IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 1, 2018, accepted October 17, 2018, date of publication October 24, 2018, date of current version November 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2877890

Benchmark Analysis of Representative Deep Neural Network Architectures

SIMONE BIANCO[1], REMI CADENE[2], LUIGI CELONA[1], AND PAOLO NAPOLETANO[1]
[1]Department of Informatics, Systems and Communication, University of Milano-Bicocca, 20126 Milan, Italy
[2]LIP6, CNRS, Sorbonne Université, 75005 Paris, France

Corresponding author: Luigi Celona (luigi.celona@disco.unimib.it)

This work was supported in part by the FooDesArt, CUP: E48I16000350009 – Call "Smart Fashion and Desing" project, in part by the POR FESR 2014-2020; TEINVEIN, CUP: E96D17000110009 – Call "Accordi per la Ricerca e l'Innovazione" project, and in part by the POR FESR 2014-2020 project.

ABSTRACT This paper presents an in-depth analysis of the majority of the deep neural networks (DNNs) proposed in the state of the art for image recognition. For each DNN, multiple performance indices are observed, such as recognition accuracy, model complexity, computational complexity, memory usage, and inference time. The behavior of such performance indices and some combinations of them are analyzed and discussed. To measure the indices, we experiment the use of DNNs on two different computer architectures, a workstation equipped with a NVIDIA Titan X Pascal, and an embedded system based on a NVIDIA Jetson TX1 board. This experimentation allows a direct comparison between DNNs running on machines with very different computational capacities. This paper is useful for researchers to have a complete view of what solutions have been explored so far and in which research directions are worth exploring in the future, and for practitioners to select the DNN architecture(s) that better fit the resource constraints of practical deployments and applications. To complete this work, all the DNNs, as well as the software used for the analysis, are available online.

INDEX TERMS Deep neural networks, convolutional neural networks, image recognition.

I. INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable results in many computer vision tasks [1]. AlexNet [2], that is the first DNN presented in the literature in 2012, drastically increased the recognition accuracy (about 10% higher) with respect to traditional methods on the 1000-class ImageNet Large-Scale Visual Recognition Competition (ImageNet-1k) [3]. Since then, literature has worked both in designing more accurate networks as well as in designing more efficient networks from a computational-cost point of view.

Although there is a lot of literature discussing new architectures from the point of view of the layers composition and recognition performance, there are few papers that analyze the aspects related to the computational cost (memory usage, inference time, etc.), and more importantly how computational cost impacts on the recognition accuracy.

Canziani *et al.* [4] in the first half of 2016 proposed a comprehensive analysis of some DNN architectures by performing experiments on an embedded system based on a NVIDIA Jetson TX1 board. They measured accuracy, power consumption, memory footprint, number of parameters and operations count, and more importantly they analyzed the relationship between these performance indices. It is a valuable work, but it has been focused on a limited number (i.e. 14) of DNNs and more importantly the experimentation has been carried out only on the NVIDIA Jetson TX1 board. In [5], speed/accuracy trade-off of modern DNN-based detection systems has been explored by re-implementing a set of meta-architectures inspired by well-known detection networks in the state of the art. Results include performance comparisons between an Intel Xeon CPU and a NVIDIA Titan X GPU.

The aim of this work is to provide a more comprehensive and complete analysis of existing DNNs for image recognition and most importantly to provide an analysis on two hardware platforms with a very different computational capacity: a workstation equipped with a NVIDIA Titan X Pascal (often referred to as Titan Xp) and an embedded system based on a NVIDIA Jetson TX1. To this aim we analyze and compare more than 40 state-of-the-art DNN architectures in terms of computational cost and accuracy. In particular we experiment the selected DNN architectures on the ImageNet-1k challenge and we measure: accuracy rate, model complexity,

memory usage, computational complexity, and inference time. Further, we analyze relationships between these performance indices that provide insights for: 1) understanding what solutions have been explored so far and in what direction it would be appropriate to go in the future; 2) selecting the DNN architecture that better fits the resource constraints of practical deployments and applications.

The most important findings are that: i) the recognition accuracy does not increase as the number of operations increases; ii) there is not a linear relationship between model complexity and accuracy; iii) the desired throughput places an upper bound to the achievable accuracy; iv) not all the DNN models use their parameters with the same level of efficiency; v) almost all models are capable of super real-time performance on a high-end GPU, while just some of them can guarantee it on an embedded system; vi) even DNNs with a very low level model complexity have a minimum GPU memory footprint of about 0.6GB.

The rest of paper is organized as follows: in Section II hardware and software used for experiments are detailed; in Section III the considered DNN architectures are briefly introduced; in Section IV the measured performance indices are described; finally, in Section V obtained results are reported and analyzed, and Section VI presents our final considerations.

II. BENCHMARKING

We implement the benchmark framework for DNNs comparison in Python. The PyTorch package [6] is used for neural networks processing with cuDNN-v5.1 and CUDA-v9.0 as back-end. All the code for the estimation of the adopted performance indices, as well as all the considered DNN models are made publicly available [7].

We run all the experiments on a workstation and on an embedded system:

1) The workstation is equipped with an Intel Core I7-7700 CPU @ 3.60GHZ, 16GB DDR4 RAM 2400 MHz, NVIDIA Titan X Pascal GPU with 3840 CUDA cores (top-of-the-line consumer GPU). The operating system is Ubuntu 16.04.
2) The embedded system is a NVIDIA Jetson TX1 board with 64-bit ARM® A57 CPU @ 2GHz, 4GB LPDDR4 1600MHz, NVIDIA Maxwell GPU with 256 CUDA cores. The board includes the JetPack-2.3 SDK.

The use of these two different systems allows to highlight how critical the computational resources can be depending on the DNN model adopted especially in terms of memory usage and inference time.

III. ARCHITECTURES

In this section we briefly describe the analyzed architectures. We select different architectures, some of which have been designed to be more performing in terms of effectiveness, while others have been designed to be more efficient and therefore more suitable for embedded vision applications. In some cases there is a number following the name of the architecture. Such a number depicts the number of layers that contains parameters to be learned (i.e. convolutional or fully connected layers).

