IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 31, 2018, accepted August 31, 2018, date of publication September 13, 2018, date of current version October 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2869929

Optimum Selection of DNN Model and Framework for Edge Inference

DELIA VELASCO-MONTERO, JORGE FERNÁNDEZ-BERNI, RICARDO CARMONA-GALÁN, (Senior Member, IEEE), AND ÁNGEL RODRÍGUEZ-VÁZQUEZ, (Fellow, IEEE)
Institute of Microelectronics of Seville, Universidad de Sevilla-CSIC, 41092 Seville, Spain

Corresponding author: Delia Velasco-Montero (delia@imse-cnm.csic.es)

This work was supported in part by the Spanish Government MINECO (European Region Development Fund, ERDF/FEDER) under Project TEC2015-66878-C3-1-R, in part by Junta de Andalucía CEICE under Project TIC 2338-2013, and in part by EU H2020 MSCA ACHIEVE-ITN under Grant 765866.

ABSTRACT This paper describes a methodology to select the optimum combination of deep neural network and software framework for visual inference on embedded systems. As a first step, benchmarking is required. In particular, we have benchmarked six popular network models running on four deep learning frameworks implemented on a low-cost embedded platform. Three key performance metrics have been measured and compared with the resulting 24 combinations: accuracy, throughput, and power consumption. Then, application-level specifications come into play. We propose a figure of merit enabling the evaluation of each network/framework pair in terms of relative importance of the aforementioned metrics for a targeted application. We prove through numerical analysis and meaningful graphical representations that only a reduced subset of the combinations must actually be considered for real deployment. Our approach can be extended to other networks, frameworks, and performance parameters, thus supporting system-level design decisions in the ever-changing ecosystem of embedded deep learning technology.

INDEX TERMS Benchmarking, deep learning, convolutional neural networks, edge inference, embedded vision, high-level specifications.

I. INTRODUCTION

Deep Learning [1] (DL) is revolutionizing computer vision. One of its most prominent assets is that it unifies the various approaches existing in classical computer vision for a number of tasks: object detection and localization, image recognition, segmentation, optical flow, etc. A common processing architecture for all these tasks facilitates the exploitation of a suitable software-hardware infrastructure across multiple application frameworks. This unification comes with another crucial advantage, i.e. much higher accuracy [2]. It is enabling the commercialization of vision-based products that were previously restricted to prototype labs because of lack of precision in real scenarios.

On the flip side, the computational power required for Deep Neural Networks (DNNs) is notably larger than for classical computer vision algorithms. This has an impact on the practical realization of embedded systems [3], where form factor, cost, energy consumption and hardware resources are key parameters. Indeed, a major challenge these days is to achieve the energy efficiency of traditional algorithms while keeping the inference accuracy of DNNs [4]. This would definitely speed up the adoption of vision technologies in massive markets [5].

The interest of both industry and academia in DL is spurring the development of a myriad of software environments and hardware platforms tailored for DNNs [6]. Likewise, new network models are reported almost on a daily basis. Each of these components comes along with claims about enhancements in certain aspects of performance, ease of use, compatibility etc. While this variety in principle covers a wide spectrum of specifications, there are no guidelines for system designers to perform an optimal selection meeting prescribed application-level requirements.

In this paper, we propose a methodology to establish those guidelines. Firstly, we analyze 24 combinations of popular network models and software frameworks from a practical point of view. We have implemented all of them in a Raspberry Pi 3 (RPi) Model B [7], extracting three key performance parameters, namely accuracy, throughput and power consumption. The measurements reveal that we are

not far from performance levels enabling low-cost pervasive visual intelligence. We then define a Figure of Merit (FoM) encoding the relative weight of each parameter according to high-level requirements. We show that a proper numerical treatment and graphical representation of this FoM expedite the assessment for optimum model/framework selection. The results demonstrate that only a reduced subset of the analyzed combinations performs the best in at least one point of the exploration domain.

In addition to the proposed methodology itself, the main contributions of this paper are:

1) Comprehensive analysis and comparison of performance figures from state-of-the-art DNN models and frameworks deployed on an off-the-shelf low-cost embedded computer. Details of their implementations are provided.
2) Definition of a FoM that incorporates the possibility of weighing particular performance metrics according to application-level specifications.
3) Procedure to treat the numerical values of the FoM and graphically represent them for rapid identification of both optimal and worst model/framework combinations.

The rest of the paper is organized as follows: we review related work in Section II. Section III highlights the most relevant aspects about the DL technological components benchmarked in our study. In Section IV we report experimental results. Section V introduces the proposed FoM and the performance evaluation based on it. The outcome of this evaluation is discussed in Section VI. We finish the manuscript with the conclusions of our work in Section VII.

II. RELATED WORK

These days, benchmarking is crucial for DL-based embedded systems. A non-optimal selection of DL components within the vast – and growing – spectrum of possibilities could lead to the apparent impossibility of fulfilling prescribed application requirements. In this section, we briefly review relevant previous works on framework benchmarking and DNN evaluation for edge inference. We also stress where our contributions lie in.

A. FRAMEWORK BENCHMARKING

Some comparative analysis focused on DL software frameworks have been reported in the literature. In [8], five frameworks are evaluated in terms of speed and accuracy but only for a simple DNN model trained for recognition of handwritten digits. In the present paper, we assess six models for 1000-category image recognition running on four frameworks. Throughput is the only metric concerning inference measured in [9], also for five frameworks. Some figures about GPU memory usage are reported as well in this work. Two inference models, namely LeNet for digit recognition and AlexNet for image recognition, are benchmarked. In our case, we report measurements of three performance metrics for all of the considered model/framework pairs. The study in [10] is extensive, encompassing CPU and GPU analysis, again for five frameworks. Six network models for digit and image recognition are assessed. However, the benchmarked high-end hardware platforms are suitable for cloud-based services rather than for edge inference. None of these works incorporate prescribed high-level specifications in their analysis.

B. DNN EVALUATION

A thorough analysis of up to 14 state-of-the-art DNN architectures for image recognition is described in [11]. This analysis adds memory and operation count to the three metrics of our study, i.e. accuracy, throughput and power consumption. However, these metrics are measured for a single software framework supporting the execution of all the models. Moreover, the benchmarked hardware platform is still expensive and consumes too much power – over 10 watts – to consider massive edge deployment. Finally, no weighing of the different metrics according to application requirements is proposed.

If we specifically focus on the low-cost embedded platform benchmarked in our work, i.e. RPi, some embedded visual applications have been proved to be feasible on the basis of CNNs [12], [13], [14], [15], [16]. Performance metrics useful for each particular application are reported, such as accuracy, computation time or power consumption. Nevertheless, these metrics are measured once the application is designed and one or more CNNs are carrying out inference on a particular software framework. There is no methodology to extrapolate the results and conclusions obtained in these cases to other applications.

Lastly, we must single out the benchmarking reported in [17] as it exclusively deals with low-cost low-power hardware platforms, including RPi. Five network models for image recognition are evaluated when performing inference on two different software frameworks. The measured metrics are throughput and power consumption. No further analysis is provided, apart from selecting a particular model/framework/platform combination according to the raw numerical values of these two metrics.

III. BENCHMARKING ENVIRONMENT

A. HARDWARE PLATFORM

The performance of DL models and frameworks was assessed on the Raspberry Pi 3 Model B. Despite its limited resources, this low-cost embedded platform features enough computational power for real-time DNN inference. Its Broadcom BCM2837 System-on-a-Chip (SoC) comprises a Quad Core ARM Cortex-A53 1.2GHz 64-bit CPU that can work at frequencies ranging from 700 MHz up to 1.2 GHz. Its instantaneous value depends on the policy set by the user and the operation conditions: CPU load, temperature, etc. The system incorporates 1GB RAM LPDDR2 at 900MHz. We also endowed it with non-volatile storage capacity through a 64GB class-10 microSD card where we installed Raspbian

OS [18]. The RPi includes a Broadcom VideoCore IV GPU, but we could not exploit it since the DL software tools considered in this study only allow to run the models on the CPU of the RPi at the present time. We therefore allocated as much memory as possible for the CPU, setting the memory for the GPU to the minimum possible value (16 MB). Likewise, in order to reduce the impact of the operating system on the performance, the booting process of the RPi was configured to prevent needless processes and services from being started. We also disconnected all peripherals during the characterization.

B. SOFTWARE FRAMEWORKS

Various open-source software tools have been developed for DL prototyping and deployment. Among them, we chose four of the most widely-used frameworks for vision compatible with the RPi. They rely on different techniques and libraries for inference optimization, as described next.

