Improving Edge-AI Image Classification Through the Use of Better Building Blocks

Lucas Mohimont[a], Lilian Hollard[b] and Luiz Angelo Steffenel[c]
Université de Reims Champagne-Ardenne, CEA, LRC DIGIT, LICIIS, Reims, France

Keywords: Deep Learning, Image Classification, Blocks, Benchmark, Edge AI.

Abstract: Traditional CNN architectures for classification, while successful, suffer from limitations due to diminishing spatial resolution and vanishing gradients. The emergence of modular "building blocks" offered a new approach, allowing complex feature extraction through stacked layers. Despite the popularity of models like VGG, their high parameter count restricts their use in resource-constrained environments like Edge AI. This work investigates efficient building blocks as alternatives to VGG blocks, comparing the performance of diverse blocks from well-known models alongside our proposal block. Extensive experiments across various datasets demonstrate that our proposed block surpasses established blocks like Inception v1 in terms of accuracy while requiring significantly fewer resources regarding computational cost (GFLOPs) and memory footprint (number of parameters). This showcases its potential for real-world applications in Edge AI.

1 INTRODUCTION

Edge computing has gained popularity thanks to high-performance devices integrating microcontrollers and multicore computing units GPUs on a single board. This has allowed the emergence of AI Edge Computing, a research field that brings AI capabilities closer to the network's edge. Among the most known Edge-AI devices, NVIDIA Jetson is a notable example of this technology, offering high performances for many computing-intensive tasks such as computer vision, all while keeping energy consumption low. Despite these advances, the edge nodes' performance is not comparable to high-end servers. For this reason, Edge AI applications must be designed with resource constraints in mind.

In (De Lucia et al., 2022), three different strategies are proposed to deploy AI models at the Edge. The first relies on framework design, which adapts the AI models to the new environment, mainly through data or functional decomposition, using Federated Learning (McMahan et al., 2017), for example. Another strategy, model adaption, implies using data compression techniques and filtering the model through quantization and pruning. Finally, these authors propose using process acceleration to rewrite the models according to the available device features (presence of tensor operation support, multicore processing). Overall, these strategies require some modification before deploying a model to the Edge. In this paper, we propose a different strategy, namely the choice of better models (or convolutional blocks, in our case), to achieve higher performances with less cost. This strategy may bring important cost reductions and prevent a continuity gap between model training on high-end servers (ideal for faster training and prototyping) and model deployment for inference at the Edge.

In this work, we target image classification, one of the basic operations in machine learning. Early Convolutional Neural Networks (CNN) use a succession of convolution and pooling layers before the fully connected classification. This includes the first CNN, LeNet in 1989 (LeCun et al., 1989), but also early models like AlexNet (Krizhevsky et al., 2012), VGG (Simonyan and Zisserman, 2015), or ZFNet (Zeiler and Fergus, 2014). For instance, the traditional building block of Convolutional Neural Networks (CNNs) often comprises (i) a convolutional layer with padding, (ii) a non-linearity like ReLU, and (iii) a pooling layer for dimensionality reduction. While this approach is effective, it suffers from significant spatial resolution loss and a cumulative vanishing gradient problem, which limits the number of successive convolutional layers.

[a] https://orcid.org/0000-0001-8006-6656
[b] https://orcid.org/0000-0001-6906-561X
[c] https://orcid.org/0000-0003-3670-4088

Mohimont, L., Hollard, L. and Steffenel, L.
Improving Edge-AI Image Classification Through the Use of Better Building Blocks.
DOI: 10.5220/0012728000003711

In *Proceedings of the 14th International Conference on Cloud Computing and Services Science (CLOSER 2024)*, pages 303-310
ISBN: 978-989-758-701-6; ISSN: 2184-5042

Over the years, the design of neural network architectures evolved from a focus on individual neurons to larger-scale abstractions. Indeed, the rise of deep learning ushered in a new era, where researchers began leveraging ”building blocks” - pre-defined modular units composed of stacked layers capable of capturing more complex features.