We consider the following architectures: AlexNet [2]; the family of VGG architectures [8] (VGG-11, −13, −16, and −19) without and with the use of Batch Normalization (BN) layers [9]; BN-Inception [9]; GoogLeNet [10]; SqueezeNet-v1.0 and −v1.1 [11]; ResNet-18, −34, −50, −101, and −152 [12]; Inception-v3 [13]; Inception-v4 and Inception-ResNet-v2 [14]; DenseNet-121, −169, and −201 with growth rate corresponding to 32, and DenseNet-161 with growth rate equal to 48 [15]; ResNeXt-101 (32×4d), and ResNeXt-101 (64×4d), where the numbers inside the brackets denote respectively the number of groups per convolutional layer and the bottleneck width [16]; Xception [17]; DualPathNet-68, −98, and −131, [18]; SE-ResNet-50, SENet-154, SE-ResNet-101, SE-ResNet-152, SE-ResNeXt-50 (32×4d), SE-ResNeXt-101 (32×4d) [19]; NASNet-A-Large, and NASNet-A-Mobile, whose architecture is directly learned [20].

Furthermore, we also consider the following efficientcy-oriented models: MobileNet-v1 [21], MobileNet-v2 [22], and ShuffleNet [23].

IV. PERFORMANCE INDICES

In order to perform a direct and fair comparison, we exactly reproduce the same sampling policies: we directly collect models trained using the PyTorch framework [6], or we collect models trained with other deep learning frameworks and then we convert them in PyTorch.

All the pre-trained models expect input images normalized in the same way, i.e. mini-batches of RGB images with shape $3 \times H \times W$, where H and W are expected to be:

- 331 pixels for the NASNet-A-Large model;
- 229 pixels for InceptionResNet-v2, Inception-v3, Inception-v4, and Xception models;
- 224 pixels for all the other models considered.

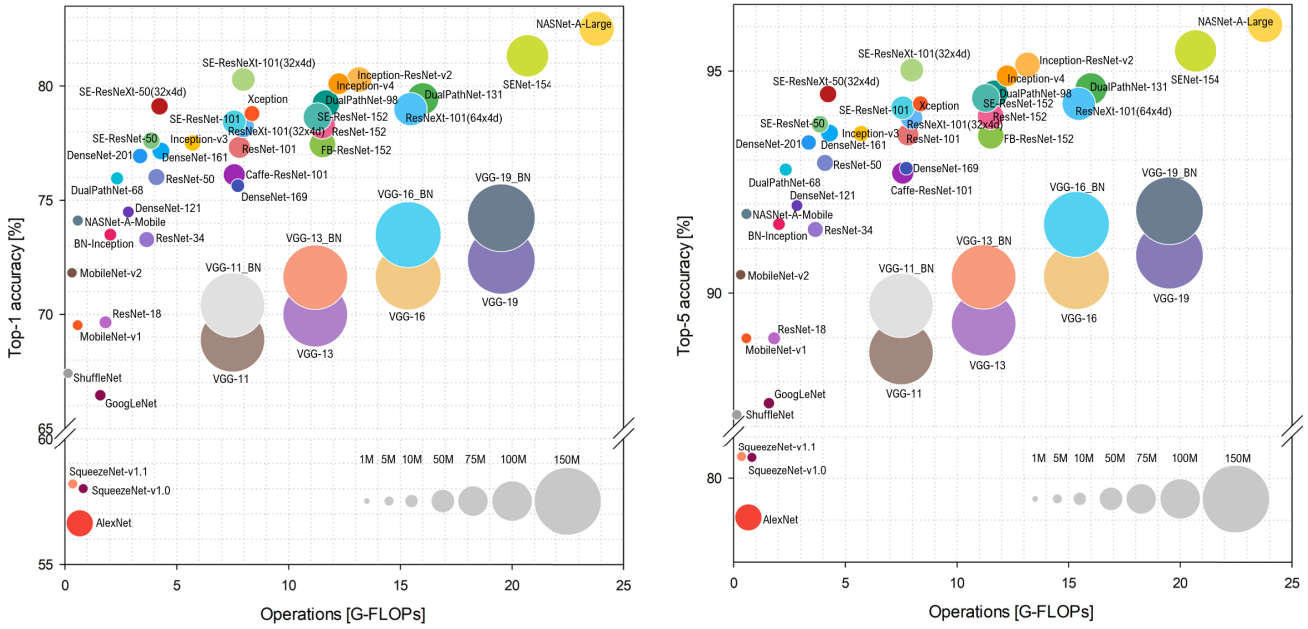
We consider multiple performance indices useful for a comprehensive benchmark of DNN models. Specifically, we measure: accuracy rate, model complexity, memory usage, computational complexity, and inference time.

A. ACCURACY RATE

We estimate Top-1 and Top-5 accuracy on the ImageNet-1k validation set for image classification task. The predictions are computed by evaluating the central crop only. Slightly better performances can be achieved by considering the average prediction coming from multiple crops (four corners plus central crop and their horizontal flips).

B. MODEL COMPLEXITY

We analyze model complexity by counting the total amount of learnable parameters. Specifically, we collect the size of the parameter file in terms of MB for the considered models. This information is very useful for understanding the minimum amount of GPU memory required for each model.



(a) (b)

FIGURE 1. Ball chart reporting the Top-1 and Top-5 accuracy *vs.* computational complexity. Top-1 and Top-5 accuracy using only the center crop versus floating-point operations (FLOPs) required for a single forward pass are reported. The size of each ball corresponds to the model complexity. (a) Top-1; (b) Top-5.

C. MEMORY USAGE

We evaluate the total memory consumption, which includes all the memory that is allocated, i.e. the memory allocated for the network model and the memory required while processing the batch. We measure memory usage for different batch sizes: 1, 2, 4, 8, 16, 32, and 64.