Caffe [19], developed by the Berkeley Vision and Learning Center (BVLC), offers an extensive number of state-of-the-art network models, most of them tailored for computer vision applications. This framework requires Basic Linear Algebra Subprograms (BLAS) as the back-end for matrix and vector computations. We compiled Caffe with OpenBLAS [20], an optimized library for CPU speedup enabling to leverage the four ARM cores of the RPi. Its core code is written in C++. Python/Numpy and Matlab bindings are also available for Caffe.

TensorFlow [21] (TF) expresses computations as stateful dataflow graphs that manage tensors. This enables various optimizations, e.g. the elimination of redundant copies of the same operation in the graph or the implementation of careful operation scheduling that improves data transfer performance and memory usage. TensorFlow also makes use of BLAS optimized libraries for performing matrix multiplication on the CPU. We compiled TensorFlow version 1.3.0 for RPi [22]. The supported front-end languages are C++, Java, Go and Python. Additionally, TensorFlow provides high- and mid-level APIs based on Python. In particular, we used the TF-Slim image classification high-level API [23], that provides some pre-trained DNNs.

OpenCV [24] (Open Source Computer Vision Library) includes a DNN module for inference on pre-trained models from Caffe, TensorFlow or Torch frameworks. We built OpenCV version 3.3.1 [25], setting the appropriate compilation options to enable both ARM Neon and VFPv3 optimizations. They are designed to make the most of the Quad Core ARM processor of the RPi.

Caffe2 [26], released in April 2017 by Facebook AI Research group (FAIR), focuses on boosting speed, scalability and portability for mobile and embedded realizations. Caffe2 also uses computation graphs for network definition. The building process of Caffe2 on the RPi under Raspbian OS admits ARM Neon optimizations as well. In addition to pre-trained network models [27], Caffe2 includes a built-in tool to convert models from Caffe to Caffe2 format [28].

C. DNN MODELS

Convolutional Neural Networks (CNNs) are DNNs where convolution plays a primary computational role. They have become the core processing architecture underpinning most of the latest advances in visual inference. We have benchmarked six well-known CNNs performing 1000-category image recognition. Some of them integrate optimizations making them suitable for embedded applications.

Network in Network [29] (NiN) is a model made up of stacked so-called *MLPconv* layers, namely micro neural networks with a multilayer perceptron equivalent to Convolutional Layers (CONV) including point-wise kernels – i.e. 1×1 filters operating across tensors. Moreover, the last MLPconv layer generates one feature map for each class being predicted. This permits to replace Fully-Connected (FC) layers for a *global average pooling* and a subsequent SoftMax layer. By exploiting both strategies, this model achieves a considerable reduction of parameters compared to preceding models like AlexNet [30].

GoogLeNet [31] was the winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. Its most distinctive characteristic is the introduction of the so-called *Inception* module, composed of parallel convolutions restricted to filter sizes of 1×1, 3×3 and 5×5. The particular realization of the Inception architecture that won the challenge – called GoogLeNet or Inception-v1 – comprises 3 convolutional layers, followed by 9 inceptions modules (each containing 6 CONV layers), and finally one FC layer with 1024 units. Subsequent versions of GoogLeNet have been developed, the most recent one being Inception-v4 [32].

ResNet [33], or Residual Network, won the ILSVRC competition in 2015 as the first DNN to exceed human-level accuracy. It features *shortcut connections* performing identity mappings, so that one or more weight layers can be skipped. It also implements a *bottleneck* design, with 3 stacked layers made up of 1×1, 3×3, and 1×1 convolutions. The 1×1 layers reduce and increase the data dimensions respectively, alleviating the computational load of the intermediate 3×3 layer. ResNet-50 consists of 50 layers: one CONV layer, followed by 16 shortcut layers (containing 3 CONV layers each) and one FC layer. Other versions of ResNet have different depths defined by the number of 3-layer blocks. The winner of the ILSVRC 2015 was ResNet-152.

SqueezeNet [34] is tailored for the reduction of network parameters while maintaining the accuracy of AlexNet. For this purpose, it uses a great deal of 1×1 filters to massively bring down the number of weights. The building block of the network is the so-called *Fire* module, which contains two stacked layers: a squeeze convolution layer exclusively composed of 1×1 filters and an expand layer made up of 1×1 and 3×3 convolution filters. Overall, the architecture of this model includes one CONV layer, 8 Fire modules and a final CONV layer, dismissing any FC layer.

MobileNet [35] aims to be an efficient model for mobile and embedded vision applications. It is based on factorizing a standard convolution into two layers, namely depth-wise and

TABLE 1. Main parameters defining the architectures of the benchmarked CNN networks for 1000-category image recognition.

	Network in Network	GoogLeNet Inception-v1	ResNet 50	SqueezeNet v1.1	MobileNet v1 1.0-224	SimpleNet v1
INPUT Size	224x224x3	224x224x3	224x224x3	227x227x3	224x224x3	227x227x3
CONV Layers						
# stacked CONV layers	12	21	49	18	27	13
# CONV layers	12	57	49	26	27	13
# channels/filter (input ch.)	3-1024	3-832	3-2048	3-512	3-1024	3-2048
# filters (output ch.)	96-1024	3-1024	64-2048	64-1000	32-1024	64-2048
Kernel sizes	1,3,5,11	1,3,5,7	1,3,7	1,3	1,3	1,3,7
Stride	1,4	1,2	1,2	1,2	1,2	1,2,3
FC Layers						
# FC layers	0	1	1	0	1	1
# channels	0	1024	2048	0	1024	256
# filters	0	1000	1000	0	1000	1000
Kernel sizes	0	1	1	0	1	1
PARAMETERS	∼7.6M	∼7M	∼25.6M	∼1.3M	∼4.2M	∼6.4M

TABLE 2. Download sources of the benchmarked pre-trained CNN models. Networks sharing the reference with Caffe correspond to models converted from the corresponding Caffe model as they were not available for that framework.

	Network in Network	GoogLeNet	ResNet 50	SqueezeNet v1.1	MobileNet v1 1.0-224	SimpleNet v1
Caffe	[41]	[42]	[43]	[44]	[45]	[46]
TensorFlow	[41]	[47]	[48]	[44]	[49]	[46]
OpenCV	[41]	[42]	[43]	[44]	[45]	[46]
Caffe2	[41]	[50]	[43]	[51]	[45]	[46]

point-wise convolutions (1×1), drastically reducing the computational model parameters. It introduces a width multiplier $\alpha \in (0, 1]$ to reduce the number of input/output channels at each layer as well as a resolution multiplier $\rho \in (0, 1]$ to thin the input image of the network, whose typical resolutions are 224, 192, 168 and 128. Both multipliers have the effect of reducing the computational cost by α^2 and ρ^2, respectively. The baseline MobileNet model sets these factors to 1. Counting depth-wise and point-wise convolutions as separate layers, and using one FC layer, MobileNet comprises a total of 28 layers.

SimpleNet [36] is a new proposal of a simple architecture that, with fewer parameters, outperforms deeper architectures such as ResNet, VGGNet [37] or GoogLeNet in some popular datasets. Its creators employ different design principles to attain a simple model with high accuracy, e.g. progressive reduction of spatial resolution through the network (with progressive increase of the number of feature maps) but avoiding rapid down-sampling and 1×1 kernels in early layers. Larger convolution kernels are replaced with a cascade of smaller ones. The network comprises 13 CONV layers and 1 FC layer. Although this model was initially adjusted for CIFAR10, CIFAR100, SVHN and MNIST datasets, it can also be trained using ImageNet [38] – i.e. for 1000-category recognition.

The architectures of these six network models are summarized in Table 1. Note that the entries '# stacked CONV layers' and '# CONV layers' for GoogLeNet and SqueezeNet do not match. This is simply due to the fact that the Inception and Fire modules of these models are composed of several parallel sub-layers. The number of parameters were obtained using Netscope [39].

We have characterized ready-to-use implementations of these models provided by each framework – see references in Table 2. In the absence of a specific model on a particular framework, we translated it from Caffe since 1) this is the analyzed DL software that offers the largest set of pre-trained models; and 2) there are ready-made tools for conversion from Caffe to Caffe2 [28] and TensorFlow [40]. In the case of OpenCV, the inference was directly carried out on the models from Caffe.

IV. PERFORMANCE METRICS

For performance evaluation and comparison, we used Python as the common coding language for all of the frameworks. The parameters defined next were measured for each combination of CNN model on DL framework within the aforementioned set.