The VGG network (Simonyan and Zisserman, 2015) stands as a pioneering example of exploiting modularity by utilizing multiple convolutional blocks. Contrary to earlier models such as AlexNet, the convolutional layers in VGG are grouped in nonlinear transformations that leave the dimensionality unchanged, followed by a resolution-reduction step, as depicted in Figure 1.

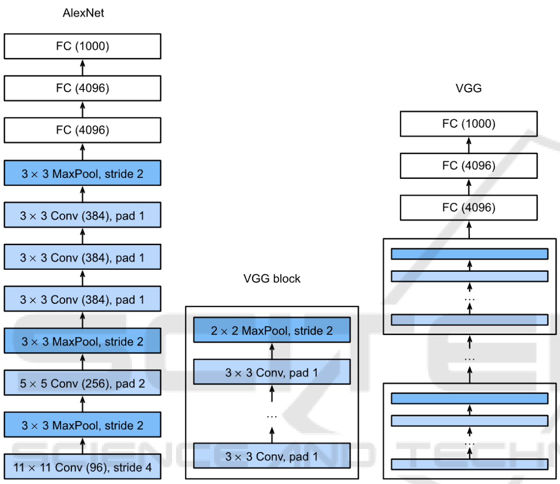

Figure 1: Comparison between AlexNet and VGG.

This modular design philosophy readily translates into modern deep learning frameworks, as loops and subroutines facilitate the implementation of repetitive structures, further promoting the adoption of such building blocks in architecture design. Hence, modern CNN architectures are composed of blocks performing many different operations, such as classical or factorized convolutions with different kernel sizes, tensor addition or concatenation, and parallelism.

Despite its age, VGG remains a reference in the domain of image classification, as its sequential architecture is easy to understand and implement. Nonetheless, the traditional VGG implementation induces an important number of parameters, which penalizes its usage in resource-constrained environments such as the case of Edge AI.

For this reason, in this paper, we aim to identify more efficient blocks instead of VGG blocks. Using a VGG-like architecture as a framework, we compare the performances of different blocks used in well-known classification models such as MobileNet (Howard et al., 2017), ResNet (He et al., 2016), Inception (Szegedy et al., 2014), SqueezeNet (Iandola et al., 2016), GhostNet (Han et al., 2020). We also include blocks from other models for object detection YOLO v8 (Jiang et al., 2022; Jocher et al., 2023) or segmentation CGNet (Wu et al., 2019), and propose our block based on a simplified Inception block.

The remainder of this paper is organized as follows: Section 2 presents the main blocks considered for this work and their characteristics. Section 3 described the datasets and benchmark parameters used in the comparison. Section 4 presents the performance benchmarks and analyses the obtained results. Finally, Section 5 discusses the impact of the experiments in the context of Edge AI and draws some future work directions.

2 DEEP LEARNING ”BUILDING” BLOCKS

Early CNNs only used a succession of convolution and pooling layers before the fully connected classification. This includes the first CNN, LeNet in 1989 (LeCun et al., 1989), and early CNN of the deep learning era like AlexNet (Krizhevsky et al., 2012), VGG (Simonyan and Zisserman, 2015), ZFNet (Simonyan and Zisserman, 2015), etc. Those models were simple because they are sequential but prone to problems such as the vanishing gradient problem. Since 2012, deep learning has become the state-of-the-art in computer vision, and research in deep learning is still very active.

Modern CNNs use a succession of complex components that we call blocks. Blocks are made of different operations, such as classical or factorized convolutions with different kernel sizes, tensor addition or concatenation, parallelism, etc. Some of these blocks are far more sophisticated than the original VGG blocks, making interesting alternatives to optimize image classification models’ performance and memory footprint. Therefore, this work uses selected blocks from well-known models to replace VGG’s original blocks.