D. COMPUTATIONAL COMPLEXITY

We measure the computational cost of each DNN model considered using the floating-point operations (FLOPs) in the number of multiply-adds as in [16]. More in detail, the multiply-adds are counted as two FLOPs because, in many recent models, convolutions are bias-free and it makes sense to count multiply and add as separate FLOPs.

E. INFERENCE TIME

We report inference time per image for each DNN model for both the NVIDIA Titan X Pascal GPU and the NVIDIA Jetson TX1. We measure inference time in terms of milliseconds and by considering the same batch sizes described in Section IV-C. For statistical validation the reported time corresponds to the average over 10 runs.

V. RESULTS

A. ACCURACY-RATE VS COMPUTATIONAL COMPLEXITY VS MODEL COMPLEXITY

The ball charts reported in Figures 1 (a) and (b) show Top-1 and Top-5 accuracy on the ImageNet-1k validation set with respect to the computational complexity of the considered architectures for a single forward pass measured for both the workstation and the embedded board. The ball size corresponds to the model complexity. From the plots it can be seen that the DNN model reaching the highest Top-1 and Top-5 accuracy is the NASNet-A-Large that is also the one having the highest computational complexity. Among the models having the lowest computational complexity instead (i.e. lower than 5 G-FLOPs), SE-ResNeXt-50 (32×4d) is the one reaching the highest Top-1 and Top-5 accuracy showing at the same time a low level of model complexity, with approximately 2.76 M-params. Overall, it seems that there is no relationship between computational complexity and recognition accuracy, for instance SENet-154 needs about 3× the number of operations that are needed by SE-ResNeXt-101 (32×4d) while having almost the same accuracy. Moreover, it seems that there is no relationship also between model complexity and recognition accuracy: for instance VGG-13 has a much higher level of model complexity (size of the ball) than ResNet-18 while having almost the same accuracy.

B. ACCURACY-RATE VS LEARNING POWER

It is known that DNNs are inefficient in the use of their full learning power (measured as the number of parameters with respect to the degrees of freedom). Although many papers exist that exploit this feature to produce compressed DNN models with the same accuracy of the original models [24] we want here to measure how efficiently each model uses its parameters. We follow [4] and measure it as Top-1 accuracy density, i.e. Top-1 accuracy divided by the number

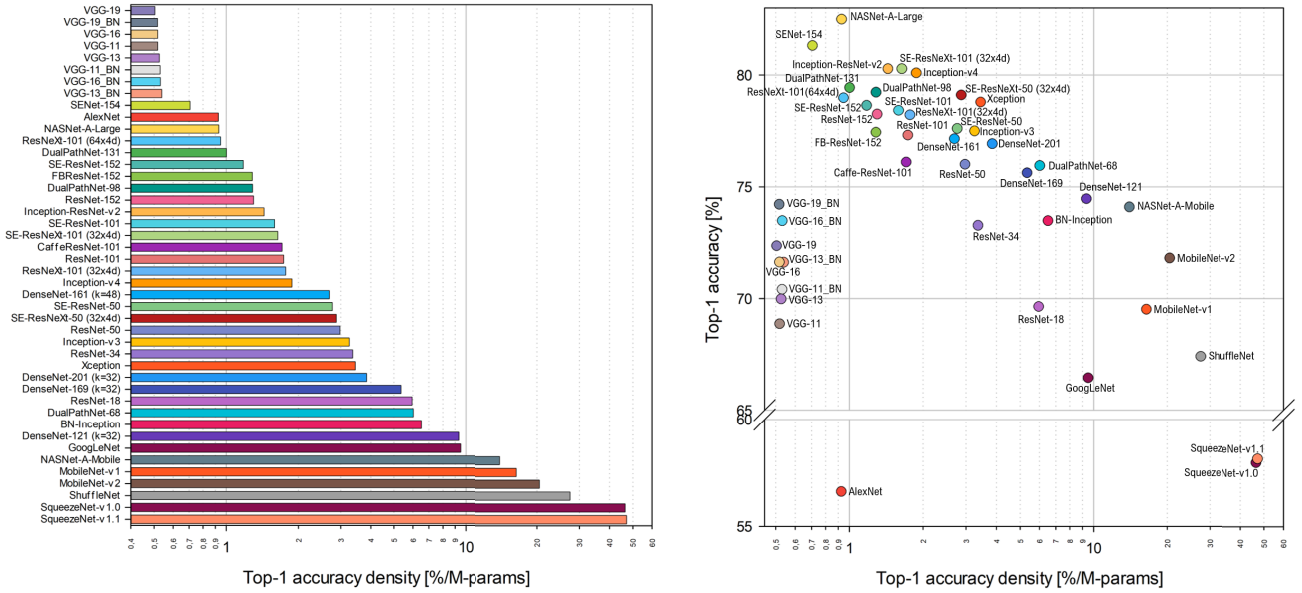


FIGURE 2. **Top-1 accuracy density (a) and Top-1 accuracy *vs.* Top-1 accuracy density (b). The accuracy density measures how efficiently each model uses its parameters.**

of parameters. The higher is this value and the higher is the efficiency. The plot is reported in Figure 2(a), where it can be seen that the models that use their parameters most efficiently are the SqueezeNets, ShuffleNet, the MobileNets and NASNet-A-Mobile. To focus to the density information, we plot the Top-1 accuracy with respect to the Top-1 accuracy density (see Figure 2(b)), that permits to find more easily the desired trade-off. In this way it is possible to easily see that among the most efficient models, NASNet-A-Mobile and MobileNet-v2 are the two providing a much higher Top-1 accuracy. Among the models having the highest Top-1 accuracy (i.e. higher than 80%) we can observe how the models using their parameters more efficiently are Inception-v4 and SE-ResNeXt-101 (32×4d).