A. ACCURACY

We have calculated the *Top-5* correct category classification rate over the 50k validation images of the ImageNet ILSVRC 2012 dataset. This dataset is made up of variable-resolution color images encompassing 1000 object categories. Since the DNN models require fixed-size inputs (see Table 1),

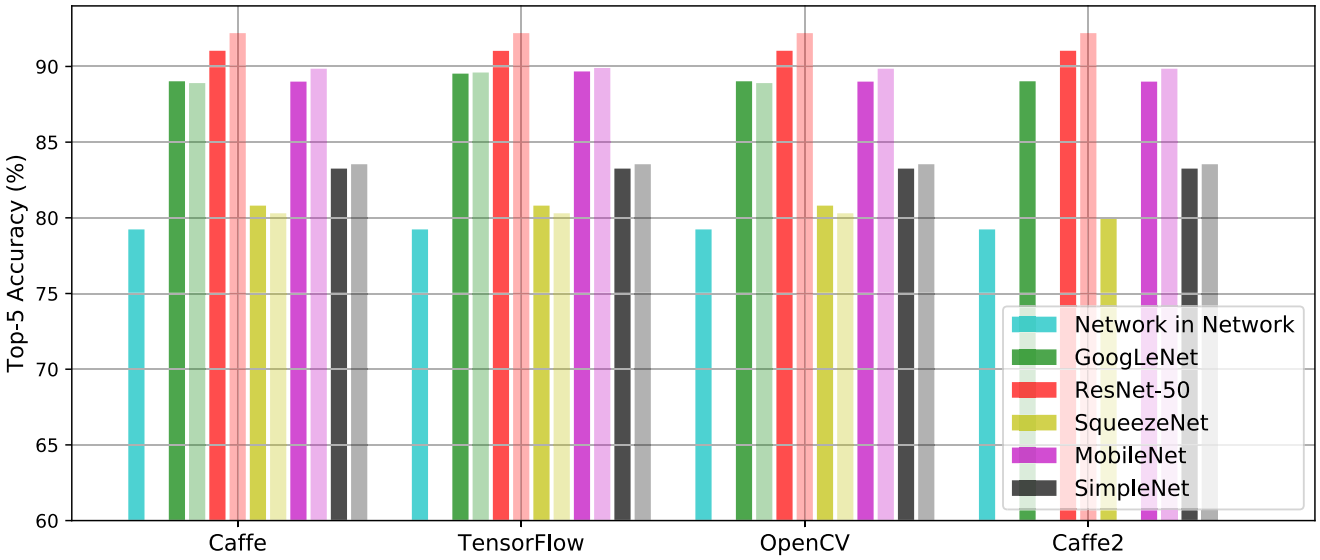


FIGURE 1. Top-5 accuracy of the DNN models. Darker bars show measured values, while companion lighter ones stand for reported accuracies – if available, this is why there are missing lighter bars. The small deviations may be due to the application of slightly different image pre-processing.

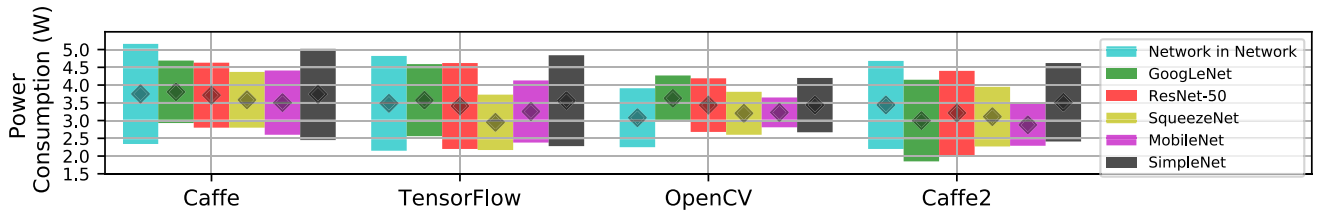


FIGURE 2. Limits and mean values of power consumption during inference.

we resized the images to 256x256x3 and then cropped out the central patch with the resolution specifically fitting the DNN input size. We also subtracted the mean values over the dataset from each pixel. Mean subtraction and scale values were extracted from the model repositories if provided, taking the mean value over the ImageNet dataset otherwise. This pre-processing has been carried out with OpenCV functions and single-precision floating-point format. It is worth noting though that accuracy was measured without any data augmentation, e.g. multiple crops, ensemble of multiple models etc. The results are shown in Fig. 1. They mostly agree with the figures reported in the corresponding original research papers and repositories. The small deviations could come from slight differences in the applied pre-processing.

B. POWER CONSUMPTION

We have measured the instantaneous power consumption when performing inference over 100 images of the resized ImageNet set (256x256x3). However, in some cases the SoC of the RPi heats up significantly, forcing the CPU clock to go down. In these situations, inference was performed over fewer images, always ensuring that the CPU frequency was kept near its maximum, i.e. higher than 1100 MHz. These exceptions were GoogLeNet on Caffe – stopping inference at image #64 –, ResNet on Caffe, TF and OpenCV – stopping at image #24, #69 and #75, respectively –, and SimpleNet on Caffe – stopping at image #55. The results are depicted in Fig. 2. The wide variation range of the measurements seems to be caused by the different characteristics of the layers within the models in terms of computational load and memory usage.

C. THROUGHPUT

We have measured the throughput concurrently with power consumption. The batch size was set to 1 since we are aiming at real-time applications where latency should be minimal. The *frames per second* rate takes into account both the time required to get the input image ready for the particular model – i.e. the time required for scaling, mean subtraction etc. – plus the inference time itself. Fig. 3 depicts the average throughput achieved by each combination under study.

According to these results, ResNet-50 is the most accurate model within the analyzed set. However, its complexity leads to a high number of operations per inference and, consequently, to the lowest throughput. On the other hand, SqueezeNet has the fewest parameters by far, being computationally lighter than the other models. It achieves the highest throughput at the cost of lower – but not the lowest – accuracy.

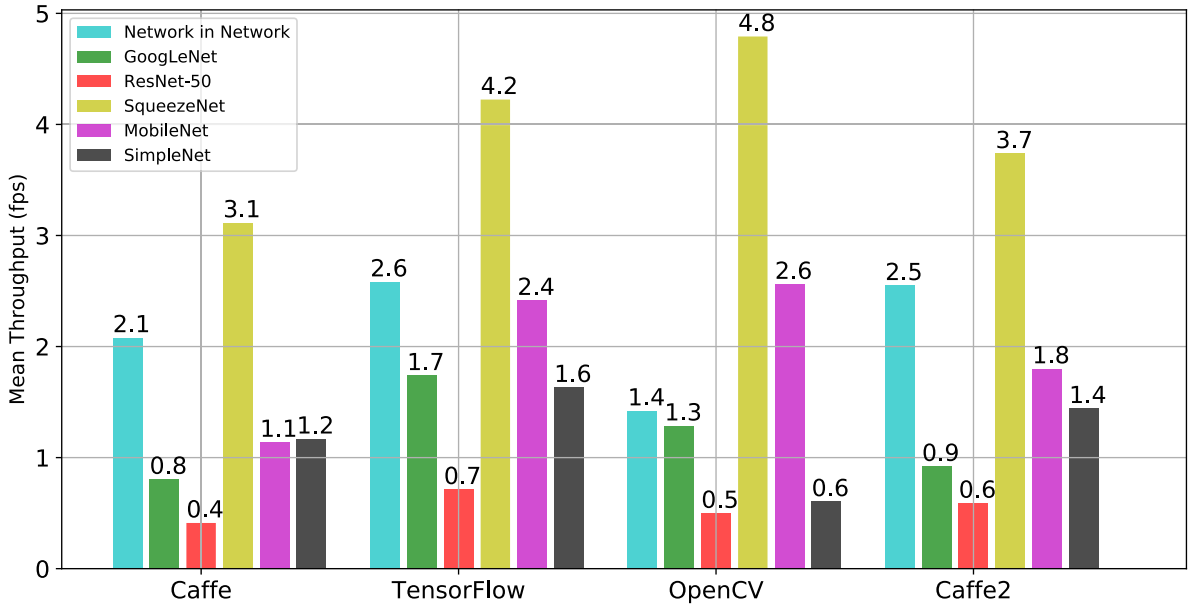


FIGURE 3. Average rate of frames per second for each combination of DNN model and DL framework.

Concerning power consumption, OpenCV performs steadier, exhibiting narrower intervals of power consumption during inference. MobileNet is one of the less energy-demanding models, reaching the minimum average consumption when running on Caffe2 – interestingly, both model and framework have been designed for embedded applications.

V. OPTIMUM MODEL/FRAMEWORK SELECTION

A FoM that merges the performance metrics just discussed in a meaningful way can be defined as:

$$\text{FoM} := \text{Accuracy} \cdot \frac{\text{Throughput}}{\text{Power}} \tag{1}$$

It can be expressed as 'number of correctly inferred images per joule', providing insight about the joint computational and energy efficiency of models and frameworks on the considered hardware platform.

