# Irrelevant Pixels are Everywhere: Find and Exclude Them for More Efficient Computer Vision

Caleb Tung[1], Abhinav Goel[1], Xiao Hu[1], Nick Eliopoulos[1], Emmanuel S. Amobi[2], George K. Thiruvathukal[2], Vipin Chaudhary[3], Yung-Hsiang Lu[1]
[1]Purdue University, [2]Loyola University Chicago, [3]Case Western Reserve University
{tung3, goel39, hu440, neliopou, yunglu}@purdue.edu, eamobi@luc.edu, gkt@cs.luc.edu, vxc204@case.edu

2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS) | 978-1-6654-0996-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/AICAS54282.2022.9870012

***Abstract*—Computer vision is often performed using Convolutional Neural Networks (CNNs). CNNs are compute-intensive and challenging to deploy on power-contrained systems such as mobile and Internet-of-Things (IoT) devices. CNNs are compute-intensive because they indiscriminately compute many features on all pixels of the input image. We observe that, given a computer vision task, images often contain pixels that are irrelevant to the task. For example, if the task is looking for cars, pixels in the sky are not very useful. Therefore, we propose that a CNN be modified to only operate on relevant pixels to save computation and energy. We propose a method to study three popular computer vision datasets, finding that 48% of pixels are irrelevant. We also propose the focused convolution to modify a CNN's convolutional layers to reject the pixels that are marked irrelevant. On an embedded device, we observe no loss in accuracy, while inference latency, energy consumption, and multiply-add count are all reduced by about 45%.**

***Index Terms*—computer vision, low-power devices, embedded systems, datasets**

## I. Introduction

Convolutional Neural Networks (CNNs) are known for their high accuracy at many computer vision tasks. However, this accuracy comes at a cost: CNNs are compute-intensive. ResNet, a popular computer vision CNN for performing the relatively simple task of image classification, needs to compute nearly 30 million parameters across all the pixels in the input image [1]. This high compute requirement is typically satisfied by running the CNN on powerful Graphics Processing Units (GPUs) or other hardware accelerators. However, *low-power systems* (i.e., Internet-of-Things (IoT), mobile, and embedded devices) often impose power and memory constraints that make it challenging to deploy CNNs on them [1].

To lessen the computation of a CNN on low-power systems, many methods opt to reduce the sheer magnitude of CNN parameters. These include *pruning* [2] and *quantization* [3] to cut away redundant parameters or reduce precision, and further include *knowledge distillation* [4] and *neural architecture search* [5] to train small CNNs with fewer parameters.

These methods all improve efficiency by changing the computer vision model itself; we instead propose changing the *input* to the model. CNNs operate indiscriminately on every single pixel in the input; therefore, if we reduce the input, we reduce the computation.

In this paper, we propose that input images have many pixels that can be deleted, and that by doing so, a CNN would save energy. We confirm this idea by creating methods to (1) identify that three different popular computer vision datasets all contain many such pixels and (2) demonstrate the energy and inference speed improvements possible by using our focused convolution to delete those pixels.

An *irrelevant pixel* (Figure 1) (formally defined in Section II) is one that is not useful for the computer vision task (e.g., a building's pixels are not useful when looking for cars). We use depth maps to convert ground truth labels into irrelevant pixel maps, showing that in the Microsoft Common Objects in Context (COCO, 164,000+ images) [6], Multi-Object Tracking Challenge (MOT Challenge, 5,500 images) [7], and PASCAL VOC [8] (17,900+ images) datasets, roughly 48% of pixels are irrelevant.