C. INFERENCE TIME

Average per image inference time over 10 runs for all the DNN models considered are reported in Tables 1(a) and (b) for batch size equal to 1, 2, 4, 8, 16, 32, and 64 on both the Titan Xp and the Jetson. Inference time is measured in milliseconds and the entries in Tables 1(a) and (b) are color coded to easily convert them in frames per second (FPS). From the table it is possible to see that all the DNN models considered are able to achieve super real-time performances on the Titan Xp with the only exception of SENet-154, when a batch size of 1 is considered. On the Jetson instead, only a few models are able to achieve super real-time performances when a batch size of 1 is considered, namely: the SqueezeNets, the MobileNets, ResNet-18, GoogLeNet, and AlexNet. Missing measurements are due to the lack of enough system memory required to process the larger batches.

D. ACCURACY-RATE VS INFERENCE TIME

In Figure 3(a) and (b) we report the plots of the top-1 accuracy with respect to the number of images processed per second (i.e. the number of inferences per second) with a batch size of 1 on both the Titan Xp and the Jetson TX1. On each plot the linear upper bound is also reported; the two have almost the same intercept (≈ 83.3 for Titan Xp and ≈ 83.0 for the Jetson TX1), but the first has a slope that is almost $8.3\times$ smaller than the second one (-0.0244 *vs.* -0.2025); these bounds show that the Titan Xp guarantees a lower decay of the maximum accuracy achievable when a larger throughput is needed. Note that this bound appears a curve instead of line in the plots because of the logarithmic scale of the images per second axis. From the Titan Xp plot it is possible to see that if one targets a throughput of more than 250 FPS, the model giving the highest accuracy is ResNet-34, with 73.27% Top-1 accuracy; with a target of more than 125 FPS the model giving the highest accuracy is Xception, with 78,79% Top-1 accuracy; with a target of more than 62.5 FPS the model giving the highest accuracy is SE-ResNeXt-50 (32×4d), with 79,11% Top-1 accuracy; with a target of more than 30 FPS the model giving the highest accuracy is NASNet-A-Large, with 82,50% Top-1 accuracy. This analysis shows how even the most accurate model in the state of the art, i.e. NASNet-A-Large, is able to provide super real-time performance (30.96 FPS) on the Titan Xp. Considering the Jetson TX1 plot it is possible to see that if one targets super real-time performance, the model giving the highest accuracy is MobileNet-v2, with a Top-1 accuracy of 71.81%; if one targets a Top-1 accuracy larger than 75%, the maximum throughput is achieved by ResNet-50 (18,83 FPS); targeting a Top-1 accuracy larger than 80%, the maximum throughput is

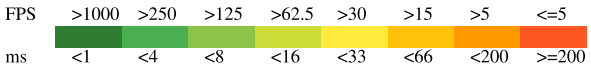
TABLE 1. Inference time *vs.* batch size. Inference time per image is estimated across different batch sizes for the Titan Xp (left), and Jetson TX1 (right). Missing data are due to the lack of enough system memory required to process the larger batches.

DNN	1	2	4	8	16	32	64
AlexNet	1.28	0.70	0.48	0.27	0.18	0.14	0.15
BN-Inception	5.79	3.00	1.64	1.10	0.87	0.77	0.71
CaffeResNet-101	8.20	4.82	3.32	2.54	2.27	2.16	2.08
DenseNet-121 (k=32)	8.93	4.41	2.64	1.96	1.64	1.44	1.39
DenseNet-169 (k=32)	13.03	6.72	3.97	2.73	2.14	1.87	1.75
DenseNet-201 (k=32)	17.15	9.25	5.36	3.66	2.84	2.41	2.27
DenseNet-161 (k=48)	15.50	9.10	5.89	4.45	3.66	3.43	3.24
DPN-68	10.68	5.36	3.24	2.47	1.80	1.59	1.52
DPN-98	22.31	13.84	8.97	6.77	5.59	4.96	4.72
DPN-131	29.70	18.29	11.96	9.12	7.57	6.72	6.37
FBResNet-152	14.55	7.79	5.15	4.31	3.96	3.76	3.65
GoogLeNet	4.54	2.44	1.65	1.06	0.86	0.76	0.72
Inception-ResNet-v2	25.94	14.36	8.82	6.43	5.19	4.88	4.59
Inception-v3	10.10	5.70	3.65	2.54	2.05	1.89	1.80
Inception-v4	18.96	10.61	6.53	4.85	4.10	3.77	3.61
MobileNet-v1	2.45	0.89	0.68	0.60	0.55	0.53	0.53
MobileNet-v2	3.34	1.63	0.95	0.78	0.72	0.63	0.61
NASNet-A-Large	32.30	23.00	19.75	18.49	18.11	17.73	17.77
NASNet-A-Mobile	22.36	11.44	5.60	2.81	1.61	1.75	1.51
ResNet-101	8.90	5.16	3.32	2.69	2.42	2.29	2.21
ResNet-152	14.31	7.36	4.68	3.83	3.50	3.30	3.17
ResNet-18	1.79	1.01	0.70	0.56	0.51	0.41	0.38
ResNet-34	3.11	1.80	1.20	0.96	0.82	0.71	0.67
ResNet-50	5.10	2.87	1.99	1.65	1.49	1.37	1.34
ResNeXt-101 (32x4d)	17.05	9.02	6.27	4.62	3.71	3.25	3.11
ResNeXt-101 (64x4d)	21.05	15.54	10.39	7.80	6.39	5.62	5.29
SE-ResNet-101	15.10	9.26	6.17	4.72	4.03	3.62	3.42
SE-ResNet-152	23.43	13.08	8.74	6.55	5.51	5.06	4.85
SE-ResNet-50	8.32	5.16	3.36	2.62	2.22	2.01	2.06
SE-ResNeXt-101 (32x4d)	24.96	13.86	9.16	6.55	5.29	4.53	4.29
SE-ResNeXt-50 (32x4d)	12.06	7.41	5.12	3.64	2.97	3.01	2.56
SENet-154	53.80	30.30	19.32	13.27	10.45	9.41	8.91
ShuffleNet	5.40	2.67	1.37	0.82	0.66	0.59	0.56
SqueezeNet-v1.0	1.53	0.84	0.66	0.59	0.54	0.52	0.53
SqueezeNet-v1.1	1.60	0.77	0.44	0.37	0.32	0.31	0.30
VGG-11	3.57	4.40	2.89	1.56	1.19	1.10	1.13
VGG-11_BN	3.49	4.60	2.99	1.71	1.33	1.24	1.27
VGG-13	3.88	5.03	3.44	2.25	1.83	1.75	1.79
VGG-13_BN	4.40	5.37	3.71	2.42	2.05	1.97	2.00
VGG-16	5.17	5.91	4.01	2.84	2.20	2.12	2.15
VGG-16_BN	5.04	5.95	4.27	3.06	2.45	2.36	2.41
VGG-19	5.50	6.26	4.71	3.29	2.59	2.52	2.50
VGG-19_BN	6.17	6.67	4.86	3.56	2.88	2.74	2.76
Xception	6.44	5.35	4.90	4.47	4.41	4.41	4.36