Our baseline reference are the blocks from VGG (Simonyan and Zisserman, 2015), implemented with only two or three sequential convolutional layers with 3x3 filters and ReLU activation.

More recent models implement blocks that have at least two paths. The first to propose a two paths block is ResNet (He et al., 2016) with the residual unit. ResNet blocks are like VGG blocks with two layers and a skip connection. The skip connection is the sum of the input of the block and the output of the last convolutional layer.

Concatenation can also be used instead of addition. This is used in many blocks such as the Inception models, GhostNet (Han et al., 2020), SqueezeNet (Iandola et al., 2016), the C2F block of YOLOv8 model (Jiang et al., 2022; Jocher et al., 2023), or the Context-Guided block of the CGNet (Wu et al., 2019) segmentation models. We have selected the main block of those models for this study.

Hence, the Inception V1 block, used in GoogLeNet (Szegedy et al., 2014), processes the features in a parallel manner with three convolutional layers with filters of size 1x1, 3x3, and 5x5. The output of each layer is concatenated with the output of a pooling layer to be used in the next block. Point-wise convolution is used before each layer to reduce the number of channels.

The GhostNet block is based on the principle of redundant features in a CNN. Simple transformations can be used to increase the number of channels. This is simply done by concatenating the feature maps with a transformed version of the same features. In the GhostNet module, it is done with depth-wise convolution (each filter is only applied to one channel). Another well-known model, SqueezeNet (Iandola et al., 2016), uses a similar approach to GhostNet with two paths with 1x1 and 3x3 convolutions.

GhostNet and SqueezeNet were designed to be lightweight when compared to heavier models such as VGG and GoogLeNet. Other lightweight models include the MobileNet and YOLO families.

Another simple block used in this work is the main component of MobileNet V1 (Howard et al., 2017), a single Depthwise Separable Convolution layer. On the other hand, MobileNet V2 (Sandler et al., 2019) and V3 (Howard et al., 2019) are two more modern versions popular for their efficiency and good performance. They used the same inverted bottleneck residual block: the number of channels is increased with 1x1 convolutions; this is the expanding or projection step, which will be processed by depthwise filters. Then, another 1x1 layer will squeeze back the number of channels. In both models, the block also used the skip connection of ResNet. However, MobileNet V3 adds another component called Squeeze-and-Excitation (Hu et al., 2019). This fully connected network assigns a dynamic weight to each channel of the feature maps. It is really small because it only uses the output of a global pooling as input. This is one of the most straightforward ways to generate dynamic weights. This can be interpreted as a basic channel-wise attention (Vaswani et al., 2017).

So far, we have discussed blocks designed for classification networks. However, many blocks were designed for other tasks, such as object detection or segmentation, even though they perform some form of internal classification. Therefore, we have selected two components from the YOLOv8 object detection and the CGNet segmentation model.

It is difficult to find the original motivation for the design of the C2F module used in YOLO v8 (Terven and Cordova-Esparza, 2023). It is a succession of residual bottleneck blocks wired in the same manner as the Dense block of the DenseNet (Huang et al., 2018) model: each feature map is concatenated before the final convolutional layer. To be more efficient, the first bottlenecks are only applied to half of the input features. Despite their apparent complexity, the YOLO models are efficient for real-time detection on edge devices.

Similarly, CGNet was developed as a lightweight alternative to bigger segmentation models such as DeepLabV3 (Chen et al., 2017) or HRNet (Wang et al., 2020). It uses a succession of Context-Guided blocks to avoid too many down-sampling steps. Those blocks have two paths that are combined with a convolutional 1x1 layer. The first path uses Separable Depthwise Convolution, and the second uses a similar layer with a bigger dilation rate. In this manner, the block can compare the features at different scales. It also uses skip connection and the previously mentioned squeeze-and-excitation technique to create a deep network.

