DOI: 10.32347/2412-9933.2024.60.221-229
UDC 004.8

Hozak Yaroslav
Postgraduate student of information systems and technologies,
https://orcid.org/0009-0008-2963-7204
Taras Shevchenko national university of Kyiv, Kyiv, Ukraine
Paliy Sergiy
PhD (Eng.), associate professor, Department of information systems and technologies,
https://orcid.org/0000-0001-9742-1116
Taras Shevchenko national university of Kyiv, Kyiv, Ukraine

MODERN SMALL NETWORKS FOR IMAGE CLASSIFICATION. FEATURE ANALYSIS

Abstract. *With the increasing demand for deploying deep learning models on resource-constrained devices, such as smartphones, IoT sensors, and edge computing platforms, the need for efficient convolutional neural networks (CNNs) has become paramount. This paper offers a comprehensive review of several state-of-the-art lightweight CNN architectures designed to address these challenges by reducing computational complexity and memory usage, while maintaining competitive performance in image classification tasks. Key architectures reviewed include MobileNets, ShuffleNet, DiceNet, and ESPNet, each of which employs distinct strategies to optimize network efficiency. MobileNets introduce the concept of depthwise separable convolutions, which decompose the standard convolution operation into a depthwise convolution and a point-wise convolution (1x1). This drastically reduces the number of parameters and computations compared to traditional convolutions. ShuffleNet, on the other hand, leverages group convolutions and channel shuffling to enhance efficiency, allowing feature maps to be split and recombined, which reduces computational cost without significantly compromising accuracy. DiceNet builds upon these concepts by introducing multi-branch architecture with different dilation rates to capture features at multiple scales, enhancing both accuracy and efficiency in low-resource environments. ESPNet employs efficient spatial pyramidal structures, along with point-wise convolutions, to handle diverse spatial features at different scales while being highly computationally efficient. Despite these advancements, a common bottleneck across these architectures is the reliance on point-wise (1x1) convolutions, which, while more efficient than standard convolutions, still contribute significantly to the overall computational cost, particularly in deeper layers of the network. Furthermore, filter sizes are often optimized for performance in a cloud-based setting but may not be ideal for edge environments where computational and energy efficiency are crucial. We see the potential in changing filter sizes in some layers to 2x2 which is the smallest possible filter for spatial information extraction. Also it worth paying attention to the way the information is spread across channels as well as how the channels number is formed by replacing 1x1 convolution with a generic but yet predictable mathematical operation.*

Keywords: convolutional neural networks; image classification; computer vision; IoT

Introduction

In recent years, neural networks, particularly convolutional neural networks (CNNs), have revolutionized image recognition by enabling systems to process and classify images with remarkable accuracy. The deployment of such systems on mobile and edge devices has become increasingly significant, driven by the growth of mobile applications, autonomous systems, and the Internet of Things (IoT). This shift allows for real-time processing and decision-making at the edge of the network, reducing the need for cloud-based computation and mitigating latency and privacy concerns.

Neural networks, specifically deep learning models like CNNs, are designed to automatically learn features from input images, enabling them to identify patterns such as shapes, textures, and objects. CNNs, in particular, have gained popularity in image recognition due to their ability to handle spatial data efficiently.

Deep learning automates the learning process, producing models capable of superior generalization across different tasks. These models have become integral in various applications, including facial recognition, augmented reality, autonomous vehicles [8], video surveillance [6; 7] and medical diagnostics.

Standard CNNs often have high latency due to large model sizes and extensive computations. Optimizing

models for inference speed, memory efficiency, and reduced Floating Point Operations Per Second (FLOPs) ensures they can process data fast enough for these real-time applications.

Large neural networks not only consume more resources but also introduce higher latency in image recognition tasks. For real-time applications any delay in processing could be detrimental. Therefore, it is crucial to design models that reduce latency without sacrificing accuracy.