(a)

DNN	1	2	4	8	16	32	64
AlexNet	28.88	13.00	8.58	6.56	5.39	4.77	
BN-Inception	35.52	26.48	25.10	23.89	21.21	20.47	
CaffeResNet-101	84.47	91.37	70.33	63.53	56.38	53.73	
DenseNet-121 (k=32)	66.43	50.87	50.20	43.89	40.41	38.22	
DenseNet-169 (k=32)	137.96	130.27	110.82	100.56	92.97	88.94	
DenseNet-201 (k=32)	84.57	61.71	62.62	53.73	49.28	46.26	
DenseNet-161 (k=48)	103.20	76.11	77.10	68.32	62.73	59.14	
DPN-68	113.08	52.73	42.85	43.32	38.18	36.40	36.22
DPN-98	243.51	148.51	135.30	125.92	123.34	118.68	117.27
DPN-131	330.15	204.69	184.89	172.25	165.59	162.67	160.66
FBResNet-152	133.68	147.75	113.48	105.78	94.26	97.47	
GoogLeNet	32.11	27.19	23.29	21.66	19.77	19.96	
Inception-ResNet-v2	198.95	141.29	127.97	130.25	117.99	116.47	
Inception-v3	79.39	59.04	56.46	51.79	47.60	46.85	
Inception-v4	158.00	120.23	106.77	102.21	95.51	95.40	
MobileNet-v1	15.06	11.94	11.34	11.03	10.82	10.58	10.55
MobileNet-v2	20.51	14.58	13.67	13.56	13.18	13.10	12.72
NASNet-A-Large	437.20	399.99	385.75	383.55	389.67		
NASNet-A-Mobile	133.87	62.91	33.72	30.62	29.72	28.92	28.55
ResNet-101	84.52	77.90	71.23	67.14	58.11		
ResNet-152	124.67	113.65	101.41	96.76	82.35		
ResNet-18	21.16	15.30	14.35	13.82	11.99	10.73	12.45
ResNet-34	39.88	28.82	27.51	24.97	20.41	18.48	17.97
ResNet-50	53.09	44.84	41.20	38.79	35.72		
ResNeXt-101 (32x4d)	115.37	90.93	84.64	79.66	77.63		
ResNeXt-101 (64x4d)	177.40	155.77	144.82	137.43	134.07		
SE-ResNet-101	118.13	105.11	96.71	91.68	80.99		
SE-ResNet-152	169.73	155.08	139.72	133.59	116.97		
SE-ResNet-50	69.65	61.37	55.33	51.87	47.80		
SE-ResNeXt-101 (32x4d)	139.62	122.01	112.05	105.34	102.39		
SE-ResNeXt-50 (32x4d)	80.08	69.86	67.20	62.66	61.19		
SENet-154	309.48	240.80	221.84	211.00	207.06	201.49	201.66
ShuffleNet	36.58	22.61	13.80	13.36	12.91	12.66	12.50
SqueezeNet-v1.0	17.00	16.47	15.03	13.97	13.25	12.89	12.70
SqueezeNet-v1.1	11.05	9.88	8.80	7.90	7.38	7.20	7.04
VGG-11	106.44	125.84	85.84	60.10	32.56	30.51	32.27
VGG-11_BN	101.58	122.82	86.26	54.50	47.81	47.31	41.26
VGG-13	122.59	148.80	108.28	75.99	70.57	64.88	62.79
VGG-13_BN	129.69	153.68	113.90	81.19	76.39	70.59	67.38
VGG-16	151.52	169.92	129.89	96.81	91.72		
VGG-16_BN	163.37	176.35	136.85	103.45	98.11		
VGG-19	178.04	192.86	152.28	117.92	112.39		
VGG-19_BN	185.18	198.66	159.20	124.88	119.15		
Xception	98.96	93.40	90.49	87.65	86.89		

(b)



achieved by SE-ResNeXt-101 (32×4d) (7,16 FPS); targeting the highest Top-1 accuracy in the state of the art the throughput achievable is 2,29 FPS.

E. MEMORY USAGE

In Table 2 we analyze the memory consumption for all the DNN models considered for different batch sizes on the Titan Xp. From the memory footprints reported it can be seen that when a batch size of 1 is considered, most models require less than 1GB of memory, with the exception of NASNet-A-Large, the SE-ResNets, the SE-ResNeXTs, SENet-154, the VGGs and Xception. However none of them requires more than 1.5GB for a batch size of 1.

F. MEMORY USAGE VS MODEL COMPLEXITY

In Figure 4 we analyze the relationship between the initial static allocation of the model parameters (i.e. the model complexity) and the total memory utilization for a batch size of 1 on the Titan Xp. We can see that the relationship is linear, and follows two different lines with approximately the same slope (i.e. 1.10 and 1.15) and different intercepts (i.e. 910 and 639 respectively). This means that the model complexity can be used to reliably estimate the total memory utilization. From the plots we can observe that families of models belong to the same line, as for example the VGGs, the SE-ResNets ans SqueezeNets lie on the line with higher intercept, while the ResNets, DualPathNets, DenseNets, Inception nets and MobileNets line on the line with the lower intercept. In particular we can observe how models having the smallest complexity (i.e. SqueezeNet-v1.0 and SqueezeNet-v1.1 both with 5MB) have a 943MB and 921MB memory footprint, while models having slightly higher complexity (i.e. MobileNet-v1 and MobileNet-v2 with respectively 17MB and 14MB) have a much smaller memory footprint, equal to 650MB and 648MB respectively.