2.1 Proposed Block

In this section, we propose a new block, inspired by many existing blocks proposed in the literature. This block was first designed by our team in the context of image segmentation for grape disease detection (Mohimont, 2023). Indeed, a PSPNet model was used to segment grape bunches infected by gray mold. This PSPNet model uses a Pyramid Pooling Module (PPM) (Zhao et al., 2017) that is the main inspiration for our proposed block.

The PPM block applies many pooling layers to the feature maps with different window size, resulting in a pyramid of pooled features. Point-wise convolution and up-sampling are then used to concatenate every feature maps. In this manner it is also similar to the Inception block. The core idea of the PPM is to increase the receptive field size to get a better context, preventing some mistakes for object segmentation. For examples, the pixels of a car are often colocated with pixels from roads and pedestrian crossing.

In our proposed block, we use a pooling pyramid with three successive 2x2 max-pooling layers. Our assumption is that pooling is the cheapest way to in-

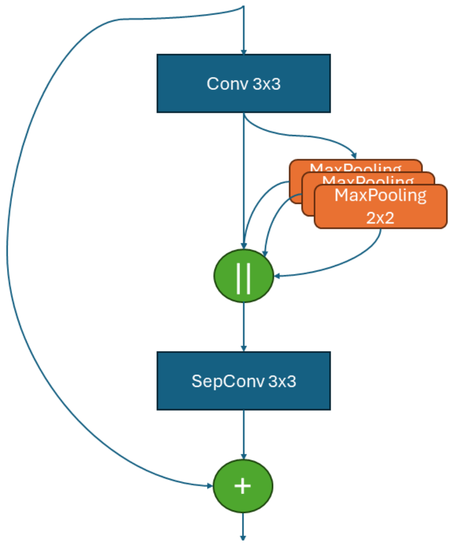

Figure 2: Structure of our proposed block.

crease the receptive field because it does not need parameters or a large window size. More importantly, every CNN, since ResNet, have been using at least two paths inside their main convolutional module. The simplest way is to put a residual connection but you can also use many layer in parallel like Inception. A simple residual block will act as features refinement, it improves the input feature in an iterative manner (Jastrzebski et al., 2018). And adding parallel processing with different kernel can also help to get bigger receptive field. In both case, the final features were computed from features with different scales. Increasing the receptive field is generally done by stacking multiple layers. In our proposed block we uses three successive max-pooling layers to anticipate the bigger receptive field of the next layers and gain more contextual information.

The proposed block is a therefore simplified Inception block based on the Spatial Pyramid Pooling Fusion module used in YOLO. The first step is the classical succession of Convolution 3x3 with batch normalization and ReLU activation. Then, max-pooling is applied three times in a cascade manner to the feature maps. The four sets of feature maps are then concatenated to be processed by a depth-wise separable convolutional layer before the skip connection. In this manner, it is less expensive than Inception because the Max-Pooling layers do not use parameters and because they are applied depth-wise. Figure 2 shows a precise illustration of the proposed block.

Table 1: Characteristics of the datasets.

Dataset	Image size	Training samples	Validation samples	Classes
MNIST	28x28x1	60,000	10,000	10
Fashion MNIST	28x28x1	60,000	10,000	10
CIFAR-10	32x32x3	50,000	10,000	10
CIFAR-100	32x32x3	50,000	10,000	100
Tiny ImageNet	64x64x3	100,000	10,000	200
ImageNette	160x160x3	9,469	3,925	10
ImageWoof	160x160x3	9,025	3,929	10

3 BENCHMARK DESCRIPTION

To compare the performance of the different blocks, we chose to train the models using different datasets from the literature. Indeed, one of the objectives of this work is to study the correlation between the model accuracy on small benchmarks with low resolution and the accuracy of the same model with higher-resolution data.