The results after introducing the values from Figs. 1-3 – mean value in the case of power consumption – in this FoM are depicted in Fig. 4. It shows that SqueezeNet performs the best according to (1), in particular when running on OpenCV. The superiority of this combination arises from achieving a notable frame rate in a power-efficient way while trading off some accuracy in comparison with more precise networks like ResNet-50 or GoogLeNet. The question is: *would SqueezeNet-OpenCV still be the best choice for a particular application where accuracy is more important, in relative terms, than throughput and/or power consumption?* In order to answer this question, we propose a re-definition of the FoM in such a way that the relative weight of each performance parameter on high-level specifications can be taken into account.

Let $[\alpha_\text{A}, \alpha_\text{T}, \alpha_\text{P}]$ be the relative importance of the metrics [Accuracy, Throughput, Power] respectively, satisfying:

$$\sum_i \alpha_i = 1, \alpha_i \in [0, 1] \tag{2}$$

where i corresponds to A, T, P. It follows from (2) that, for instance, a vector $[\alpha_\text{A}, \alpha_\text{T}, \alpha_\text{P}] = [0.35, 0.05, 0.60]$ defines an application scenario where power consumption presents a relative preeminence of 60% on the targeted performance, followed by accuracy with a weight of 35% and throughput with very little importance, only 5%.

A new weighted FoM can thus be defined as:

$$\text{FoM}(\alpha_\text{A}, \alpha_\text{T}, \alpha_\text{P}) := \text{Accuracy}^{3\alpha_\text{A}} \cdot \frac{\text{Throughput}^{3\alpha_\text{T}}}{\text{Power}^{3\alpha_\text{P}}} \tag{3}$$

Hence, (1) is just a particular case of the generalized FoM in (3): when $\alpha_i = 1/3 \ \forall i$, we are assuming that the three performance parameters have the same impact on application-level requirements. Any other distribution of α_i assigns a specific weight to their corresponding factor in (3) by setting its exponent to a value greater or less than unity. Extreme cases occur when any α_i is set to 1, boosting the influence of its associated performance parameter on the FoM at the cost of completely dismissing the other ones.

Each possible application scenario – i.e. each possible combination of α_i setting a weighted balance of accuracy, throughput and power consumption – can now be assessed in terms of the FoM proposed in (3). The objective is to find the optimum model/framework choice per application scenario, using the baseline experimental measurements presented in Section IV. For each combination of α_i, we normalize the FoM with respect to its maximum value obtained for the optimum model/framework pair. This permits to quickly identify

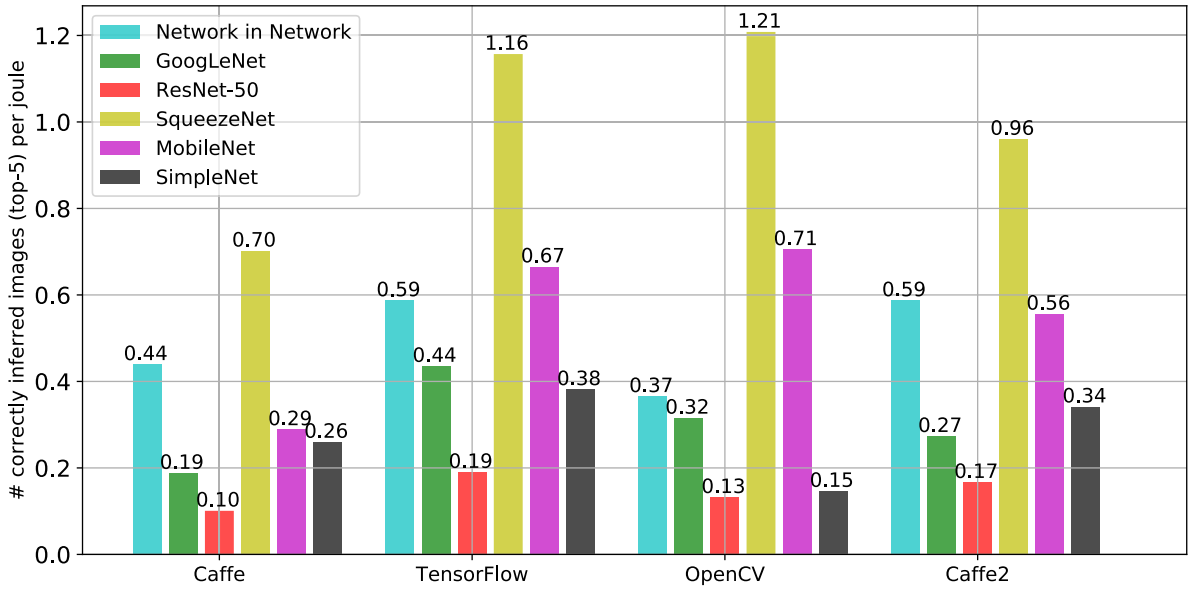


FIGURE 4. Figure of Merit defined in (1). It is expressed as 'number of correctly inferred images per joule'.

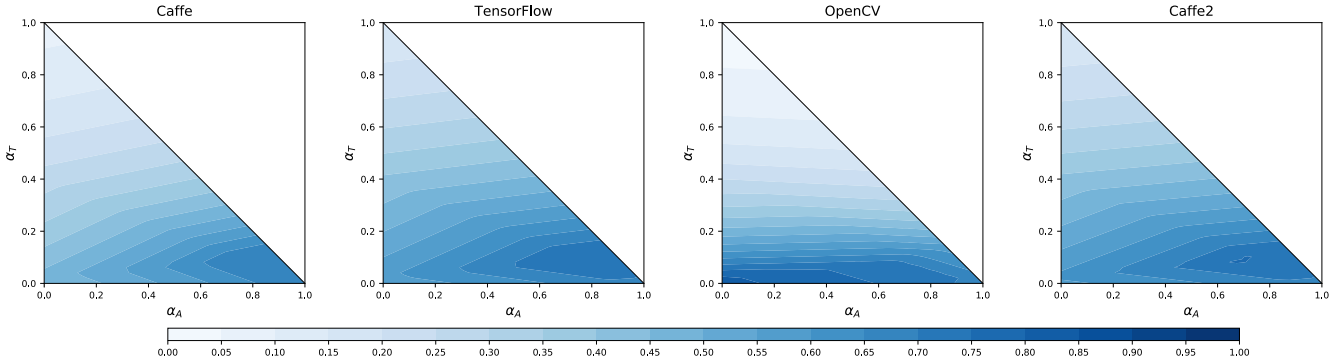


FIGURE 5. Normalized weighted FoM for Network in Network.

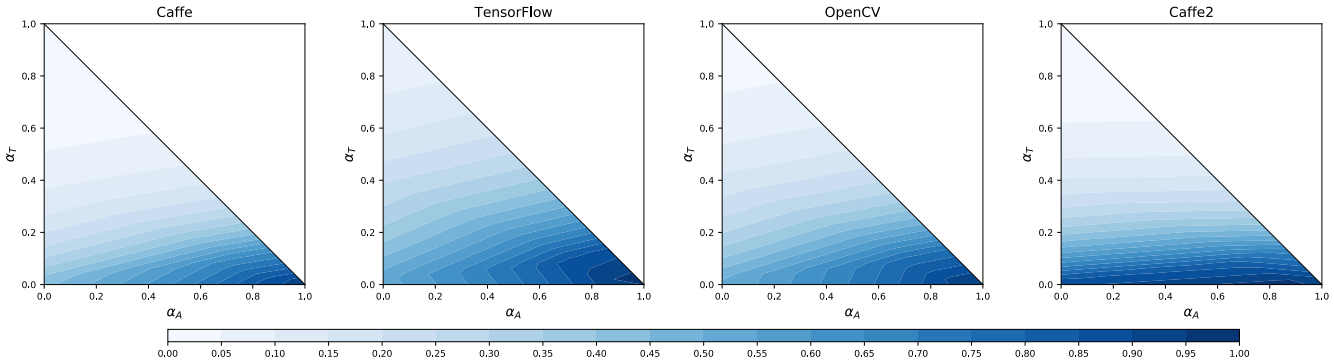


FIGURE 6. Normalized weighted FoM for GoogLeNet.

the best selection with a FoM equal to 1. Likewise, it allows to easily evaluate how far each pair is from the optimum point.

