fully convolutional network. The two layers are 5×5 group convolution with group size = 32 and 3×3 convolution. The max-pooling layer with stride 4×8 helps to downsample the feature map from 20×40 to mask shape 5×5. After passing the gate function and regularization, the mask is upsampled to the size of the feature map in the backbone and applied to 7 layers behind the branch.

The $upsample$ function is used to resize the mask shape from $a \times a$ to the size of the feature map and $\odot$ means Hadamard product.

As a case study, we use SkyNet [20] as the backbone. The backbone is generated by stacking six bundles as shown in Fig. 3. Each bundle contains a 3×3 depth-wise Conv layer [11], 1×1 point-wise Conv layer, batch normalization layer [44] and ReLU6. We add a new branch after the third bundle in SkyNet. We refer to this new model as Mask-SkyNet. By sharing convolution layers with the SkyNet backbone network, the new branch can extract preliminary features of the image. In this way, the confidence mask can be obtained through a two-layer fully convolutional network, which greatly reduces the number of network parameters.

B. Algorithm Innovations

- **Confidence Mask and Regions of Interest Generation.** As shown in Fig. 3, the output of the new branch is a confidence mask with each patch's score scaled to between 0 to 1 by $Sigmoid$ function. The higher the patch's score, the more likely that the object is in this patch. Then we set a threshold for each patch's score. If the score is above the threshold, we keep this patch and set its score to 1. Otherwise, we assume this patch to be background and set its score to 0. Then we select patches with a score of 1 as regions of interest for further computation and skip the computation on patches with a score of 0 in layers after the new branch.
- **All-Pass Mechanism.** To improve the robustness of mask generation and to avoid mis-prediction for hard-to-detect cases, if the score of all patches of a confidence mask does not exceed the threshold, we assume that the new branch fails to identify the location where the object may exist. In this case, the score of all patches in the mask is set to 1. We believe that this approach is conducive to improving IoU.
- **Two-stage Training Process.** To effectively train the new architecture, we come up with a two-stage training strategy. We first fix the weights in the backbone and only train the new branch. The new branch can learn information about the approximate location of the object in this stage. Then we fix the weights of the branch and fine-tune the weights of layers behind the branch in the backbone to make them adapt to the influence of applying the mask.

C. Hardware Innovations

- **Region of Interest Shape Regularization.** The convolution in our accelerator on FPGA is tile-based. If the whole tile is encapsulated by a 0-mask, the data loading and computation for the entire tile is skipped. If the tile is partly covered by the 0-mask, we will find out the non-zero patches in the mask and calculate them. However, the shape of regions of interest formed by the patches with a score of 1 in a mask may not be rectangular. In this case, every time before loading and calculation of one patch in a tile, we have to do a judgement based on the score of the patch. This judgement will introduce complex control logic in hardware. To avoid this problem, we propose to regularize the shape of regions of interest in the generated mask into rectangular as shown in Fig. 3. The rectangular regions of interest in the mask can then be directly mapped to the corresponding feature map regions by upsampling. Without extra judgement on the score of the patch, we can easily control where to calculate on FPGA.
- **Channel Shuffle.** To reduce the computation overhead and the number of parameters of the new branch, we propose to use group convolution. In the case study, the first group convolution reduces the number of channels from 192 to 32. In each group, 6 input channels correspond to 1 output channel. To reduce data movement between DDR and on-chip memory, we use the channel shuffle method [10]. We

use the pointer to find the location of data we need to read in DDR. The pointer points to the data address of each channel in the DDR, and it skips 6 channels every time it moves. Thus we can load the data we find into on-chip memory in a specific order.

D. Design Space Exploration

After applying the mask to the detection backbone, the execution time of layers before the new branch will occupy about 75% of the execution time of the whole network. This time imbalance will affect the overall performance of the accelerator. To avoid this imbalance, we do a design space exploration to find the best allocation scheme for DSPs to maximize the Mask-Net accelerator throughput with a fixed number of extra DSPs. A Mask-Net accelerator with a new branch can be divided into three parts: layers before the new branch, layers after the new branch and new branch layers. We assume that the number of DSP across three parts are N_{pre}, N_{post} and N_b respectively, and the total available DSP count is N_{total}. The relationship between them satisfies the following equation:

$$N_{total} = N_{pre} + N_{post} + N_b \tag{2}$$

To simplify our model, we assume that the computational speed is proportional to the number of DSP. Thus, the total execution time of network with mask and without mask after DSP redistribution can be calculated as follows:

$$T_m = \frac{N^*_{pre}}{N_{pre}} \times t_{mpre} + \frac{N^*_{post}}{N_{post}} \times t_{mpost} + \frac{N^*_b}{N_b} \times t_{nb} \tag{3}$$

$$T = \frac{N^*_{pre}}{N_{pre}} \times t_{pre} + \frac{N^*_{post}}{N_{post}} \times t_{post} \tag{4}$$