We have selected seven datasets created for model benchmarking: MNIST (Lecun et al., 1998), Fashion MNIST (Xiao et al., 2017), CIFAR-10 and CIFAR-100 (Krizhevsky,), Tiny ImageNet (Tavanaei, 2020; Tin,), ImageNette and ImageWoof (Howard, 2020). The details of each dataset are shown in Table 1.

Widely known, the two MNIST datasets are used in our experiments as a baseline to check the implementations (absence of bugs). Indeed, high accuracy levels superior to 90% are expected for both datasets, as the complexity of the datasets offers almost no challenge to the blocks.

Similarly, we use CIFAR datasets because they are the smallest color image datasets available. MNIST and CIFAR are standard for the early development of new models but shall not be considered a reference for real applications.

The Tiny ImageNet, with 64x64 pixels images, was selected as an intermediate step between low-resolution and higher-resolution data found in real applications. Indeed, this resolution shall offer sufficient information to start favoring more recent blocks.

Finally, for higher resolution datasets, we used the 160p versions of ImageNette and ImageWoof (Howard, 2020). These datasets comprise ten classes selected from the ImageNet-1k benchmark (Deng et al., 2009). Results obtained with these datasets are more valuable to our analysis because they represent real images.

3.1 Benchmark Setting

Our objective is not to achieve the best accuracy for each model but to compare their performances similarly. For this reason, we used a simple VGG-like

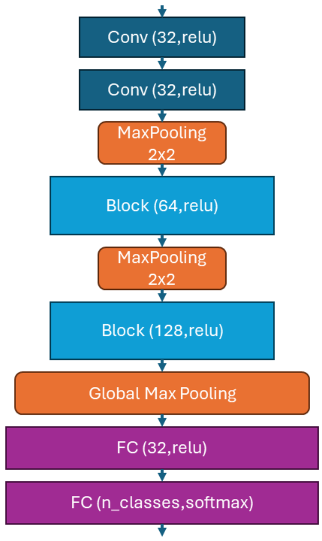

Figure 3: VGG-like architecture used in the experiments.

architecture, as shown in Figure 3. This architecture is then adapted by replacing the convolutional layers (blocks represented in blue in Figure 3) with other blocks proposed in the literature. In this manner, we can compare different blocks in a fast way. Another advantage of this strategy is that it enables quick prototyping of new models on edge devices.

The training parameters are similar for all variants, with a **maximum of 500 epochs** and an **early stopping** after 20 epochs of stagnation of validation loss. The classical **cross entropy loss** is used as the objective function, with the **Adam optimizer** (Kingma and Ba, 2017). Also, **accuracy** is used as the primary metric because every class has the same number of samples.

4 RESULTS

4.1 MNIST Datasets

Table 2 shows the training and validation accuracy for each model. Both C2F and the proposed pooling block reach the best accuracy of 99.6% on the original MNIST dataset. However, this simple test is insignificant because every other model also reaches high accuracy. The differences are slightly wider on the Fashion MNIST dataset, with accuracies going from 89.3% for the MobileNet V1 to 92.7% for the Inception V1 block.

Table 2: Results on the MNIST datasets.

Block	**MNIST**		**Fashion MNIST**	
Split	**Train**	**Val**	**Train**	**Val**
VGG	99,7	99,2	96,5	92,2
MobileNet V1	99,9	99,1	92,6	89,3
MobileNet V2	99,8	99,1	94,8	91,6
MobileNet V3	99,9	99,3	94,5	91,6
ResNet V1	99,9	99,4	94,8	91,3
Inception V1	99,9	99,4	96,4	**92,7**
SqueezeNet	99,8	99,1	94,7	92
GhostNet	99,5	98,4	94,2	91,1
CGNet	1	99,4	97,3	91,9
C2F	1	**99,6**	94,2	90,1
Pooling block	1	**99,6**	94,5	91,4

4.2 CIFAR Datasets

The subsequent results shown in Table 3 concern the CIFAR datasets. These tests bring more interesting insights as the performances among different block models are much more evident despite the relatively similar image size to MNIST and Fashion MNIST. From our experiments, the Inception V1 module performs best on both datasets with 74.9% and 38.8%, while our proposed block only reaches 46.5% on CIFAR-10 and 15.7% on CIFAR-100. Our interpretation is that a pooling-based block will perform poorly on tiny images (32x32) because the loss of information will be too significant. This is also true for the C2F block because it was designed for object detection in high-resolution images.