All in all, Figs. 5-10 depict the normalized values of the proposed weighted FoM for each DNN model per software framework. Note that we only consider α_A and α_T since,

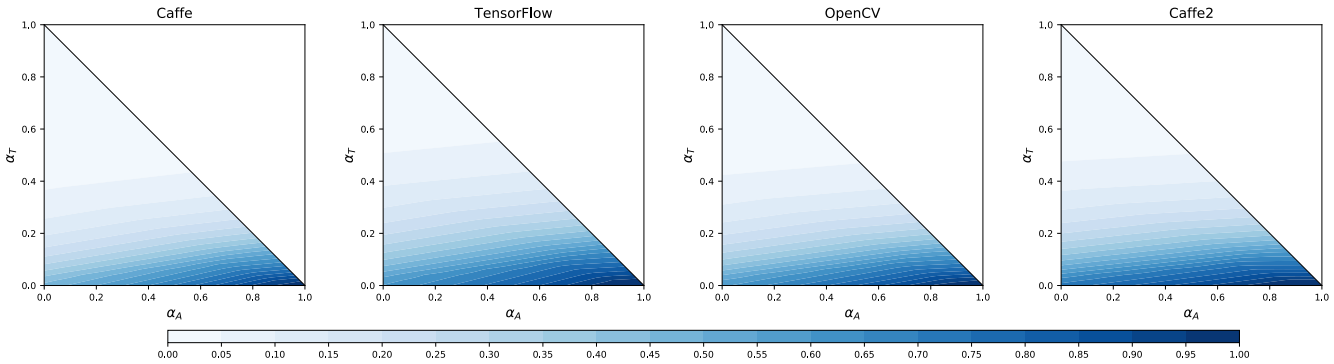


FIGURE 7. Normalized weighted FoM for ResNet-50.

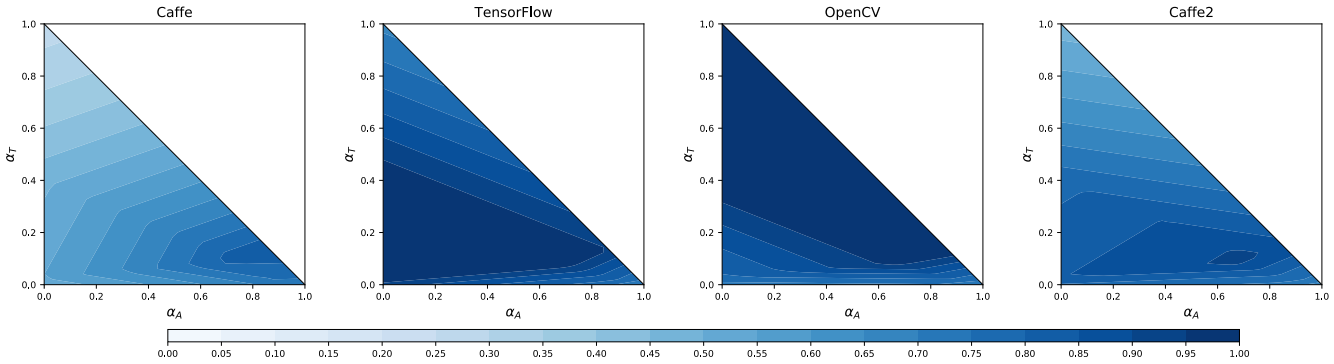


FIGURE 8. Normalized weighted FoM for SqueezeNet.

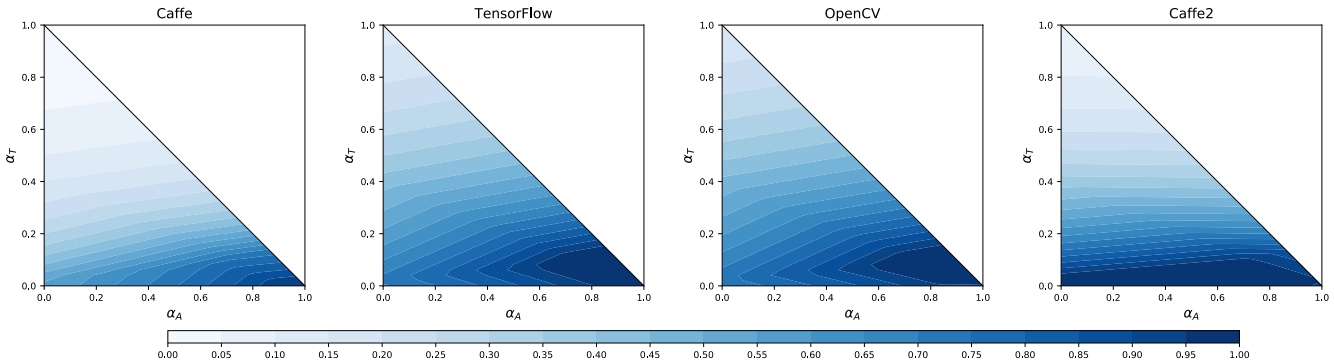


FIGURE 9. Normalized weighted FoM for MobileNet.

according to (2), once their values are established, α_P is automatically determined from $\alpha_P = 1 - \alpha_A - \alpha_T$. In these plots, the maximum is represented in blue. Points getting far from this maximum undergo color degradation towards lighter tones. Bar plots similar to that of Fig. 4 can be extracted for particular application scenarios, that is, for specific points within the exploration domain of α_i. For instance, the scenario defined by the triplet previously mentioned, $[\alpha_A, \alpha_T, \alpha_P] = [0.35, 0.05, 0.6]$, renders the plot in Fig. 11. Interestingly, MobileNet on Caffe2 is the best choice in this case.

A rapid visual inspection of the FoM plots in Figs. 5-10 permits to conclude that SqueezeNet is still the best network for most application scenarios. This is confirmed by Fig. 12 where we represent the optimum selection, i.e. the network/framework pair reaching the maximum FoM,

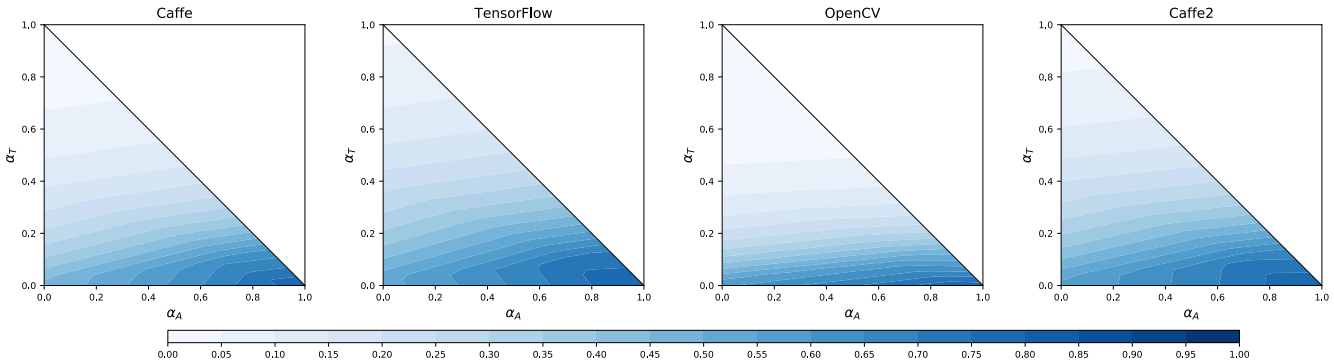


FIGURE 10. Normalized weighted FoM for SimpleNet.

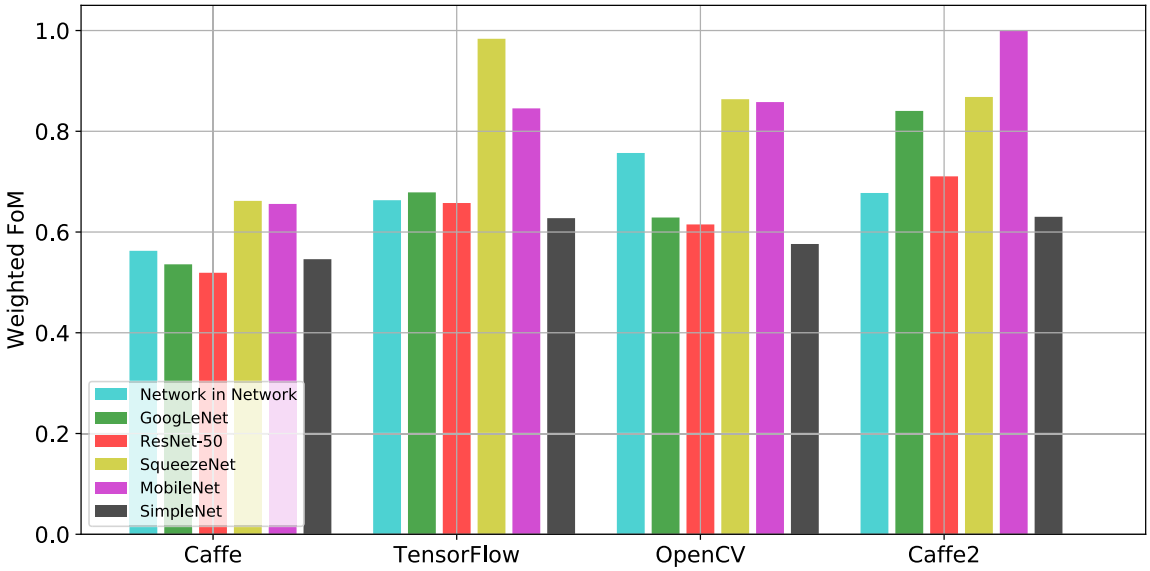


FIGURE 11. Normalized values of the FoM defined in (3) for a scenario where power consumption presents a relative preeminence of 60% on the targeted performance whereas the importance of accuracy and throughput is 35% and 5% respectively. In this particular case, MobileNet on Caffe2 is the best choice.