Many real-world applications of deep learning, such as mobile apps, embedded systems, and IoT devices, involve hardware with limited computational resources, memory, and power. Standard deep learning models like VGG [2] or ResNet [4] are computationally heavy and unsuitable for such environments without significant optimization. Also, despite optimizations, neural networks still demand considerable computational resources and running them on mobile devices can drain battery life quickly. Edge devices in remote or mobile applications are particularly sensitive to power constraints. This has led to ongoing research into lightweight architectures such as MobileNets [11 – 13], ShuffleNet [14] and the recently proposed DiCENet [18]. These networks make it feasible to run high-performing models on low-power devices without sacrificing much accuracy.

While progress has been made in deploying neural networks for image recognition on mobile and edge devices, several limitations remain:

Hardware Limitations: Although mobile processors and edge computing hardware are becoming more powerful, they still lag behind server-grade GPUs and TPUs in terms of computational capability. Future developments in specialized hardware, such as neuromorphic chips or low-power AI accelerators, are expected to further enhance on-device neural network performance.

Model Optimization Trade-offs: Reducing the size and complexity of models often results in a trade-off between performance and accuracy. While techniques like model pruning and quantization help in reducing model size, they may also lead to a loss in recognition accuracy, especially in complex environments.

There are also other limitations such as continuous learning on edge devices without frequent updates from the cloud or interoperability and standards that create challenges in deploying neural networks across different devices and platforms.

The purpose of the article

The purpose of this article is to provide comparative analysis of existing efficient CNNs and their features, development of further improvement recommendations and setting the task of network accuracy improvement with respect to its computational cost.

Efficient convolutional neural networks

The main building block of a CNN is a convolution layer which is responsible for spatial feature extraction. A lot of research has been done around the convolution layer itself [1] as well as the entire blocks which can incorporate one or more convolutional layers [2 – 4]. Other investigations are related to alternative activation functions, regularization or parameter optimization [5].

We discuss the following CNNs: MobileNet [11], MobileNet V2 [12] and MobileNet V3 [13], ShuffleNet [14], ShuffleNet V2 [15], ESPNetv1 [16] and DiCENet [18] and SqueezeNe t[19] and of course we will refer to AlexNet [1] as a baseline for accuracy.

AlexNet. The network that started it all. Since the innovation of LeNet network in 1980s [20] convolutional neural networks haven't been evolving until AlexNet emerged in 2012 taking the 1st place in ImageNet competition. This model gives the top-5 accuracy of 83.0% and top-1 accuracy of 62.5% as per their reports.

This network consists of 8 layers with 5 first layers being convolutional and 3 last layers being fully connected. Due to computational resources limitation of that time the original network had specific architecture which divides some convolutional and some fully connected layers into two separate ones to train them on 2 different GPUs. Thanks to modern processing libraries it is possible to combine back each separated layer and present it as a single solid one.

Some other important features that AlexNet proposes are the use of ReLU [21] activation function as a nonlinearity function as well as local response normalization and overlapping pooling.

It worth mentioning that such a big network tends to overfit even using such big datasets as ImageNet so the authors also used the dropout [22] of 0.5 during training which positively affected training results. A neuron is removed from the neural network during dropout with specified probability. A neuron that is dropped does not make any contribution to either forward or backward propagation. So every input is processed by a separate neural network architecture but all architectures share the same weights. The acquired weight parameters are therefore more reliable and less prone to overfitting.

The model has 61.1M parameters which was (and still is) considered to be quite a substantial number.

SqueezeNet. The model was proposed in 2016 by Forrest N. Iandola et al with the primary goal of reducing the network size while preserving accuracy. The authors chose the AlexNet as a baseline for inference accuracy. The authors list the following advantages of smaller networks:

1. Less communication between servers during distributed training;
2. Less bandwidth is required for uploading new model onto the devices over-the-air;

3. Smaller CNNs are more feasible to deploy on limited-resources devices such as IoT devices, mobile phones or other embedded devices.