We can also make two observations. First, an original VGG block still reaches good accuracy for both MNIST and CIFAR datasets. Secondly, our pooling module was not affected by the low resolution of the MNIST datasets. This is easily comprehensible because both MNIST datasets contain well-defined segmented shapes with a black background, while CIFAR datasets use complex objects at 32x32 size. Hence, the difficulty gap between the MNIST and CIFAR datasets is wide: almost any model can reach good accuracy on the handwritten digits classification task, but classifying objects at such low resolution is, by definition, ambiguous.

4.3 Tiny ImageNet

The performances for the Tiny ImageNet dataset are shown in Table 4. Those images are still low-resolution, but their size is four times bigger than CIFAR, with 64x64 pixels. In this context, our Pooling block reaches the best accuracy with 32% compared to 28.3% obtained by MobileNet V3 or C2F. This result indicates that the image size reaches a sufficient

Table 3: Results on CIFAR datasets.

Block	CIFAR-10		CIFAR-100	
Split	**Train**	**Val**	**Train**	**Val**
VGG	84	73.3	50.3	37.2
MobileNet V1	72.3	66.1	38.4	31.6
MobileNet V2	82.2	71.3	52.4	38.2
MobileNet V3	80.1	71.7	46.9	36.3
ResNet V1	84.3	74	50.8	36.6
Inception V1	83.7	**74.9**	50.2	**38.8**
SqueezeNet	78.1	70.5	46.9	36.8
GhostNet	76.4	69.4	45.4	38.2
CGNet	90.9	72	43.2	34.9
C2F	35.8	35.5	11.2	10.8
Pooling block	48	46.5	16.2	15.7

Table 4: Results on average-size images (Tiny ImageNet).

Split	Train	Val
VGG	28,7	23
MobileNet V1	23,8	20,1
MobileNet V2	34,1	27,9
MobileNet V3	35,9	28,3
ResNet V1	33,8	25
Inception V1	25,7	18,3
SqueezeNet	22,5	18,7
GhostNet	29,2	24,6
CGNet	36,1	26,1
C2F	42,7	28,3
Pooling block	41,6	**32**

size to be explored by our Pooling block.

Please note that images like MNIST, CIFAR, or Tiny ImageNet do not represent actual images found in real applications. We recommend the usage of those datasets for specific Tiny-ML applications (for example, face expression detection from 48x48p images (Shao and Cheng, 2021)).

4.4 ImageNette and ImageWoof

Finally, this section compares the blocks with larger images obtained from ImageNette and ImageWoof datasets. These two datasets are less commonly used than MNIST and CIFAR because they were proposed in 2019 by FastAI without a research publication. Nonetheless, we argue that these datasets are more representative for benchmarking on high-resolution images (we have selected the 160p version for fast training, but there is also the 320p and full size available). Both datasets use classes from the ImageNet-1k benchmark, aiming to propose challenging classification benchmarks. Indeed, ImageWoof is the most difficult because it only uses classes of dog breeds.

Table 5: Results on larger images (ImageNette, ImageWoof).