TABLE 3. Area covered by optimum model/framework combinations in Fig. 12.

Model - Framework	Area (%)
ResNet-50 - Caffe/OpenCV/Caffe2	0.00
ResNet-50 - TF	0.04
ResNet-50 - Caffe2	0.25
MobileNet - OpenCV	1.27
MobileNet - TF	2.58
MobileNet - Caffe2	9.61
SqueezeNet - TF	28.17
SqueezeNet - OpenCV	58.09

for the whole domain of α_i. The corresponding percentage of area covered by each optimum pair in Fig. 12 is reported in Table 3. The combinations of ResNet-50 with Caffe/OpenCV/Caffe2 – note that these frameworks share the same realization of ResNet-50, slightly more accurate than that of TensorFlow; see Table 2 – constitute a singular case since they jointly reach a maximum when $\alpha_A \rightarrow 1$. This is why their covered area is negligible in practical terms.

TABLE 4. Area covered by worst model/framework combinations in Fig. 13.

Model - Framework	Area (%)
SimpleNet - Caffe	0.04
Network in Network - OpenCV	0.43
GoogLeNet - Caffe	1.76
SimpleNet - OpenCV	5.77
Network in Network - Caffe	5.95
ResNet-50 - Caffe	86.05

Finally, Fig. 13 shows which combinations perform the worst across the domain of α_i. The corresponding covered area is presented in Table 4. These pairs produce the lowest values of the weighted FoM. These lowest values are depicted

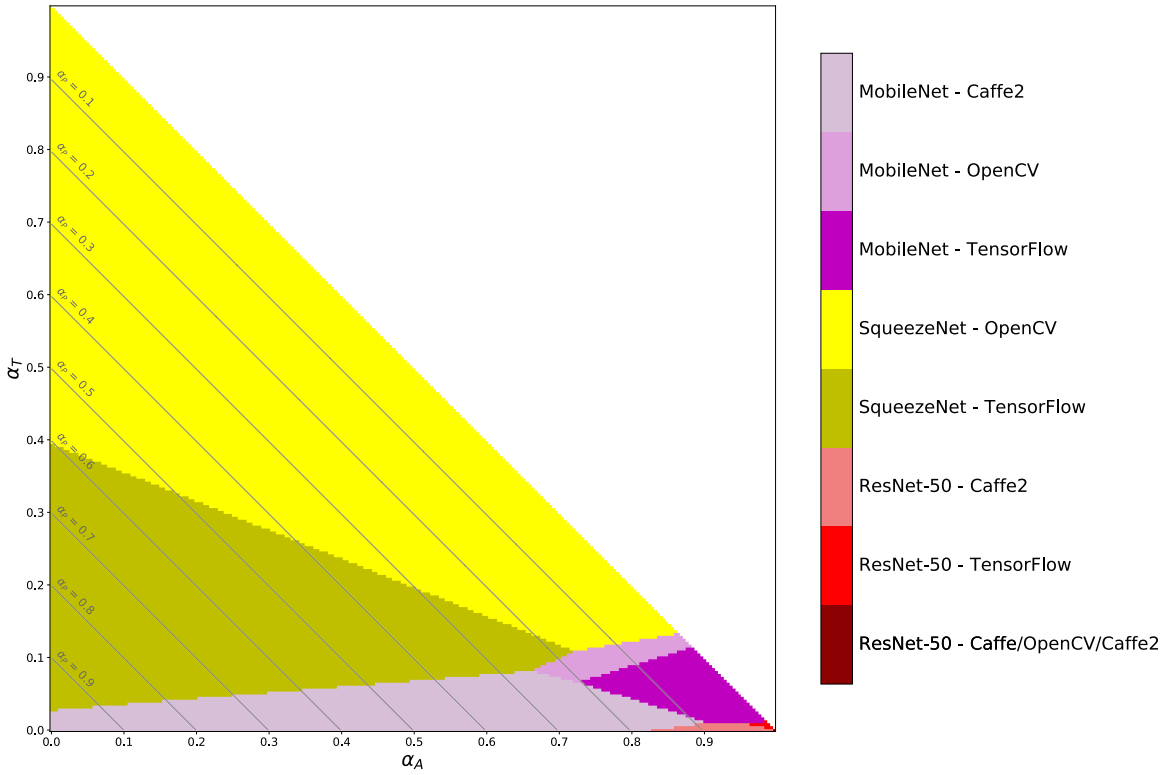


FIGURE 12. Optimal network/framework selection according to the weighted FoM defined in (3). Only 8 out of the 24 analyzed combinations perform the best in at least one point of the exploration domain.

in Fig. 14, highlighting the fact that some network/framework combinations can be really far from optimum performance.

VI. DISCUSSION

The most significant outcome from these results is that SqueezeNet is the most suitable network for the majority of the explored performance weightings. For application scenarios where throughput is critical, SqueezeNet must be implemented on OpenCV. When energy efficiency gains relevance, TensorFlow makes the most of this model. For relaxed frame rate requirements, MobileNet performs the best on combining high accuracy and reduced power consumption. When power loses importance, keeping relaxed constraints on throughput, MobileNet on TensorFlow is the best choice. Applications where accuracy is extremely relevant should obviously select the most accurate DNN model: ResNet-50 in our case study. Ruling out extreme cases, we have narrowed down the original list of 24 network/framework pairs to 5 combinations to be selected. Furthermore, in practical terms only 3 combinations – SqueezeNet on OpenCV and TensorFlow; MobileNet on Caffe2 – cover more than 95% of cases.

Our results also point out that Network in Network, GoogLeNet and SimpleNet are never the best choice, no matter the software framework considered. On the contrary, they are the worst selections in a number of scenarios, as shown in Fig. 13. Network in Network performs the worst because of its lowest accuracy and high power consumption when running on Caffe, and also its moderate frame rate on OpenCV. Despite the high accuracy of GoogLeNet, it features a poor performance in the other metrics, being the most energy-demanding model in combination with Caffe. Concerning ResNet-50 on Caffe, it covers most of the area in Fig. 13 – more than 85%. The highest accuracy of this model does not make up for its high power consumption and poorest performance in terms of throughput. In particular, its throughput is notably lower than the best one – 0.41fps vs. 4.79fps for SqueezeNet-OpenCV. This is why the normalized FoM in Fig. 14 degrades for increasing values of α_T. Finally, SimpleNet, whose simplicity is reflected in a poor accuracy, also exhibits a deficient performance on the other metrics.

We must emphasize that this comparative study is strictly quantitative in the sense that our discussion is exclusively focused on measurable performance. Thus, the FoM defined in (3) does not include any qualitative parameters encoding the suitability of a particular model, framework or hardware platform according to, for instance, a long-term technological strategy in a company or a product development roadmap. However, taking into account the current lack of standardization in the field of DL-based vision, and its rapid evolution pace, qualitative considerations should not have too much influence yet on the final decision to be made. The conclusions drawn in this manuscript should not therefore change significantly when such considerations were included in the analysis.

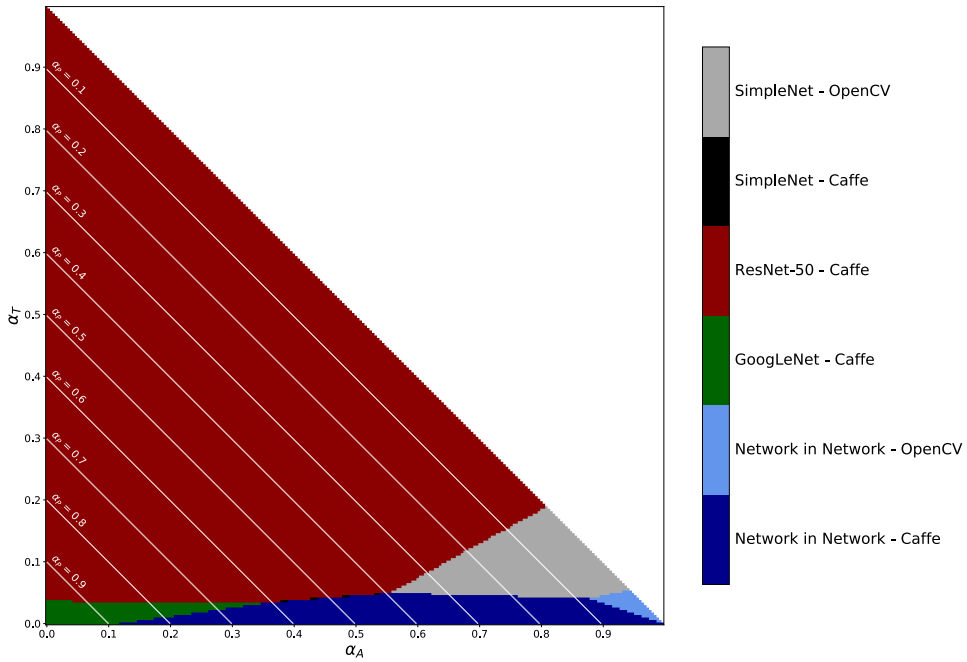


FIGURE 13. Worst network/framework selection according to the weighted FoM. The normalized values of the FoM for each point in this plot are represented in Fig. 14.