The model's main aim is to reduce the size on a disk consumed by the network. Most popular networks make heavy use of 3x3 convolutional layers which are proved to be the most effective in deep neural networks. Though it is admitted that 1x1 convolutional layers have 9 times fewer learnable parameters, however they provide drastically less spatial information. Another conceptual idea is to reduce the number of input channels to the remaining 3x3 filters. A *squeeze layer* is proposed to decrease the number of input channels for 3x3 convolutional filers.

A module called *Fire module* is proposed. The module represents design decision at a microarchitecture level which defines how a basic network building block should look like. The term of *microarchitecture* was also introduced by Forrest N. Iandola et al in this document and defines microarchitecture as architecture of high-level building block or a module comprised of multiple convolutional layers with a specific fixed organization.

The fire module consists of a small 1x1 squeeze layer which reduces the number of input channels to the next *expansion layer*, which in turn consists of 1x1 and 3x3 filters. The expansion layer is called so because it expands back the number of channels after the squeeze layer. The amount of 1x1 filters in squeeze later as well as 1x1 and 3x3 filters in expansion layer is controlled by hyper parameters that effectively allows to modify the network architecture on module level.

The network consists of 1st convolutional layer, followed by 8 fire modules, 3 maxpool and 1 averagepool layers and ending with the final 1x1 convolutional layer fed into softmax unit for classification which results in 22 layers in total (since fire module consists of 2 layers). The use of 1x1 convolutional layer instead of fully connected one was inspired by the NiN[23] and is also known to reduce the number of parameters.

Important to note that authors of SqueezeNet proposed a more disciplined approach to searching the novel CNN architectures. The authors provided comparison for model efficiency based on squeeze ratio and 3x3 filters in expand layer ratio. The results showed that the best accuracy can be achieved with the squeeze ratio equals to 1 (which effectively means no squeeze) and 3x3 filters taking 100 of expand layer, however consuming more memory. Accuracy doesn't correlate linearly to squeeze/expand ratio changes which gives authors the way to propose the best accuracy/model size ratio.

MobileNet. MobileNet V1 was originally developed by Andrew G. Howard et al in 2017. [11] It leverages the performance of depthwise separable convolution inspired by Xception network [24] as many other modern small networks do.

The standard convolution operation has the effect of filtering features based on the convolutional kernels and combining features to produce a new representation. The filtering and combination steps can be split into two steps via the use of factorized convolutions called depthwise separable convolutions for substantial reduction in computational cost.

As stated above depthwise separable convolution consists of depthwise convolution and pointwise convolution. First a depthwise convolution is used by applying individual spatial filters to feature map channels and then pointwise convolution in a form of 1x1 convolution is applied to create a linear combination of depthwise convolution outputs as shown on Figure 3.

As mentioned in the paper standard convolutions have the computational cost of:

$$DK \cdot DK \cdot M \cdot N \cdot DF \cdot DF \quad (1)$$

And depthwise separable convolutions costs:

$$DK \cdot DK \cdot M \cdot DF \cdot DF + M \cdot N \cdot DF \cdot DF \quad (2)$$

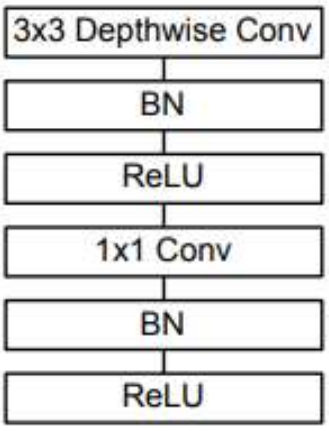
Using depthwise separable convolution we get a reduction in computation of:

$$1/N+1/(D\_k\text{^}2\ ) \quad (3)$$

Because MobileNet uses 3x3 filters each its depthwise separable convolutional layer gives approximately 9 times reduction in computation.