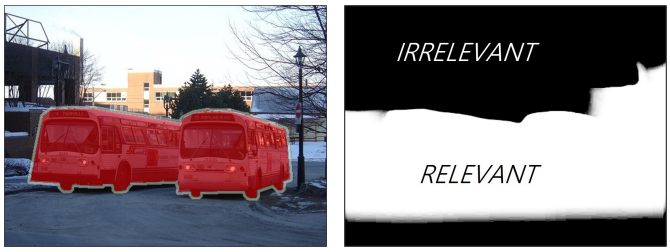
Given the irrelevant pixels, we further experiment to find that explicitly excluding those pixels during inference significantly reduces a CNN's compute expenses, saving energy and speeding inference. Our proposed *focused convolution* technique modifies a CNN such that the model itself can exclude pixels marked as irrelevant while still using the same parameters and General Matrix-Multiplication convolutional technique. Similarly inspired work includes Uber's SBNet and its variations; SBNet does convolution in ResNet-sized blocks and tries to fit the blocks into the relevant pixels [9], [10]. Those other methods still require the model to be changed and re-trained, ours does not. Replacing normal convolution with the focused convolution on the three datasets reduces multiply-add operations, energy consumption, and inference latency by about 45% in two popular CNNs.

This paper's contributions: (1) use depth maps to find how many pixels are irrelevant in three popular computer vision datasets, and (2) demonstrate that CNNs can save on inference latency and energy consumption by excluding irrelevant pixels from computation and only performing operations on relevant ones, via our focused convolution.

## II. Finding Irrelevant Pixels in Datasets

We propose a method to study datasets for irrelevant pixels. We find that irrelevant pixels are commonly found in the COCO, PASCAL VOC, and MOTChallenge datasets.

This project is supported in part by NSF OAC-2104709 and NSF OAC-2107020. Any opinions, findings, conclusions, or recommendations expressed in this material by the authors do not necessarily reflect the views of the sponsors.




(a) (b)

Figure 1: Example of irrelevant pixels in a PASCAL VOC dataset image. **(a)** The shuttle buses are highlighted in red. **(b)** Our method shows that a significant number of pixels (black) are irrelevant, providing little utility, even though CNNs waste computation on them.

(a) (b)

Figure 2: Ground truth pixels alone **(a)** are not always a sufficient amount of relevant pixels for a CNN to make good predictions (red box is inaccurate, misses second bus); contextual pixels at similar depth levels are also needed **(b)** for CNN accuracy (two good detections).

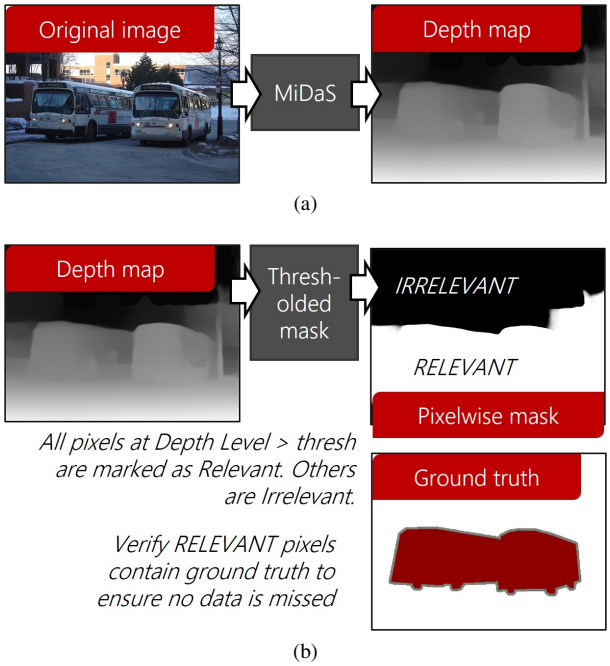



Figure 3: This is how to count the number of irrelevant pixels in datasets: First, use MiDaS to generate the image's depth map **(a)**. Next, threshold the depth map, producing a small Relevant Pixels region **(b)**. Verify against ground truth to ensure all important data is captured.

### A. Counting Irrelevant Pixels with Depth Maps

Our study defines a dataset's relevant pixels as: *all pixels that comprise the dataset ground truth objects and their associated depth levels.* In Figure 1a, the ground truth pixels are represented by the red area around the shuttle buses. The final pixelwise map of relevant pixels, shown in Figure 1b, is generated using depth level thresholding (explained below) and includes additional pixels.