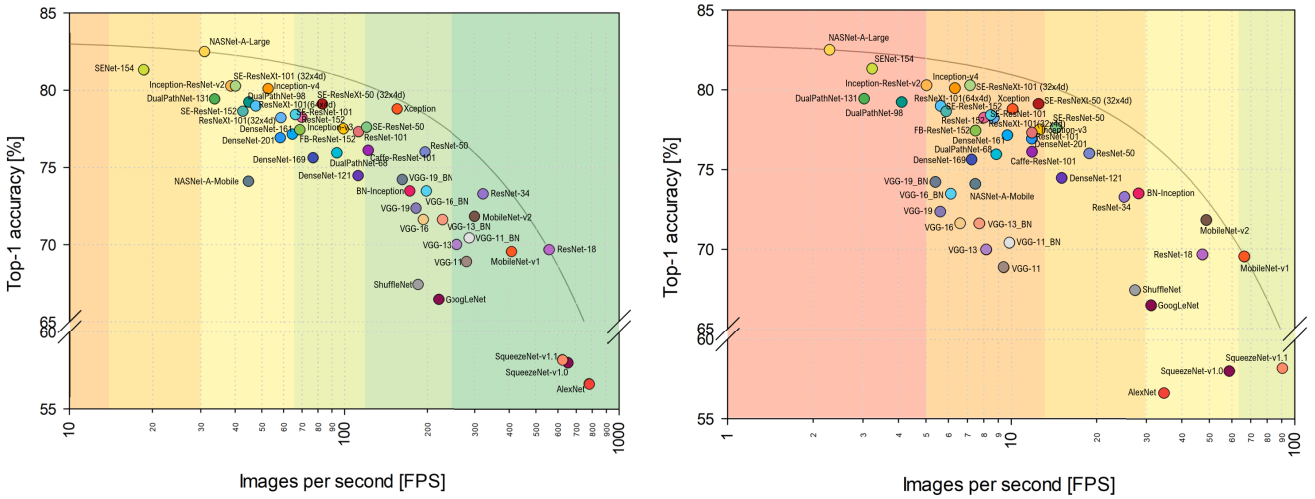


FIGURE 3. Top-1 accuracy *vs.* number of images processed per second (with batch size 1) using the Titan Xp (a) and Jetson TX1 (b).

TABLE 2. Memory consumption of the different DNN models considered on the Titan Xp for different batch sizes.

DNN	1	2	4	8	16	32	64
AlexNet	0.82	0.83	0.83	0.83	0.84	0.92	0.99
BN-Inception	0.67	0.71	0.81	0.97	1.29	1.97	3.24
CaffeResNet-101	0.78	0.80	0.80	0.80	0.81	0.95	1.06
DenseNet-121 (k=32)	0.67	0.67	0.67	0.69	0.71	0.71	0.99
DenseNet-169 (k=32)	0.87	0.87	0.88	0.91	0.93	0.97	1.04
DenseNet-201 (k=32)	0.72	0.72	0.73	0.75	0.77	0.80	0.87
DenseNet-161 (k=48)	0.76	0.77	0.77	0.80	0.82	0.88	0.96
DPN-68	0.65	0.65	0.66	0.67	0.67	0.68	0.71
DPN-98	0.87	0.88	0.90	0.92	0.98	1.10	1.29
DPN-131	0.95	0.95	0.96	0.97	1.05	1.04	1.28
FBResNet-152	0.89	0.90	0.92	0.94	0.97	1.12	1.31
GoogLeNet	0.66	0.70	0.76	0.87	1.09	1.51	2.35
Inception-ResNet-v2	0.87	0.88	0.88	0.89	0.91	0.95	1.02
Inception-v3	0.72	0.73	0.75	0.75	0.77	0.83	0.92
Inception-v4	0.80	0.81	0.82	0.84	0.90	0.90	1.18
MobileNet-v1	0.63	0.64	0.64	0.65	0.67	0.71	0.78
MobileNet-v2	0.63	0.63	0.63	0.64	0.66	0.70	0.78
NASNet-A-Large	1.09	1.19	1.38	1.78	2.56	4.12	7.26
NASNet-A-Mobile	0.63	0.65	0.67	0.71	0.79	0.93	1.23
ResNet-101	0.82	0.83	0.86	0.93	1.08	1.37	1.94
ResNet-152	0.89	0.90	0.92	1.00	1.15	1.43	2.01
ResNet-18	0.67	0.68	0.68	0.69	0.71	0.75	0.89
ResNet-34	0.74	0.74	0.75	0.80	0.90	1.09	1.47
ResNet-50	0.74	0.74	0.77	0.85	0.99	1.28	1.86
ResNeXt-101 (32x4d)	0.77	0.78	0.78	0.79	0.84	0.87	1.06
ResNeXt-101 (64x4d)	0.90	0.92	0.91	0.96	1.01	1.19	1.38
SE-ResNet-101	1.09	1.11	1.10	1.13	1.13	1.27	1.36
SE-ResNet-152	1.19	1.21	1.25	1.35	1.54	1.93	2.69
SE-ResNet-50	1.02	1.04	1.08	1.18	1.38	1.76	2.53
SE-ResNeXt-101 (32x4d)	1.07	1.07	1.08	1.09	1.12	1.16	1.25
SE-ResNeXt-50 (32x4d)	1.00	1.03	1.08	1.19	1.38	1.76	2.53
SENet-154	1.31	1.32	1.33	1.36	1.40	1.48	1.65
ShuffleNet	0.91	0.91	0.92	0.93	0.95	0.99	1.05
SqueezeNet-v1.0	0.92	0.92	0.92	0.93	0.94	0.97	1.02
SqueezeNet-v1.1	0.90	0.90	0.91	0.92	0.94	0.99	1.07
VGG-11	1.41	1.43	1.43	1.43	1.53	1.55	1.81
VGG-11_BN	1.44	1.49	1.59	1.78	2.39	3.59	5.99
VGG-13	1.44	1.43	1.51	1.60	2.02	2.41	3.99
VGG-13_BN	1.44	1.49	1.59	1.78	2.39	3.59	5.99
VGG-16	1.46	1.51	1.61	1.80	2.41	3.61	6.02
VGG-16_BN	1.46	1.51	1.61	1.80	2.41	3.61	6.02
VGG-19	1.49	1.54	1.63	1.83	2.43	3.64	6.04
VGG-19_BN	1.49	1.54	1.63	1.83	2.43	3.64	6.04
Xception	1.03	1.05	1.06	1.08	1.16	1.24	1.53