For a feature map of size 112x112x32 applying standard convolutional filter of size 3x3(x32) 32 times results in 3x3x32x32x112x112 = 115,605,504 computations.

The same can be achieved with depthwise separable convolution using only 3x3x32x112x112 + + 32x32x112x112 = 3,612,672 + 12,845,056 = = 16,457,728 computations which is 7.02 times better than full convolutional layer.



Picture – Depthwise separable convolutional layer [24]

MobileNet consists of 28 layers with only the 1st layer being fully convolutional and other layers being depthwise separable convolutional layers and the last fully connected layer preceding the softmax unit.

A model provides the following hyperparameters for tuning the network: width multiplier hyperparameter and resolution multiplier hyperparameter. The values of them are intended to change the accuracy and size of the model as per user needs. The width multiplier α essentially represents the number of channels on each layer compared to the original structure. Typical values for it are (0.25, 0.5, 0.75, 1). $\alpha = 1$ is the baseline MobileNet and $\alpha < 1$ are reduced MobileNets. Width multiplier has the effect of reducing computational cost and the number of parameters quadratically by roughly $\alpha 2$.

The second hyper-parameter to reduce the computational cost of a neural network is a resolution multiplier ρ. The parameter is applied to the very first layer in the network, essentially to the input image. This parameter reduces the size of the input image and internal representation of every subsequent layer is reduced by the same multiplier. Resolution multiplier also has the effect of reducing computational cost by $\rho 2$.

Results of the paper show that at similar computation and number of parameters, making MobileNets thinner is 3% better than making them shallower.

On ImageNet classification task MobileNet 1.0 has Top-1 accuracy of 70.6%, 4.2M parameters with 569M MAdds.

MobileNetV2. Introduced in 2019 MobileNetV2 [12] is the next iteration of MobileNet [11]. The basic model structure remains the same and also makes use of depthwise separable convolution, however, introduces inverted residual blocks. Inverted residual blocks create connections between thin layers as opposed to classical residual blocks that connect expansion layers.

The inverted residual blocks appear similar to residual block where each block contains an input followed by several bottlenecks then followed by expansion [25]. However, inspired by the intuition that the bottlenecks actually contain all the necessary information, while an expansion layer acts more like an implementation detail that accompanies a non-linear transformation of the tensor, the authors suggest using shortcuts directly between the bottlenecks. The shortcuts allow the network to learn better using the same principle as ResNet does, but the shortcuts connect narrow layers instead of wide ones which also saves computation.

A network block structure is defined by a 1x1 expansion convolution followed by depthwise convolutions and a 1x1 projection layer. The input and output are connected with a residual connection if and only if they have the same number of channels.

The architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottlenecks layers. Model uses ReLU6 as the non-linearity because of its robustness when used with low-precision computation [11]. The kernel size is always 3×3 as is standard for modern networks.

On ImageNet classification task MobileNetV2 1.0 has Top-1 accuracy of 72.0%, 3.4M parameters with 300M MAdds.

MobileNetV3. Presented in 2019 by authors from Google MobileNetV3 [13] became the latest iteration of MobileNets family.

Authors used Platform-Aware NAS for Block-wise Search approach which then followed by NetAdapt for Layer-wise Search. In this way authors automated the search for a better network structure.

MobileNetV3 uses lightweight attention modules [28] based on squeeze and excitation in the bottleneck structure. The module is placed after the depthwise filters in the expansion in order for attention to be applied on the largest representation. Layers are also upgraded with modified swish nonlinearities [26, 27]. Both squeeze and excitation as well as the swish nonlinearity use the sigmoid which can be inefficient to compute as well challenging to maintain accuracy in fixed point arithmetic, so the sigmoid is replaced with the hard sigmoid nonlinearity.