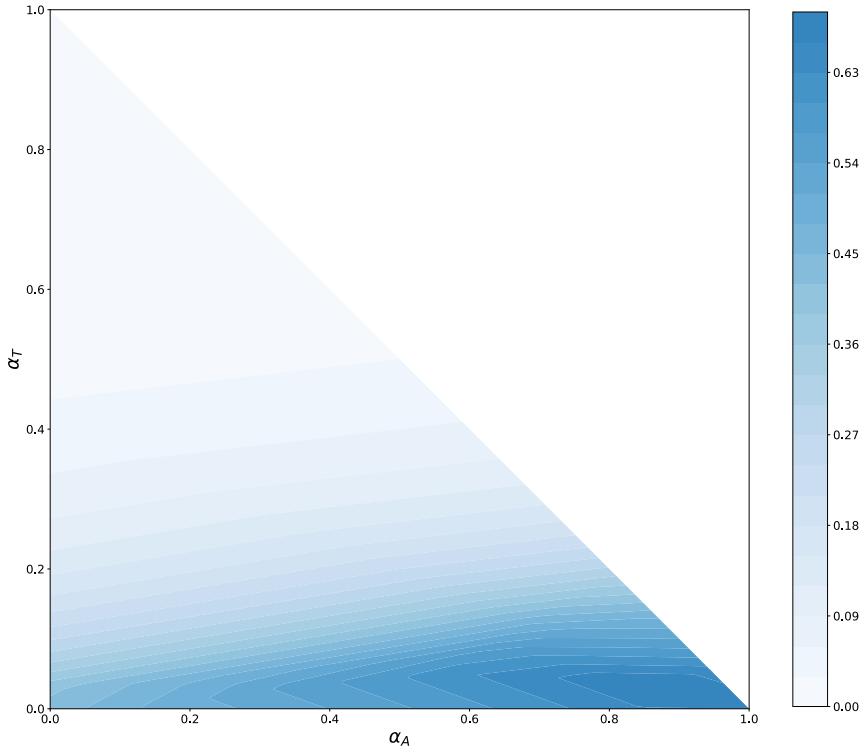


FIGURE 14. Lowest values of the normalized FoM. They correspond to the network/framework pairs represented in Fig. 13.

VII. CONCLUSIONS

We have proposed a methodology to support system designers when making decisions in the broad ecosystem of DL components for embedded vision. Firstly, we have demonstrated through measured metrics that an inexpensive embedded computer like Raspberry Pi 3 Model B is capable of implementing real-time image recognition among 1000 categories. Secondly, we have introduced a FoM that

facilitates the evaluation of DL components according to the weighted relevance of key operation parameters on application-level specifications. It has proved to be effective on filtering out most of the studied alternatives. Specifically, we have demonstrated that SqueezeNet currently constitutes the best choice in most application scenarios. In the future, we will be extending our analysis to incorporate other essential performance considerations like memory or temperature restrictions.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[2] M. Verhelst and B. Moons, "Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to IoT and edge devices," *IEEE Solid-State Circuits Mag.*, vol. 9, no. 4, pp. 55–65, Nov. 2017.

[3] V. Sze, "Designing hardware for machine learning," *IEEE Solid-State Circuits Mag.*, vol. 9, no. 4, pp. 46–54, 2017.

[4] A. Suleiman, Y.-H. Chen, J. Emer, and V. Sze, "Towards closing the energy gap between HOG and CNN features for embedded vision," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2017, pp. 444–447.

[5] Qualcomm. (Aug. 2016). *Emerging Vision Technologies: Enabling a New Era of Intelligent Devices*. [Online]. Available: https://www.qualcomm.com/documents/emerging-vision-technologies-enabling-new-era-intelligent-devices

[6] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[7] (2016). *Raspberry Pi 3 Model B*. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[8] V. Kovalev, A. Kalinovsky, and S. Kovalev, "Deep learning with theano, torch, caffe, tensorflow, and deeplearning4J: Which one is the best in speed and accuracy?" in *Proc. Int. Conf. Pattern Recognit. Inf. Process.*, 2016, pp. 99–103.

[9] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah. (2015). "Comparative study of caffe, neon, theano, and torch for deep learning." [Online]. Available: https://arxiv.org/abs/1511.06435

[10] S. Shi, Q. Wang, P. Xu, and X. Chu. (2016). "Benchmarking state-of-the-art deep learning software tools." [Online]. Available: https://arxiv.org/abs/1608.07249

[11] A. Canziani, A. Paszke, and E. Culurciello. (2017). "An analysis of deep neural network models for practical applications." [Online]. Available: https://arxiv.org/abs/1605.07678

[12] Y. Zhang, H. Peng, and P. Hu, "CS341 final report: Towards real-time detection and camera triggering," Standford, CA, USA, Tech. Rep. CS 341, 2017.

[13] J. Ahmad, I. Mehmood, S. Rho, N. Chilamkurti, and S. W. Baik, "Embedded deep vision in smart cameras for multi-view objects representation and retrieval," *Comput. Elect. Eng.*, vol. 61, pp. 297–311, Jul. 2017.

[14] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Syst. Appl.*, vol. 72, pp. 327–334, Apr. 2017.

[15] D. C. De Oliveira and M. A. Wehrmeister, "Towards real-time people recognition on aerial imagery using convolutional neural networks," in *Proc. 19th IEEE Int. Symp. Real-Time Distrib. Comput. (ISORC)*, 2016, pp. 27–34.

[16] O. Dürr, Y. Pauchard, D. Browarnik, R. Axthelm, and M. Loeser, "Deep learning on a raspberry Pi for real time face recognition," in *Proc. Eurograph.*, 2015, pp. 1–5.

[17] D. Pena, A. Forembski, X. Xu, and D. Moloney, "Benchmarking of CNNs for low-cost, low-power robotics applications," in *Proc. RSS Workshop, New Frontier Deep Learn. Robot.*, 2017, pp. 1–5.

[18] *Raspbian*. Accessed: Sep. 14, 2018. [Online]. Available: https://www.raspberrypi.org/downloads/raspbian/

[19] Y. Jia *et al.* (2014). "Caffe: Convolutional architecture for fast feature embedding." [Online]. Available: https://arxiv.org/abs/1408.5093

[20] *OpenBLAS, Optimized BLAS Library Based on GotoBLAS2 1.13 BSD Version*. Accessed: Sep. 14, 2018. [Online]. Available: https://github.com/xianyi/OpenBLAS

[21] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[22] (2017). *A Docker Image for Tensorflow*. Accessed: Sep. 14, 2018. [Online]. Available: https://github.com/DeftWork/rpi-tensorflow

[23] *TensorFlow-Slim Image Classification Model Library*. Accessed: Sep. 14, 2018. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/slim

[24] *OpenCV*. Accessed: Sep. 14, 2018. [Online]. Available: https://opencv.org/

[25] *Open Source Computer Vision Library*. Accessed: Sep. 14, 2018. [Online]. Available: https://github.com/opencv/opencv

[26] Caffe2. *Caffe2 Framework*. Accesed: Sep. 14, 2018. [Online]. Available: https://caffe2.ai/

[27] Caffe2. *Model Zoo*. Accessed: Sep. 14, 2018. [Online]. Available: https://github.com/caffe2/caffe2/wiki/Model-Zoo

[28] *Caffe to Caffe2*. Accessed: Sep. 14, 2018. [Online]. Available: https://caffe2.ai/docs/caffe-migration.html#caffe-to-caffe2

[29] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400

[30] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[31] C. Szegedy *et al.* (2014). "Going deeper with convolutions," [Online]. Available: https://arxiv.org/abs/1409.4842

[32] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. (2016). "Inception-v4, inception-resnet and the impact of residual connections on learning." [Online]. Available: https://arxiv.org/abs/1602.07261

[33] K. He, X. Zhang, S. Ren, and J. Sun. (2015). "Deep residual learning for image recognition." [Online]. Available: https://arxiv.org/abs/1512.03385