GB <1 <2 <3 <4 <5 <6 <7 >=7

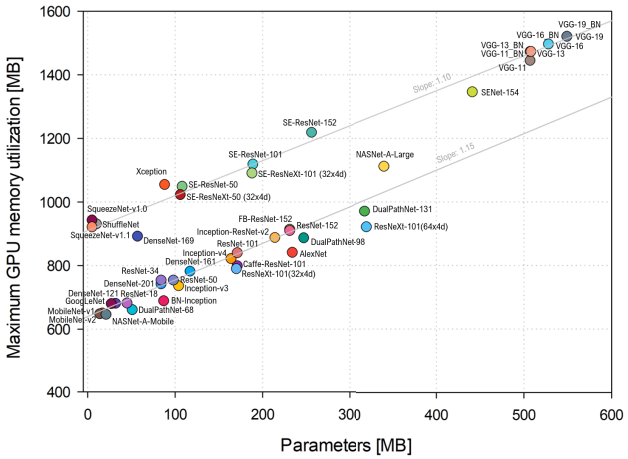


FIGURE 4. Plot of the initial static allocation of the model parameters (i.e. the model complexity) and the total memory utilization with batch size 1 on the Titan Xp.

G. BEST DNN AT GIVEN CONSTRAINTS

Table 3 shows the best DNN architectures in terms of recognition accuracy when specific hardware resources are given as computational constraints. This analysis is done for both the Titan Xp and Jetson TX1. We define the following constraints:

- Memory usage: high (≤1.4GB), medium (≤1.0GB) and low (≤0.7GB);
- Computational time: half real-time (@15FPS), real-time (@30FPS), super real-time (@60FPS);

A Titan Xp, with a low memory usage as constraint, achieves a recognition accuracy of at most 75.95% by using the DPN-68 network independently of the computational time. Having more resources, for instance medium and high memory usage, Titan Xp achieves a recognition accuracy of at most 79.11% by using the SE-ResNeXt-50 (32×4d) with a super real-time throughput. Having no requirements in terms of memory usage, the Jetson TX1 achieves a recognition accuracy of at most 69.52% by using the MobileNet-v1, which guarantees a super real-time throughput. To have a DNN running on a Jetson that is comparable in terms

TABLE 3. Top 5 models (sorted in decreasing Top-1 accuracy) satisfying memory consumption (≤0.7GB, ≤1.0GB, ≤1.4GB) and inference speed (≥15FPS, ≥30FPS, ≥60FPS) constraints on the Titan Xp (a) and Jetson TX1 (b).

Titan Xp						Jetson					
≤ 0.7GB, @15FPS	Acc.	≤ 0.7GB, @30FPS	Acc.	≤ 0.7GB, @60FPS	Acc.	≤ 0.7GB, @15FPS	Acc.	≤ 0.7GB, @30FPS	Acc.	≤ 0.7GB, @60FPS	Acc.
DPN-68	75.95	DPN-68	75.95	DPN-68	75.95	DenseNet-121 (k=32)	74.47	MobileNet-v2	71.81	MobileNet-v1	69.52
DenseNet-121 (k=32)	74.47	DenseNet-121 (k=32)	74.47	DenseNet-121 (k=32)	74.47	BN-Inception	73.48	ResNet-18	69.64		
NASNet-A-Mobile	74.10	NASNet-A-Mobile	74.10	BN-Inception	73.48	MobileNet-v2	71.81	MobileNet-v1	69.52		
BN-Inception	73.48	BN-Inception	73.48	MobileNet-v2	71.81	ResNet-18	69.64	GoogLeNet	66.45		
MobileNet-v2	71.81	MobileNet-v2	71.81	ResNet-18	69.64	MobileNet-v1	69.52				
≤ 1.0GB, @15FPS	Acc.	≤ 1.0GB, @30FPS	Acc.	≤ 1.0GB, @60FPS	Acc.	≤ 1.0GB, @15FPS	Acc.	≤ 1.0GB, @30FPS	Acc.	≤ 1.0GB, @60FPS	Acc.
Inception-ResNet-v2	80.28	Inception-ResNet-v2	80.28	SE-ResNeXt-50(32x4d)	79.11	ResNet-50	76.01	MobileNet-v2	71.81	MobileNet-v1	69.52
Inception-v4	80.10	Inception-v4	80.10	ResNet-152	78.25	DenseNet-121 (k=32)	74.47	ResNet-18	69.64	SqueezeNet-v1.1	58.18
DPN-131	79.44	DPN-131	79.44	Inception-v3	77.50	BN-Inception	73.48	MobileNet-v1	69.52		
DPN-98	79.23	DPN-98	79.23	FBResNet-152	77.44	ResNet-34	73.27	GoogLeNet	66.45		
SE-ResNeXt-50(32x4d)	79.11	SE-ResNeXt-50(32x4d)	79.11	ResNet-101	77.31	MobileNet-v2	71.81	SqueezeNet-v1.1	58.18		
≤ 1.4GB, @15FPS	Acc.	≤ 1.4GB, @30FPS	Acc.	≤ 1.4GB, @60FPS	Acc.	≤ 1.4GB, @15FPS	Acc.	≤ 1.4GB, @30FPS	Acc.	≤ 1.4GB, @60FPS	Acc.
NASNet-A-Large	82.50	NASNet-A-Large	82.50	SE-ResNeXt-50(32x4d)	79.11	ResNet-50	76.01	MobileNet-v2	71.81	MobileNet-v1	69.52
SENet-154	81.32	Inception-ResNet-v2	80.28	Xception	78.79	DenseNet-121 (k=32)	74.47	ResNet-18	69.64	SqueezeNet-v1.1	58.18
Inception-ResNet-v2	80.28	SE-ResNeXt-101(32x4d)	80.28	SE-ResNet-101	78.42	BN-Inception	73.48	MobileNet-v1	69.52		
SE-ResNeXt-101(32x4d)	80.28	Inception-v4	80.10	ResNet-152	78.25	ResNet-34	73.27	GoogLeNet	66.45		
Inception-v4	80.10	DPN-131	79.44	SE-ResNet-50	77.61	MobileNet-v2	71.81	SqueezeNet-v1.1	58.18		