Also, the model contains redesigned computationally expensive layers. During development it was found out that the last convolutional layer and the first convolutional layer are more expensive than others. The last 1x1 convolutional layer is pretty computationally heavy but also is critical for prediction so it is impossible to remove it, however authors managed to move the layer past the average pooling layer. In this way the expensive 7x7 convolutional layer could be replaced with 1x1 spatial layer. For the first convolutional layer authors proposed to use h-swish implemented as piece-wise function nonlinearity instead of ReLU6 (h-swish is a little bit faster) and also to reduce the number of filters from 32 to 16, they proved that this reduction resulted in almost no loss in accuracy.

As a conclusion this network's main advantage compared to MobileNetV2 is use of h-swish nonlinearity combined with quantization. MobileNetV3 provides -Large and -Small models to cover variety of performance/accuracy requests.

On ImageNet classification task MobileNetV3-Large 1.0 has Top-1 accuracy of 75.2%, 5.4M parameters with 219M MAdds. MobileNetV3-Small has Top-1 accuracy of 67.4%, 2.5M parameters with 56M MAdds.

ShuffleNet. This architecture was introduced in 2017 by Xiangyu Zhang et al from Megvii Inc. [14] The autors note that popular large architectures such as Xception or ResNeXt become less efficient for small networks because of costly 1x1 convolutions. The

architecture uses pointwise group convolutions to reduce the complexity of 1x1 pointwise convolutions and to efficiently share information between channel groups it uses channel shuffle operation.

ShuffleNet employs group convolutions with the number of groups varying from 3 to 8. It is noted that sequentially using multiple group convolutions leads to significant information loss because every group convolution gets information from only a group of channels from previous layer rather than the whole layer. Straight sequential using of group convolutions without information sharing such as pointwise convolution leads to a group getting information from only a small part of previous layer. Authors propose to replace expensive 1x1 pointwise convolution for information sharing with cheap channel shuffle operation.

Channel shuffle operation takes GConv layer output with g groups each of which has n channels resulting in g x n channels and reshapes their dimension into (g, n). This matrix is then transposed and flattened back and this result is provided as an input to the next channel.

In addition, in ShuffleNet depthwise convolution only performs on bottleneck feature maps. Even though depthwise convolution usually has very low theoretical complexity, it may be difficult to efficiently implement on lowpower mobile devices, which may result from a worse computation/memory access ratio compared with other dense operations.

ShuffleNet architecture contains 50 layers while MobileNet only has 28 layers. Making the model shallower by removing half of blocks thus reducing the number of layers to 26 slightly decreases accuracy but the best performance can be achieved by using 50 layers.

On ImageNet classification task ShuffleNet 1.0 has Top-1 accuracy of 67.6%, at 140M FLOPs.

We are sure it is important to note that group convolutions with channel shuffling have their drawback in that the number of groups is limited to relatively small number which makes groups quite big in deeper layers when the number of channels increases and as a result group convolution becomes heavy operation.

ShuffleNet V2. ShuffleNet V2 is an evaluation of ShuffleNet V1 [14] and was proposed in 2018 by Ningning Ma at al. [15]

The paper mainly assesses theoretical metrics accuracy of GFLOPs compared to real performance on different devices. One of things to consider is memory access cost (MAC) can significantly influence real performance due to limited high speed cache on device. In general the following guidelines are discussed and proposed by the research:

• (G1) Equal channel width minimizes memory access cost (MAC).

• (G2) Excessive group convolution increases MAC.

• (G3) Network fragmentation reduces degree of parallelism.

• (G4) Element-wise operations are non-negligible.

ShuffleNet V2 design uses ShuffleNet as the base and introduces changes to comply with the stated above guidelines.

ShuffleNet V2 unit is based on unit from [14] but contains the following differences. An introduced channel split operation splits incoming channels c into two branches with c – c` and c` channels, respectively. One branch remains untouched while the other branch applies sequentially 1x1 convolution, 3x3 depth-wise convolution and 1x1 convolution with each convolution having the same number of input and output channels to satisfy G1. The network uses c` = c/2 in its original version.