Ground truth pixels alone do not represent all relevant pixels. A typical CNN uses not only the pixels comprising the object (Figure 2a), but also the *pixels from the surrounding area of an object* (Figure 2b) to extract features that contextualize and understand the object [11]. To properly include these contextual pixels, our method uses *depth maps*. Shown in Figure 3a, an image's depth map represents each pixel in the image with one value, referred to as the pixel's *depth level*, denoting how far away from the camera that pixel is.

Pixels that have similar depth levels to a given object on the ground are known to provide useful, contextual pixels that surround the object [12]. For example, a car will be at a similar depth level as its context: the road on which it drives. Therefore, by thresholding the depth level appropriately, we can capture all the relevant pixels: both the contextual pixels and their associated objects.

We generate depth maps with "MiDaS," a depth estimation model by Ranftl, et al. [13] MiDaS is chosen from among other monocular depth estimation techniques because it is uniquely trained using a mix of diverse datasets, dramatically improving its performance on Microsoft COCO, our largest target dataset.

Our method thresholds the depth map to create a binary, pixelwise mask of relevant and irrelevant pixels, such that the relevant pixel region encapsulates the ground truth (see Figure 3). Using the observation that camera focal lengths tend to cause pictures to be taken at mid-range depth levels [11], we threshold an image's MiDaS depth map according to the mid-range of its distribution, filtering out pixels at short-range and long-range depth levels. The filtered pixels are considered irrelevant and the remaining pixels are considered relevant. We verify this binary, pixelwise mask against the ground truth labels: if the mask's relevant-marked region contain the ground truth, then we assume all ground truth and associated contextual pixels are captured, and we record the number of irrelevant VS relevant pixels. If not, the threshold is expanded and the process is repeated.

### B. Exploring Datasets for Irrelevant Pixels

In our study, we use our depth map method to count irrelevant pixels in each image in the three datasets, totaling nearly 200,000 images. Results are shown in Figure 4. Irrelevant pixels are quite common in the three datasets. On average, 42% of all pixels are irrelevant in PASCAL VOC, 49% are irrelevant in MOT Challenge, and 45% are irrelevant in COCO. This suggests that CNNs are wasting significant amounts of compute resources on embedded devices by computing convolutions on all those irrelevant pixels.

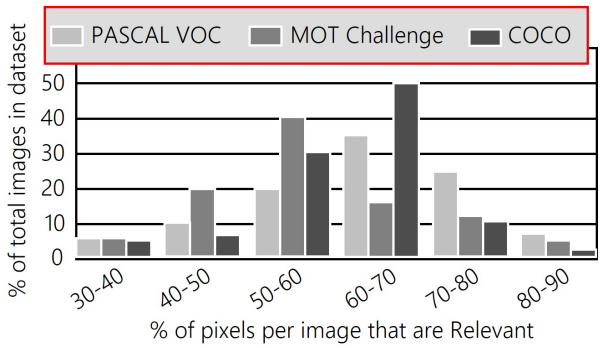



Figure 4: Average percentage of pixels that are relevant. Less than 1% of images contain 10-30% or 90-100% useful pixels and are not shown on the graph. As shown, most images contain 50%-60% relevant (40-50% irrelevant pixels). This means that in most images, 40-50% of pixels do not contribute to useful CNN computations and, therefore, can be excluded from processing.

## III. Excluding Irrelevant Pixels in Datasets

To demonstrate that being able to exclude irrelevant pixels can significantly save computer vision energy consumption and inference latency, we propose the *focused convolution* technique. In typical CNN computer vision models, the 2D-convolutional layer is responsible for up to 80% of the computations performed by the neural network [1]. Standard convolution layers are not designed to exclude irrelevant pixels. Instead, they operate on all the input pixels, wasting computation on irrelevant pixels. Because CNNs have multiple convolutional layers in succession, this waste is repeated across multiple layers, compounding the negative impact on the model's energy use and speed. The focused convolution reduces this waste by excluding any pixels marked irrelevant by some assumed oracle - in this case, our relevant VS irrelevant pixelwise masks generated in Section II-B.