[34] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer. (2016). "SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and <0.5 MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360

[35] A. G. Howard *et al.* (2017). "MobileNets: Efficient convolutional neural networks for mobile vision applications." [Online]. Available: https://arxiv.org/abs/1704.04861

[36] S. Hasanpour, M. Rouhani, M. Fayyaz, and M. Sabokrou. (2016). "Lets keep it simple, using simple architectures to outperform deeper and more complex architectures." [Online]. Available: https://arxiv.org/abs/1608.06037

[37] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[38] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[39] *Netscope CNN Analyzer*. Accessed: Sep. 14, 2018. [Online]. Available: http://dgschwend.github.io/netscope/quickstart.html

[40] (2017). *Caffe to TensorFlow*. [Online]. Available: https://github.com/ethereon/caffe-tensorflow

[41] *Model Zoo. Network in Network Model*. Accessed: Sep. 14, 2018. [Online]. Available: https://github.com/BVLC/caffe/wiki/Model-Zoo#network-in-network-model

[42] (2017). *BAIR/BVLC GoogLeNet Model*. [Online]. Available: https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet

[43] (2016). *Deep Residual Learning for Image Recognition*. [Online]. Available: https://github.com/KaimingHe/deep-residual-networks

[44] (2016). *SqueezeNet_v1.1*. [Online]. Available: https://github.com/DeepScale/SqueezeNet/tree/master/SqueezeNet_v1.1

[45] (2018). *Caffe Implementation of Google's MobileNets (V1 and V2)*. [Online]. Available: https://github.com/shicai/MobileNet-Caffe

[46] (2018). *SimpleNet*. [Online]. Available: https://github.com/Coderx7/SimpleNet

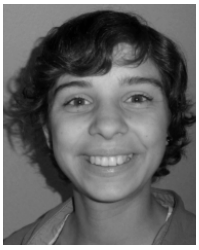
[47] (2016). *Inception V1*. [Online]. Available: http://download.tensorflow.org/models/inception_v1_2016_08_28.tar.gz

[48] (2016). *ResNet V1 50*. [Online]. Available: http://download.tensorflow.org/models/resnet_v1_50_2016_08_28.tar.gz

[49] (2018). *MobileNet_v1_1.0_224*. [Online]. Available: http://download.tensorflow.org/models/mobilenet_v1_2018_02_22/mobilenet_v1_1.0_224.tgz

[50] Caffe2 Models. *BVLC GoogLeNet*. [Online]. Available: https://github.com/caffe2/models/tree/master/bvlc_googlenet

[51] Caffe2 Models. *SqueezeNet*. Accessed: Sep. 14, 2018. [Online]. Available: https://github.com/caffe2/models/tree/master/squeezenet



DELIA VELASCO-MONTERO received the B.Eng. degree (Hons.) and the M.Sc. degree (Hons.) in telecommunication from the University of Seville, Seville, Spain, in 2014 and 2016, respectively. She then spent six months with the National Institute of Aerospace Technology, Madrid, Spain, granted by the Ministry of Defense, Spain. She joined the Institute of Microelectronics of Seville, CSIC, University of Seville, as a Ph.D. student, in 2017. Her main areas of interest are vision systems and embedded realizations of deep neural networks.

JORGE FERNÁNDEZ-BERNI received the B.Eng. degree (Hons.) in electronics and telecommunication, the M.Sc. degree (Hons.) in microelectronics, and the Ph.D. degree (Hons.) from the University of Seville, Seville, Spain, in 2004, 2008, and 2011, respectively. From 2005 to 2006, he was with Telecommunication Industry. He has been a Visiting Researcher with the Computer and Automation Research Institute, Budapest, Hungary, Ghent University, Ghent, Belgium, and the University of Notre Dame, Notre Dame, IN, USA. He is currently an Associate Professor with the Department of Electronics and Electromagnetism, University of Seville. He has authored or co-authored some 60 papers in refereed journals, conferences, and workshops. He is the first author of a book and two book chapters. He is also the first inventor of two licensed patents. His current research interests include smart image sensors, vision chips, and embedded vision systems. He received the Best Paper Award in image sensors and imaging systems from SPIE Electronic Imaging, San Francisco, CA, USA, in 2014, and the Third Prize of the Student Paper Award from the IEEE CNNA 2010: 12th International Workshop on Cellular Nanoscale Networks and their Applications, Berkeley, CA, USA.

RICARDO CARMONA-GALÁN (SM'16) received the degree in physics and the Ph.D. degree in microelectronics from the University of Seville, Spain, in 1993 and 2002, respectively. From 1996 to 1998, he was a Research Assistant with Prof. Chua Laboratory, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA. From 1999 to 2005, he was an Assistant Professor with the Department of Electronics, University of Seville. He held a post-doctoral position at the University of Notre Dame, Notre Dame, IN, USA, from 2006 to 2007, where he was involved in interfaces for CMOS compatible nanostructures for multispectral light sensing. He has collaborated with start-up companies in Anafocus, Seville, and Eutecus, Berkeley. He has designed several vision chips implementing different focal plane operators for early vision processing.

Since 2005, he has been a Tenured Scientist with the Institute of Microelectronics of Seville, University of Seville. His main research interests were focused on VLSI implementation of concurrent sensor/processor arrays for real time image processing and vision. He has co-authored over 120 papers in refereed journals and conferences, and several book chapters. His current research interests lie in the design of low-power smart image sensors and 3-D integrated circuits for autonomous vision systems. He was a recipient of the Best Paper Award for the *International Journal of Circuit Theory and Applications*. He was a co-recipient of the Award from ACET and a Certificate of Teaching Excellence from the University of Seville.

ÁNGEL RODRÍGUEZ-VÁZQUEZ (F'99) received the bachelor's degree and the Ph.D. degree in physics and electronics from the University of Seville in 1976 and 1982, respectively, with several national and international awards, including the IEEE Rogelio Segovia Torres Award in 1981. After research stays with the University of California at Berkeley, Berkeley, and Texas A&M University, he became a Full Professor of electronics with the University of Seville in 1995. He Co-Founded the Institute of Microelectronics of Seville, a joint undertaken of the Superior Council of Scientific Investigations, University of Seville, where he started a Research Lab on analog and mixed-signal circuits for sensors and communications.

He holds eight patents. AnaFocus was founded on the basis of his early patents. In 2001, he was the Main Promoter and the Co-Founder of the start-up company AnaFocus Ltd., where he served as the CEO, on leave from the University of Seville, until 2009, when the company reached maturity as a worldwide provider of smart CMOS imagers and vision systems-on-chip. While in Academia, he conducted research and development activities on mixed-signal microelectronics for massive sensory data, including vision chips and neuro-fuzzy controllers. He also pioneered the application of chaos to instrumentation and communications. His team designed the first, world-wide, integrated circuits with controllable chaotic behavior and the design and prototyping of the first world-wide chaos-based communication MoDem chips. His team also made significant contributions to the areas of structured analog and mixed-signal designs and the areas of data converter designs, including the elaboration of advanced teaching materials on this topic for different industrial courses and the production of two widely quoted books on the design of high-performance CMOS sigma-delta converters.

Many high-performance mixed-signal chips were successfully designed by him and his co-workers in the framework of different research and development programs and contracts. These include three generation of vision chips for high-speed applications, analog front-ends for XDSL MoDems, ADCs for wireless communications, ADCs for automotive sensors, chaotic signals generators, and complete MoDems for power-line communications. Many of these chips were the state of the art in their respective fields. Some of them entered in massive production. He was also active regarding industrial training. He produced teaching materials on data converters that were employed by several companies. His courses got the quality label of Europractice. His research work has received some 9,100 citations. He has an h-index of 48 and an i10-index of 177 according to Google Scholar. He has received a number of awards for his research: the IEEE Guillemin-Cauer Best Paper Award, two Wiley's IJCTA Best Paper Awards, two IEEE ECCTD Best Paper Awards, the IEEE-ISCAS Best Paper Award, the SPIE-IST Electronic Imaging Best Paper Award, the IEEE ISCAS Best Demo-Paper Award, and the IEEE ICECS Best Demo-Paper Award.

Dr. Rodríguez-Vázquez served as the Region 8 VP of the IEEE Circuits and Systems Society (2009–2012). He served on the committee of several international journals and conferences. He was the Chair of several international IEEE (NDES 1996, CNNA 1996, ECCTD 2007, ESSCIRC 2010, and ICECS 2013) and SPIE conferences. He was also the Chair of the IEEE CASS Fellow Evaluation Committee (2010, 2012, 2013, 2014, and 2015). He will be appointed as the General Chair for IEEE ISCAS 2020. He has served as an Editor, an Associate Editor, and a Guest Editor for IEEE and non-IEEE journals.