After convolution, the two branches are concatenated so the number of channels stays the same (G1). The same “channel shuffle” operation as in [14] is then used to enable information exchange between the two branches.

In each block c` channels out of c channels are skipped and are forwarded almost directly to the next block. It can be viewed as feature reuse. It is clear that the connections between the adjacent layers are stronger than the others. This implies that the dense connection between all layers could introduce redundancy.

Complete network architecture is similar to [14] with respective changes to individual blocks. Apart from that an additional 1x1 convolutional layer is added right before the global average pooling to allow for feature mix up. That makes the network to contain 51 layers in total.

DiCENet. Introduced in 2020 by Sachin Mehta et al DiCENet [18] introduces concept of dimension-wise convolutions. It describes the application of light-weight convolutional filtering across each dimension of the input tensor with further application of dimension-wise fusion to combine dimension-wise representations.

DiCE unit applies standard depth-wise convolution in all 3 dimensions (depth, width and height). A result of this operation contains 3x more convolutional results because of applying the kernel in 3 directions. This operation is called dimension-wise convolution (DimConv).

DimConv has three branches, one branch per dimension. The outputs of these independent branches are concatenated along the depth dimension, such that the first spatial plane of YD, YW, and YH are put together and so on, to produce the output.

The ability to encode local spatial and channel-wise information from all dimensions using DimConv enables the DiCE unit to use dimension-wise fusion (DimFuse) instead of computationally expensive point-wise convolutions. DimFuse consists of local fusion and global fusion. YDim ∈ R3D×H×W concatenates spatial planes along depth dimension from YD, YW, and YH.

YDim can be viewed as a tensor with D groups, each group with three spatial planes. Thus, local fusion part of DimFuse operation uses a group point-wise convolutional layer to combine dimension-wise information contained in YDim effectively encoding special dimensions.

To encode the global information the global fusion part learns special and channel-wise representations independently and then uses element-wise multiplication to propagate channel-wise encodings to special ones. To encode special representation, it uses D dept-wise convolutional kernels kS ∈ R1×n×n, where n is the size of a filter. To encode channel-wise representation inspired by Squeeze-Excitation (SE) unit [29] the fusion module squeezes special dimensions, uses 2 fully connected layers with non-linearity in between. Like the SE unit, spatial representations YG are then scaled using these channel-wise representations to produce final output Y.

Unlike in other popular networks [11, 12, 14, 15] dimension-wise convolution allows for simultaneous encoding in 3 dimensions. The computational cost of DimFuse is HWD(3 + n2 + D). Effectively, DimFuse reduces the computational cost of pointwise convolutions by a factor of (3D)/(3+n^2+D) . DimFuse uses n = 3, so the computational cost is approximately 3 times smaller than that of the point-wise convolution.

For DiCENet the overall network architecture doesn't bring anything new, and the DiCE unit efficiency is evaluated on MobileNetV2 and ShuffleNetV2 architectures.

DiCE module brings new point of view to convolutional layers by applying depth-wise convolution in all dimensions and in this way gathering spatial information in both spatial and channel-wise direction.

On ImageNet classification task DiCENet has Top-1 accuracy of 75.7%, 5.1M parameters with 297M MAdds.

ESPNet. ESPNet [16] is based on efficient spatial pyramid (ESP) modules, a factorized form of convolutions that decompose a standard convolution into a point-wise convolution and a spatial pyramid of dilated convolutions [32].

ESP module consists of a point-wise convolution that projects high-dimensional feature maps onto a low-dimensional space (effectively reducing the number of channels from M to N/K) and followed by special pyramid of dilated convolutions, where M is the number of input channels, N – number of output channels and K is a divider, chosen deliberately. ESPNet uses K = 4. A pyramid contains K dilated convolutional kernels of size n × n, and for each kernel the dilation rate is different and is 2k−1, k = {1, ..., K}. This factorization drastically reduces the number of parameters, and the memory required by the ESP module, while preserving a large effective receptive field ((n−1)2K−1 +1)2. This pyramidal convolutional operation is called a spatial pyramid of dilated convolutions, because each dilated convolutional kernel learns weights with different receptive fields and so resembles a spatial pyramid. The outputs of the K parallel dilated convolutional kernels are concatenated to produce an N-dimensional output feature map.