### A. Using Focused Convolutions to Exclude Irrelevant Pixels

Our focused convolution improves the popular General Matrix Multiplication (GEMM) technique [14]. The GEMM technique reduces a sequential, sliding-window 2D-convolution to a parallelized, matrix-multiplication operation; matrix multiplications are heavily optimized on modern computers [14]. GEMM (Figure 5a) uses a procedure called *im2col* to convert patches of the 3D input tensor (e.g., an RGB image) into the columns of a 2D matrix, and then multiplies the matrix by the convolutional weights to retrieve the results of convolution.

The focused convolution enables the *im2col* procedure to accommodate a pixelwise mask (such as those generated in Section II-B) indicating whether a pixel is relevant or not. If a patch of pixels is labeled irrelevant, then the modified *im2col* will not convert the patch into a column of the matrix. Thus, those irrelevant pixels get excluded entirely by the matrix multiplication, as shown in Figure 5b.

Because the focused convolution is simply designed to accommodate the pixelwise masks as inputs, it does not need to change the model itself. It can be used in any pretrained

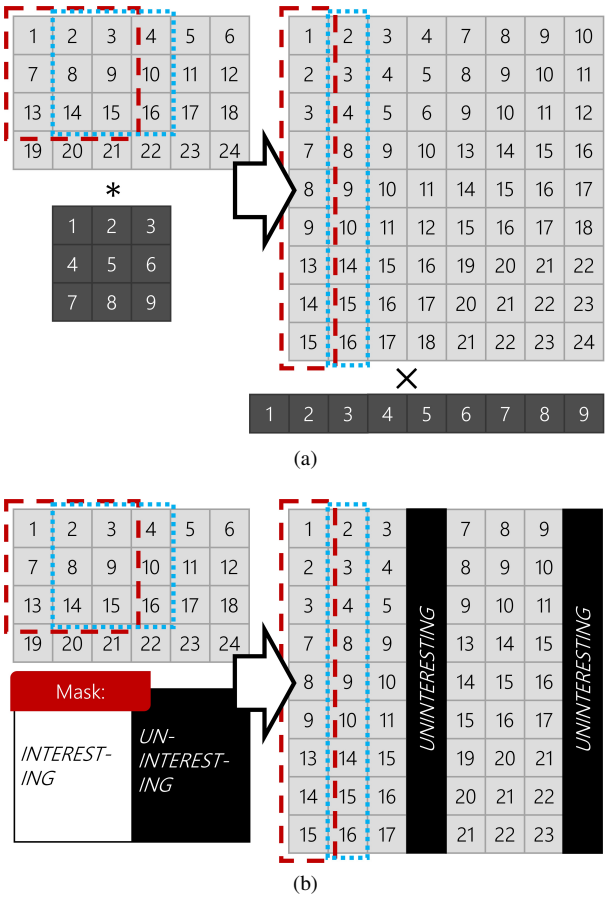



Figure 5: **(a)** Traditional GEMM convolution uses the *im2col* procedure to convert a $1\times4\times6\times1$ input tensor (light gray), assuming a stride-1, $3\times3$ weight kernel (dark gray), to a $9\times8$ matrix for matrix multiplication. Each $9\times1$ column of the matrix (e.g., red, blue dashed line) represents one $3\times3$ patch from the input tensor. **(b)** The proposed *focused convolution* modifies *im2col* exclude patches deemed irrelevant by an oracle-generated pixelwise mask. Thus, the final $9\times6$ matrix will have fewer columns than that of traditional GEMM, resulting in less computation on embedded devices.

CNN, requiring no additional training. It can replace existing convolution layers without changing the weights and biases. If no pixelwise mask is supplied, the focused convolution behaves identically to a standard convolution.