Block	ImageNette		ImageWoof	
Split	**Train**	**Val**	**Train**	**Val**
VGG	86	70,5	58,8	45,8
MobileNet V1	88,6	71,4	65,2	48,6
MobileNet V2	82,7	71,1	67	50,6
MobileNet V3	79,7	64,8	54,6	39,4
ResNet V1	85	71,6	70,9	49
Inception V1	88,8	75,6	75,4	61,5
SqueezeNet	82,6	69,3	51	42
GhostNet	79,3	68,9	58,9	47,1
CGNet	93,4	73	67,9	49,9
C2F	90,8	73,2	92,8	63,7
Pooling block	92,5	**78,5**	95,1	**66,4**

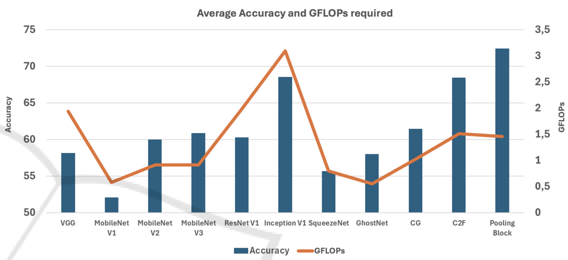

Figure 4: Accuracy and computation needs for each model.

In the experiments summarized in Table 5, our pooling block obtained the higher accuracy, with 78.5% for ImageNette and 66.4% for ImageWoof. Inception V1 is the second best on ImageNette with 75.6% and C2F on ImageWoof with 63.7%.

4.5 Image Classification for the Edge

The previous sections demonstrate the interest in recent blocks, especially with larger images. However, a higher accuracy is not enough when optimizing for the Edge. Indeed, resource-constrained environments such as those found on the Edge need to balance accuracy, computing power, and memory footprint.

Figure 4 compares the achieved accuracy and the required computing power in GFLOPs (billions of floating-point operations) when classifying a 160p image. For example, the Inception V1 model needs 3 GFLOPs for one inference, while our pooling module only needs half of this with 1.46 GFLOPs (close to the 1.5 GFLOPs of C2F).

A small correlation of 0.31R² was found between the accuracy of the models and the computation requirement. This is expected because the architecture is as important as the number of filters. For example, GhostNet reaches a better average accuracy of 58% compared to MobileNet V1, which achieves only

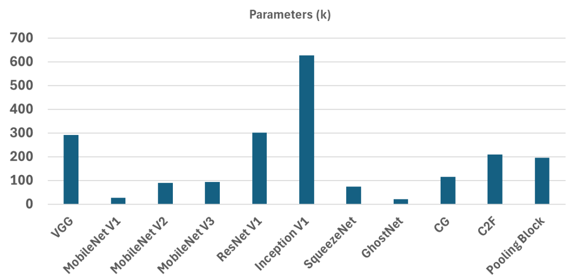

Figure 5: Number of parameters for each model.

52%. Nonetheless, both models need a similar computation of about 0.6 GFLOPs.

Another element to consider is memory consumption. As the number of parameters increases, the model grows in complexity, requiring more memory to house intermediate calculations and activation states. With the notable exception of Inception V1, most recent modules have a limited memory footprint, as shown in Figure 5.

Compared to Inception V1, which obtained the second better accuracy on the classification of 160p images, our pooling block shows several advantages. First, it only needs a third of the parameters, with 197k compared to over 600k for Inception V1. This is the advantage of using parameter-free layers like Max-Pooling instead of convolutional layers. Another advantage is that our module also acts as a bottleneck because the number of features is squeezed back after the concatenation, thus avoiding increasing parameters in the next convolutional layer.

5 CONCLUSIONS

In this work, we compare different convolutional blocks used in recent image classification models and a new Pooling-based Inception block. Systematic benchmarking was performed on seven datasets (MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, Tiny-ImageNet, ImageNette, and ImageWoof).

In a controlled setting with a VGG-like architecture, our proposed block is more accurate on medium-size images (Tiny-ImageNet, ImageNette, and ImageWoof) than other classical blocks such as ResNet or Inception V1. We also found that the MNIST and CIFAR datasets are not representative enough to benchmark models designed for high-resolution images. Furthermore, our proposed block model needs less computation and memory than Inception V1.