In these two equations, T_m and T are the execution time of network with mask and without mask respectively; N^*_{pre}, N^*_{post} and N^*_b represent DSP count before, after new branch and new branch itself after DSP redistribution; t_{mpre}, t_{mpost} and t_{nb} are the execution time of three parts in network with mask before DSP redistribution; t_{pre} and t_{post} are the execution time of layers before and after new branch in network without mask before DSP redistribution. The task is to find the value of N^*_{pre}, N^*_{post} and N^*_b to minimize T_m with fixed N_{total}. We can also get the speedup ratio r when T_m is at the minimum:

$$r = \frac{T}{T_m} \tag{5}$$

V. Experiments

A. Experimental Setup

Dataset Generation and Ground-truth Mask Label Collection. We train the Mask-Net architecture on three single object detection and tracking datasets: DAC-SDC [23], OTB100 [24] and UAV123 [25]. We use 93520 images in DAC-SDC dataset to train and 1000 images in the validation set to test our model. In UAV123 and OTB100 dataset, we shuffle all the images first and then split the training, validation and testing as 7:2:1 randomly. The mask shape we use in the experiment is 5×5 and threshold is 0.3. The ground-truth mask label used for training is generated based on the ground-truth annotations of datasets. In a ground-truth mask label, the value of patches occupied by the ground-truth annotation of the object is set to 1, and the value of other patches is set to 0.

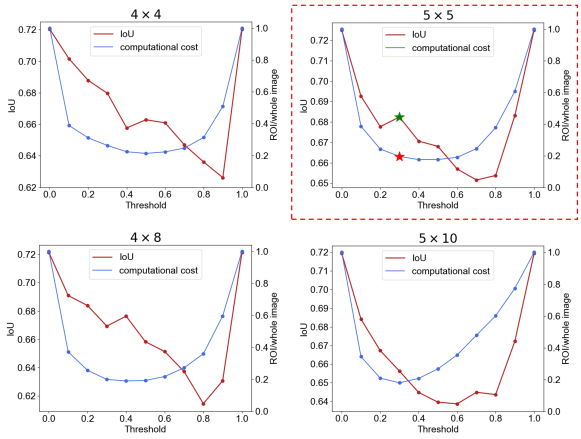

Fig. 4: The relationship between threshold, IoU and the ratio of regions of interest (ROI) divided by the whole image when using different mask shapes. We choose threshold = 0.3 and mask shape 5×5 in this case study. The data point we choose is highlighted with a star.

Evaluation Metrics. We use three evaluation metrics: IoU, throughput and energy consumption to evaluate the improvement of Mask-Net. We choose the detection backbone as the baseline of Mask-Net and design an FPGA accelerator based on SkyNet. Our accelerator is deployed on ZCU106 FPGA by using Vitis HLS 2018.3. During HLS design, we deliberately not to optimize the new branch and backbone, and keep the initiative interval (II) and parallelism of the modules the same before and after adding the new branch. We believe that doing so ensures the fairness of the comparison.

B. Experiment Key Results

Mask Shape and Threshold Study. We first study the impact of different mask shapes on Mask-SkyNet performance. We choose different thresholds from 0 to 1 and different mask shapes including 4×4, 5×5, 4×8, and 5×10. Our experiment is done on DAC-SDC dataset. Fig. 4 shows the relationship between threshold and IoU under different mask shapes. We find $\mathbf{5 \times 5}$ as the mask shape and **0.3** as the threshold can achieve a good balance between inference speed and accuracy and this setting also behaves well on two other datasets and backbones.

Mask quality. We further analyze the accuracy of the masked region proposals generated by Mask-SkyNet on DAC-SDC dataset. The experiment results show that 84.3% of proposed regions in the mask can completely cover the object while only 1.2% of region proposals are completely wrong. The high