For a given image and convolutional layer, we can count the number of operations needed to complete a GEMM convolution's matrix multiplication. Assume the input has dimensions $B_I \times C_I \times H_I \times W_I$ (batch, channels, height, width), and the convolutional weights has dimensions $S_W \times S_W$ (side lengths) with stride $s$ and no padding. A normal *im2col* would convert each channel of the input to $\frac{H_I-S_W}{s}\frac{W_I-S_W}{s}$ columns of $S_W S_W$ pixels. That amounts to multiplying the weights with a $S_W S_W \times C_I \frac{(H_I-S_W)}{s}\frac{W_I-S_W}{s}$ matrix, $B_I$ times. That is a total of $B_I \frac{C_I(H_I-S_W)}{s}\frac{W_I-S_W}{s} S_W S_W$ multiply/add operations.

If we convert the GEMM convolution to a focused convolution, we save matrix multiplication operations by excluding columns of irrelevant pixels. As we discovered in Section II-B, an average of 48% of pixels are irrelevant in popular datasets.

If the region of relevant pixels is a rectangle that takes up $p\%$ of the pixels, then the focused convolution would produce only $p\% \frac{H_I - S_W}{s} \frac{W_I - S_W}{s}$ columns, resulting in a $100 - p\%$ reduction in operations. Therefore, the focused convolution's energy and latency improvements is a function of the linear relationship between number of irrelevant pixels and image size.

### B. Evaluating the Compute Savings of Focused Convolutions

We implement and test the focused convolution using PyTorch on a Raspberry Pi 3 (average power 5 W). We compare the focused convolution with a normal PyTorch GEMM convolution. Excluding pixels is beneficial on more powerful hardware, too - we compare the focused convolution with an Intel MKL-optimized convolution [15] on a Intel Core i7 CPU (average power 28 W). For input images, we use the MOT Challenge, COCO, and PASCAL VOC. We exclude the pixels deemed irrelevant by our method from Section II-A.

Our tests comprise of two popular computer vision CNNs designed for use on low-power devices: EfficientDet [16] and SSD-Lite [17]. We use PyTorch's pretrained weights, and we replace each CNN's convolutional layer with a focused convolutional layer. The pixelwise, binary Relevant-VS-Irrelevant masks we generated in Section II-B are propagated through the CNN by scaling the mask's size for each layer.

Table I summarizes the observed energy consumption/inference latency improvements from the focused convolutions. As shown, we see that equipping a CNN with focused convolution allows it to dramatically reduce its computation expense by excluding irrelevant pixels on images from popular datasets. On Intel CPU, the MKL optimizations allow a fullsize, "wasteful" convolution to operate comparably to the focused convolution, but such optimizations are unavailable on low-power devices, so the focused convolution is preferable.

Finally, we verify the focused convolution-equipped CNN's detection accuracy remains identical to the original CNN.

### C. Assumptions, Challenges, and Opportunities

This paper assumes that including all pixels at the same depth level as ground truth accurately captures all related pixels. This may not be true when a camera's focus is extended (e.g. wide, scenic shots). Still, our technique is demonstrably sufficient for three major datasets. Additionally, depth mapping is computationally intensive, so future irrelevant pixel generation will need to either be offloaded or rapidly generated.

## IV. Conclusion

We observe that in computer vision, CNNs are compute-heavy because they waste time looking at irrelevant pixels. We propose a thresholded depth mapping technique to study modern computer vision datasets for irrelevant pixels. We find that an average of 42%, 49%, and 45% of pixels per image are irrelevant, for three popular datasets (PASCAL VOC, MOT Challenge, and COCO, respectively). We further propose significantly reducing a convolutional layer's multiply-add operations, energy consumption, and inference latency with our focused convolution: this modifies the layer's standard GEMM operations to ignore irrelevant pixels. Ignoring the irrelevant pixels we computed on the three datasets, we find that multiply-add operations are reduced by an average of 48.3%, energy consumption by 42.7%, and inference latency by 47.4% for two popular object detection CNNs on an embedded device.