of recognition accuracy to the best DNNs running on the Titan Xp, a memory size of at least 1GB is needed. In this case the most performing is the ResNet-50, able to guarantee an half real-time throughput, with a recognition accuracy of 76.01%.

VI. CONCLUSION

The design of Deep neural networks (DNNs) with increasing complexity able to improve the performance of the ImageNet-1k competition plays a central rule in advancing the state-of-the-art also on other vision tasks. In this article we present a comparison between different DNNs in order to provide an immediate and comprehensive tool for guiding in the selection of the appropriate architecture responding to resource constraints in practical deployments and applications. Specifically, we analyze more than 40 state-of-the-art DNN architectures trained on ImageNet-1k in terms of accuracy, number of parameters, memory usage, computational complexity, and inference time.

The key findings of this paper are the following:

- the recognition accuracy does not increase as the number of operations increases: in fact, there are some architectures that with a relatively low number of operations, such as the SE-ResNeXt-50 (32×4d), achieve very high accuracy (see Figures 1a and b). This finding is independent on the computer architecture experimented;
- there is not a linear relationship between model complexity and accuracy (see Figures 1a and b);
- not all the DNN models use their parameters with the same level of efficiency (see Figures 2a and b);
- the desired throughput (expressed for example as the number of inferences per second) places an upper bound to the achievable accuracy (see Figures 3a and b);
- model complexity can be used to reliably estimate the total memory utilization (see Figure 4);
- almost all models are capable of real-time or super real-time performance on a high-end GPU, while just a few of them can guarantee them on an embedded system (see Tables 1a and b);
- even DNNs with a very low level model complexity have a minimum GPU memory footprint of about 0.6GB (see Table 2).

All the DNNs considered, as well as the software used for the analysis, are available online [7]. We plan to add to the repository interactive plots that allow other researchers to better explore the results of this study, and more importantly to effortlessly add new entries.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
[3] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[4] A. Canziani, A. Paszke, and E. Culurciello. (2016). "An analysis of deep neural network models for practical applications." [Online]. Available: https://arxiv.org/abs/1605.07678
[5] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 3296–3297.
[6] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS) Workshop*, 2017.
[7] S. Bianco, R. Cadène, L. Celona, and P. Napoletano. *Paper Github Repository*. Accessed: Sep. 25, 2018. https://github.com/CeLuigi/models-comparison.pytorch
[8] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 448–456.
[10] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1–9.
[11] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50 x fewer parameters and <0.5 MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.
[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[14] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. Int. Conf. Learn. Represent. (ICLR) Workshop*, 2016, p. 12.

[15] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proc. IEEE CVPR*, vol. 1, Jun. 2017, no. 2, p. 3.
[16] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 5987–5995.
[17] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1251–1258.
[18] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 4467–4475.
[19] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 7132–7141.
[20] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 8697–8710.
[21] A. G. Howard *et al.* (2017). "MobileNets: Efficient convolutional neural networks for mobile vision applications." [Online]. Available: https://arxiv.org/abs/1704.04861
[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520.
[23] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 6848–6856.
[24] S. Han, H. Mao, and W. J. Dally. (2015). "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding." [Online]. Available: https://arxiv.org/abs/s1510.00149

SIMONE BIANCO received the B.Sc. and M.Sc. degrees in mathematics and the Ph.D. degree in computer science from the University of Milano-Bicocca, Italy, in 2003, 2006, and 2010, respectively. He is currently an Assistant Professor of computer science with the Department of Informatics, Systems and Communication, University of Milano-Bicocca. His current research interests include computer vision, machine learning, optimization algorithms, and color imaging.

REMI CADENE received the M.Sc. degree in computer science from Sorbonne University in 2016. He is currently pursuing the Ph.D. degree with LIP6 (Computer Science Laboratory), Sorbonne University, France. He is currently a Teaching Assistant with LIP6 (Computer Science Laboratory), Sorbonne University. His primary research interests are in the fields of machine learning, computer vision, and natural language processing. He focuses on neural networks, multimodal learning, and weakly supervised learning.

LUIGI CELONA received the B.Sc. degree in computer science from the University of Messina in 2011 and the M.Sc. and Ph.D. degrees in computer science from the Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Italy, in 2014 and 2018, respectively. He is currently a Post-Doctoral Fellow with DISCo, University of Milano-Bicocca. His current research interests focus on image analysis and classification, machine learning, and face analysis.

PAOLO NAPOLETANO received the master's degree in telecommunications engineering from the University of Naples Federico II in 2003 and the Ph.D. degree in information engineering from the University of Salerno in 2007. He is currently an Assistant Professor of computer science with the Department of Informatics, Systems and Communication, University of Milano-Bicocca. His current research interests include machine learning for multi-modal data classification and understanding.