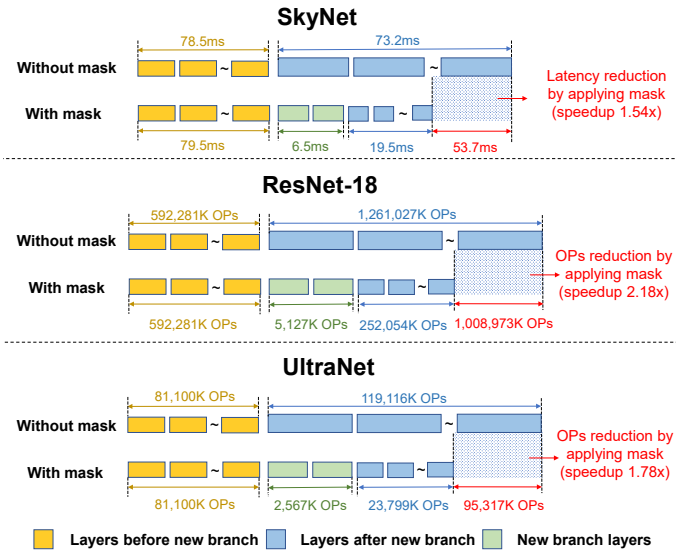

Fig. 5: C/RTL co-simulation result from Vitis. We estimate the execution time of different parts in the Mask-SkyNet accelerator on the DAC-SDC dataset from the waveform results. We also show the decrease of operations on ResNet-18 and UltraNet backbone. The operation (**OP**) here is the addition and multiplication in convolution layers.

accuracy of proposed regions means the new branch can predict the location of the object effectively.

Besides analyzing the accuracy of generated masked region proposals, we also apply the ground-truth mask to the SkyNet backbone before and after fine-tuning for comparison. Experiment results in Table. I show that using masked region proposals will only cause 5% $\sim$ 6% IoU loss.

	before FT	after FT
ground truth mask	0.7009	0.7249
mask generated	0.6654	0.6824

TABLE I: IoU after applying ground truth and generated masks. The results are from software and both weights and feature maps are float32. **FT** means fine-tuning.

	LUT	FF	BRAM	DSP
Available	230,400	460,800	312	1,728
Mask-SkyNet	69,960	70,548	209	416
SkyNet	49,554	58,203	209	329

TABLE II: Resource utilization on Xilinx ZCU106 FPGA. The clock frequency is 166MHz. Results are reported by Vitis HLS.

Software and Hardware Evaluation Results. We apply our Mask-Net technique on three different detection backbones: SkyNet [20], ResNet-18 [3] and UltraNet [22]. We then use three different single object detection and tracking datasets: DAC-SDC [23], OTB100 [24] and UAV123 [25] to evaluate our design. At software level, weights and feature maps are both 32-bit floating-point. The software evaluation results are shown in Table. III. We analyse IoU change before and after

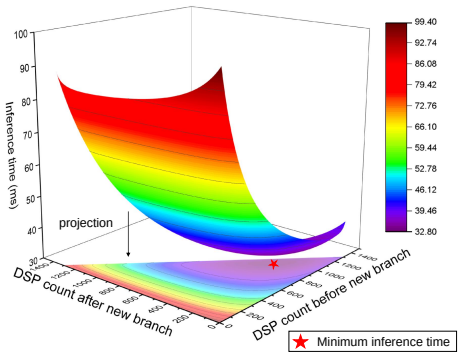

Fig. 6: DSP exploration space when extra DSP count is 1309. The DSP allocation scheme to achieve the least inference time is marked with a star.

mask application and the speedup. We also show the overhead of the new branch and the ratio of regions of interest (ROI) to the area of the input image. With IoU loss around 5%, the theoretical speedup can achieve 1.6$\times$ to 2.3$\times$ depending on the dataset and network architecture. Moreover, the computation overhead of the new branch can be ignored compared with the detection backbone.

We also design a hardware accelerator to deploy our case study design based on SkyNet backbone onto ZCU106 FPGA. The weights are 11 bits and feature maps are 9 bits. We use DAC-SDC, UAV123 and OTB100 dataset to test our design. The on-board results are shown in Table. IV, and resource utilization is shown in Table. II. By analyzing the utilization report from Vitis, we find that 73 of the 87 DSPs added come from the new branch.

Detailed Latency Breakdown. We divide our FPGA accelerator design into three separate parts: layers before new branch, layers after new branch and new branch itself as shown in Fig. 2. From C/RTL co-simulation waveform analysis, we find that the total execution time of the new branch is about 6.5ms every 4 images, which only takes about 6% of the total inference time in Mask-SkyNet. After applying the mask to backbone, we observe a 73.4% decrease in the execution time of the layers after the new branch. Detailed information about the execution time of different parts in Mask-SkyNet accelerator and the estimated operation decrease using ResNet-18 and UltraNet as detection backbone is shown in Fig. 5. It shows that applying mask can reduce the computation cost of layers after the new branch by about 70%, and by about 35% $\sim$ 50% for the total latency depending on network architecture and dataset.