These results reinforce the interest in comparing the models' accuracy and other parameters that may impact the usage of the models, especially in the case of Edge or IoT devices. Choosing an efficient model also brings side-benefits for Edge-AI applications, such as reducing the energy requirements and freeing processing cores for accessory tasks (for example, interfacing with a GPS for precise location).

We are aware that the chosen VGG-like architecture limits our current results. Our immediate future works include, therefore: (1) the creation of a new model based on our Pooling block to be trained on the ImageNet-1K dataset; (2) the optimization of the Pooling block with reparametrization techniques (Vasu et al., 2023; Ding et al., 2021); and (3) the deployment of the future model on edge applications.

ACKNOWLEDGEMENTS

This work was supported by Chips Joint Undertaking (Chips JU) in EdgeAI "Edge AI Technologies for Optimised Performance Embedded Processing" project, grant agreement No 101097300.

We also thank the ROMEO Computing Center[1] of Université de Reims Champagne-Ardenne, whose Nvidia DGX-1 server allowed us to accelerate the training steps and compare several model approaches.

REFERENCES

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05587 [cs].

De Lucia, G., Lapegna, M., and Romano, D. (2022). Towards explainable ai for hyperspectral image classification in edge computing environments. *Computers and Electrical Engineering*, 103:108381.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, page 248–255.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., and Sun, J. (2021). Repvgg: Making vgg-style convnets great again. arXiv:2101.03697 [cs].

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., and Xu, C. (2020). Ghostnet: More features from cheap operations. arXiv:1911.11907 [cs].

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H. (2019). Searching for mobilenetv3. arXiv:1905.02244.

[1]https://romeo.univ-reims.fr

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861 [cs].

Howard, J. (2020). Imagenette and imagewood. Online: https://github.com/fastai/imagenette/.

Hu, J., Shen, L., Albanie, S., Sun, G., and Wu, E. (2019). Squeeze-and-excitation networks. arXiv:1709.01507.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2018). Densely connected convolutional networks. arXiv:1608.06993 [cs].

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size. arXiv:1602.07360 [cs].

Jastrzebski, S., Arpit, D., Ballas, N., Verma, V., Che, T., and Bengio, Y. (2018). Residual connections encourage iterative inference.

Jiang, P., Ergu, D., Liu, F., Cai, Y., and Ma, B. (2022). A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics yolov8. https://github.com/ultralytics/ultralytics.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. arXiv:1412.6980 [cs].

Krizhevsky, A. Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Mohimont, L. (2023). Deep learning for post-harvest grape diseases detection. In *Workshop Proceedings of the 19th International Conference on Intelligent Environments (IE2023)*, volume 32, page 157. IOS Press.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2019). Mobilenetv2: Inverted residuals and linear bottlenecks. arXiv:1801.04381 [cs].

Shao, J. and Cheng, Q. (2021). E-fcnn for tiny facial expression recognition. *Applied Intelligence*, 51(1):549–559.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 [cs].

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. arXiv:1409.4842 [cs].

Tavanaei, A. (2020). Embedded encoder-decoder in convolutional networks towards explainable ai. arXiv:2007.06712 [cs].

Terven, J. and Cordova-Esparza, D. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolonas. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716. arXiv:2304.00501 [cs].

Vasu, P. K. A., Gabriel, J., Zhu, J., Tuzel, O., and Ranjan, A. (2023). Mobileone: An improved one millisecond mobile backbone. arXiv:2206.04040 [cs].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., and Xiao, B. (2020). Deep high-resolution representation learning for visual recognition. arXiv:1908.07919 [cs].

Wu, T., Tang, S., Zhang, R., and Zhang, Y. (2019). Cgnet: A light-weight context guided network for semantic segmentation. arXiv:1811.08201 [cs].

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, page 818–833, Cham. Springer International Publishing.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. (arXiv:1612.01105). arXiv:1612.01105 [cs].