| | | *MOT2015* | | *COCO* | | *PASCAL VOC* | |
|---|---|---|---|---|---|---|---|
| | | *ED* | *SL* | *ED* | *SL* | *ED* | *SL* |
| *Number of Mult-Add Operations (M/inference)* | | | | | | | |
| | Normal | 384.5 | 483.6 | 384.5 | 483.6 | 384.5 | 483.6 |
| | **Focused** | **196.1** | **246.8** | **211.4** | **266.0** | **223.0** | **280.4** |
| *Inference Latency (s/inference)* | | | | | | | |
| *RPi (5W)* | Normal | 2.10 | 2.26 | 2.00 | 2.33 | 2.06 | 2.29 |
| | **Focused** | **1.11** | **1.30** | **1.33** | **1.51** | **1.47** | **1.56** |
| *Intel (28W)* | Normal | 0.25 | 0.28 | 0.25 | 0.29 | 0.25 | 0.28 |
| | MKL | 0.18 | 0.19 | 0.18 | 0.20 | 0.18 | 0.20 |
| | **Focused** | **0.17** | **0.18** | **0.18** | **0.20** | **0.19** | **0.20** |
| *Energy Consumption (J/inference)* | | | | | | | |
| *RPi (5W)* | Normal | 10.22 | 11.80 | 10.15 | 11.81 | 10.20 | 10.90 |
| | **Focused** | **5.60** | **6.11** | **6.71** | **7.50** | **7.44** | **7.80** |
| *Intel (28W)* | Normal | 6.61 | 7.39 | 6.45 | 7.42 | 6.69 | 7.81 |
| | MKL | 5.18 | 5.09 | 5.09 | 5.61 | 5.23 | 5.60 |
| | **Focused** | **4.76** | **5.04** | **5.10** | **5.60** | **5.29** | **5.62** |

Table I: By excluding irrelevant pixels from popular datasets (MOTChallenge, COCO, PASCAL VOC), CNNs equipped with our **Focused** convolutions outperform those without (Normal). On an Intel CPU, MKL-optimized convolutions are comparable in performance to focused convolutions.

RPi: Raspberry Pi 3, Intel: Intel Core i7 CPU, ED: EfficientDet, SL: SSD-Lite, MKL: using Intel MKL optimized convolution.

## References


[1] G. K. Thiruvathukal *et al.*, *Low-Power Computer Vision: Improve the Efficiency of Artificial Intelligence*. CRC Press, 2022.

[2] H. Wang *et al.*, "Structured Pruning for Efficient Convolutional Neural Networks via Incremental Regularization," *IEEE JSTSP*, May 2020.

[3] M. Courbariaux *et al.*, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *arXiv:1602.02830*, 2016.

[4] B. Peng *et al.*, "Correlation Congruence for Knowledge Distillation," in *2019 IEEE/CVF ICCV*.

[5] M. Tan *et al.*, "MnasNet: Platform-Aware Neural Architecture Search for Mobile," in *2019 IEEE/CVF CVPR*.

[6] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *ECCV 2014*.

[7] L. Leal-Taixé *et al.*, "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking," *arXiv:1504.01942*, 2015.

[8] M. Everingham *et al.*, "The Pascal Visual Object Classes (VOC) Challenge," *Springer IJCV*, Jun. 2010.

[9] M. Ren *et al.*, "SBNet: Sparse Blocks Network for Fast Inference," in *2018 IEEE/CVF CVPR*.

[10] T. Verelst *et al.*, "Dynamic Convolutions: Exploiting Spatial Sparsity for Faster Inference," in *2020 IEEE/CVF CVPR*.

[11] K. Gauen *et al.*, "Comparison of Visual Datasets for Machine Learning," in *2017 IEEE IRI*.

[12] S. Song *et al.*, "Semantic Scene Completion From a Single Depth Image," in *2017 IEEE/CVF CVPR*.

[13] R. Ranftl *et al.*, "Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer," *IEEE TPAMI*, 2020.

[14] A. Anderson *et al.*, "Low-memory GEMM-based convolution for deep neural networks," *arXiv:1709.03395*, 2017.

[15] E. Wang *et al.*, "Intel Math Kernel Library," in *High-Performance Computing on the Intel® Xeon Phi™: How to Fully Exploit MIC Architectures*, Springer International Publishing, 2014.

[16] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *2020 IEEE/CVF CVPR*.

[17] M. Sandler *et al.*, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF CVPR*, 2018.