Design Space Exploration Results. In design space exploration experiments, we choose SkyNet as our detection backbone and evaluate on DAC-SDC dataset. Vitis synthesis report of Mask-SkyNet accelerator shows that layers before the new branch utilize 320 DSPs, the new branch utilize 54 DSPs and layers after the new branch utilize 313 DSPs. Note that shared DSPs are repeatedly calculated in the number of DSPs

Backbone	Dataset	IoU without mask	IoU with mask	IoU Loss	Speedup	ROI	New Branch Overhead
SkyNet	DAC-SDC	0.7192	0.6824	5.12%	2.29×	19.4%	0.83%
	OTB100	0.8331	0.7997	4.01%	1.99×	28.8%	
	UAV123	0.7653	0.7238	5.42%	2.15×	23.2%	
ResNet-18	DAC-SDC	0.5840	0.5552	4.93%	2.18×	20.0%	0.28%
	OTB100	0.8214	0.8027	2.28%	1.84×	32.6%	
	UAV123	0.6978	0.6690	4.13%	2.13×	21.7%	
UltraNet	DAC-SDC	0.6112	0.5538	9.39%	1.78×	24.3%	1.30%
	OTB100	0.7845	0.7327	6.60%	1.66×	31.2%	
	UAV123	0.6602	0.6276	4.94%	1.64×	32.1%	

TABLE III: Software evaluation results when using different detection backbones and different datasets. The weights and feature maps are both float32. The speedup and new branch overhead are targeting Xilinx ZCU106 FPGA. The ROI (regions of interest) column represents the ratio of regions of interest occupied in the input image.

Dataset	Frequency	FPS	Energy per 1000 frames	Speedup	Energy Reduction
DAC-SDC	214 MHz	34.24	8.08J	1.35×	32.3%
	166 MHz	38.10	7.47J	1.35×	33.2%
	124 MHz	31.18	8.61J	1.37×	32.8%
OTB100	214 MHz	32.62	8.45J	1.30×	29.3%
	166 MHz	36.34	7.85J	1.28×	29.5%
	124 MHz	29.62	9.29J	1.30×	27.0%
UAV123	214 MHz	31.80	8.76J	1.27×	29.1%
	166 MHz	35.33	8.16J	1.39×	32.8%
	124 MHz	28.96	9.35J	1.31×	30.1%

TABLE IV: Hardware deployment result of Mask-SkyNet on ZCU106 FPGA. The weights are quantized to 9 bits and feature maps are quantized to 11 bits. *Note that the energy is estimated by PMBus in PYNQ image v2.4.

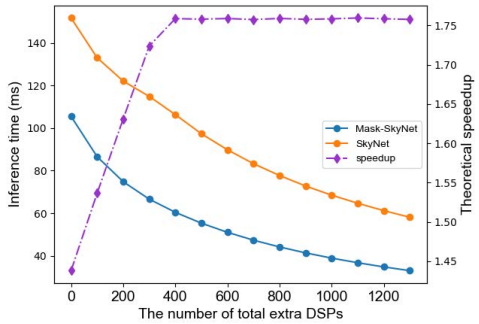

Fig. 7: The relationship between the number of extra DSPs, inference time and theoretical speedup. The theoretical speedup is Mask-SkyNet over SkyNet when allocating the same number of extra DSPs to both of the networks' backbones.

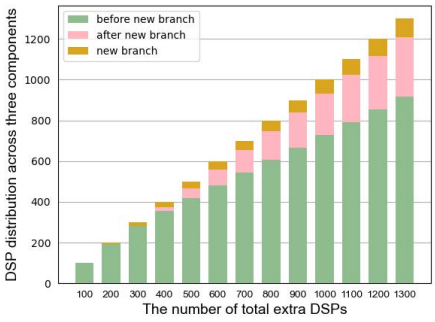

Fig. 8: The DSP distribution across three parts with different total number of extra DSPs. Most of extra DSPs are added to layers before the new branch to accelerate the generation of the mask.

of three parts. We can also get the execution time of three parts from C/RTL co-simulation results. Thus, the values of some predefined variables in Eq. 3 and Eq. 4 are: $N_{pre} = 320$, $N_{post} = 313$, $N_b = 54$, $t_{mpre} = 79.5ms$, $t_{mpost} = 19.5ms$, $t_{nb} = 6.5ms$, $t_{pre} = 78.5ms$ and $t_{post} = 73.2ms$. Two equations can be rewritten as follows:

$$T_m = \frac{79.5}{1 + A/320} + \frac{19.5}{1 + B/313} + \frac{6.5}{1 + C/54} \tag{6}$$

$$T = \frac{78.5}{1 + A/320} + \frac{73.2}{1 + B/313} \tag{7}$$

where A, B and C are the number of total extra DSPs in layers before the new branch, layers after the new branch and new branch respectively.

The exploration space when the number of extra DSPs is 1309 is shown in Fig. 6. The relationship between the number of extra DSPs, inference time and the theoretical speedup is shown in Fig. 7. The theoretical speedup can achieve 1.76× with balanced computation between three parts. DSP allocation scheme that maximizes accelerator throughput when using a fixed number of additional DSPs is shown in Fig. 8.

VI. Conclusions

In this work, we present a lightweight and hardware-friendly object detection network with masked region proposals to avoid redundant computation. Our design, **Mask-Net**, has a small computational overhead, is compatible with common object

detection backbones and works well in different scenarios. The software and on-board evaluation exhibit an increase in throughput and decrease in energy consumption with slight accuracy loss compared with detection backbones without masked region proposals. We also demonstrate its theoretical maximum speedup by doing a design space exploration about the DSP allocation scheme.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1):127–138, 2016.

[5] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W Keckler, and William J Dally. Scnn: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH Computer Architecture News*, 45(2):27–40, 2017.

[6] Xiaofan Zhang, Junsong Wang, Chao Zhu, Yonghua Lin, Jinjun Xiong, Wen-mei Hwu, and Deming Chen. Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[7] Qin Li, Xiaofan Zhang, JinJun Xiong, Wen-mei Hwu, and Deming Chen. Implementing neural machine translation with bi-directional gru and attention mechanism on fpgas using hls. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 693–698, 2019.

[8] Yijin Guan, Guangyu Sun, Zhihang Yuan, Xingchen Li, Ningyi Xu, Shu Chen, Jason Cong, and Yuan Xie. Crane: Mitigating accelerator under-utilization caused by sparsity irregularities in cnns. *IEEE Transactions on Computers*, 69(7):931–943, 2020.

[9] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[10] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[12] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

[13] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[14] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pages 2285–2294. PMLR, 2015.

[15] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. 2011.

[16] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4820–4828, 2016.

[17] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016.

[18] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.

[19] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[20] Xiaofan Zhang, Haoming Lu, Cong Hao, Jiachen Li, Bowen Cheng, Yuhong Li, Kyle Rupnow, Jinjun Xiong, Thomas Huang, Honghui Shi, et al. Skynet: a hardware-efficient method for object detection and tracking on embedded systems. *Proceedings of Machine Learning and Systems*, 2:216–229, 2020.

[21] Jihao Liu, Ming Zhang, Yangting Sun, Boxiao Liu, Guanglu Song, Yu Liu, and Hongsheng Li. Fnas: Uncertainty-aware fast neural architecture search. *arXiv preprint arXiv:2105.11694*, 2021.

[22] Zhan Kang, Guo Junnan, Song Bingyan, Zhang Wenbo, and Bao Zhenshan. Ultranet : A fpga-based object detection for the dac-sdc 2020, 2020.

[23] Xiaowei Xu, Xinyi Zhang, Bei Yu, X Sharon Hu, Christopher Rowen, Jingtong Hu, and Yiyu Shi. Dac-sdc low power object detection challenge for uav applications. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[24] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[25] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer, 2016.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[27] NI Glumov, EI Kolomiyetz, and VV Sergeyev. Detection of objects on the image using a sliding window mode. *Optics & Laser Technology*, 27(4):241–249, 1995.

[28] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, 38(4):814–830, 2015.

[29] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *2011 international conference on computer vision*, pages 1879–1886. IEEE, 2011.

[30] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[31] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 73–80. IEEE, 2010.

[32] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012.

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[35] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[36] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[37] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906*, 2021.

[38] Yichi Zhang, Ritchie Zhao, Weizhe Hua, Nayun Xu, G Edward Suh, and Zhiru Zhang. Precision gating: Improving neural network efficiency with dynamic dual-precision activations. *arXiv preprint arXiv:2002.07136*, 2020.

[39] Weizhe Hua, Yuan Zhou, Christopher De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *arXiv preprint arXiv:1805.12549*, 2018.

[40] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *arXiv preprint arXiv:2010.05300*, 2020.

[41] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4438–4446, 2017.

[42] Zekun Hao, Yu Liu, Hongwei Qin, Junjie Yan, Xiu Li, and Xiaolin Hu. Scale-aware face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6186–6195, 2017.

[43] Huiyu Wang, Aniruddha Kembhavi, Ali Farhadi, Alan L Yuille, and Mohammad Rastegari. Elastic: Improving cnns with dynamic scaling policies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2258–2267, 2019.

[44] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.