The ESP module has (NM + (Nn)2)/K parameters and its effective receptive field is ((n−1)2K−1 +1)2 . Compared to the n2NM parameters of the standard convolution, factorizing it reduces the number of parameters by a factor of (n^2 MK)/(M+n^2 N) , while increasing the effective receptive field by ∼ (2K−1)2.

While dilated convolutions give the ESP module a large receptive field the resulting output contains gridding artifacts which are caused by the fact that dilated convolutions have lots of 0 pixels. To overcome it the ESP module first adds the feature maps obtained by dilated kernels with different dilation rate and only then concatenates them. The addition operation allows for 0 pixels to be substituted by information pixels from feature maps with different dilation rates. This operation is called Hierarchical Feature Fusion (HFF).

Improvement recommendations

Some key notes that should be taken are the channel doubling when down sampling. It is important to ensure the number of channels in each layer is devisable by 32 (most common architectures have their first layers with exactly 32 channels and double this number during each down sampling). The reason for it is that GPU kernels run threads in groups of 32, which means that different number of channels leads to inefficient use of GPU threads during training. However, it is reasonable to change the number of channels in order to reduce the inference time for low-resource devices, providing it doesn't hurt the accuracy.

Depth-wise separable convolution is very effective, giving for 3x3 filters an approximate speed up to 9 times as compared to standard convolution. Depth-wise separable convolutions is de-facto a standard when building modern network architectures.

As was pointed out in [14] point-wise convolution which is responsible for encoding global information may take up to 90% of computational cost. When using group convolutions in stacked layers of same dimension it is more effective to substitute pointwise convolution by concatenating the results of group convolutions and pass the information further by shuffling channels. Channel shuffle is much more efficient thanks to its simplicity and doesn't require parameters which have to be trained. Memory access cost (MAC) affects real efficiency

compared to theoretically calculated computation cost. It makes sense to limit MAC in possible ways. Equal channel width minimizes MAC and group convolutions should be used with caution as they tend to increase MAC.

Another point of view to channel-wise encoding is proposed by [18] in applying depth-wise filtering across all dimensions. This approach allows to perform local information encoding channel-wise using light-weight convolution. However, to get global information it is still required to perform some sort of operations to convolve across resulting channels and it is done by dimension-wise fusion which leverages group convolutions followed by global pooling and some fully connected layers. We see potential in further improving DiCE unit by changing the way to globally encode channel-wise information.

The feature re-sampling methods re-sample the convolutional feature maps of different scales to a fixed scale output using different pooling rates [30, 31] and kernel sizes for efficient classification. Feature re-sampling is computationally expensive and is performed just before the classification layer to learn scale-invariant representations. This is very important as re-sampling allows for a network to process images of different sizes without need for explicit retraining of other operations to feed an image into a network.

Because point-wise convolution aims to encode cross-channel information we think it is possible to substitute point-wise convolution with proper channel shuffling among several layers with carefully selected numbers of groups in each layer and the result of the last layer can be aggregated by simple mathematical operation (similar to channel shuffle but rather aggregating information) such as max pooling operation. Such an approach may significantly reduce the number of point-wise convolutions providing the entire architecture of a network is somewhat limited by the size of groups in group convolutions and the number of sequentially applied group convolution layers. The idea is that instead of point-wise convolution in each layer the network can share information cross-channel using only channel shuffle operations and the effect must be better when applying several stacked group convolutional layers with shuffled channels in between. And the average pooling or other proposed operation at the end of such block performs the operation of convolving resulting tensor into a single feature map. The key question here is forming the operation to uniquely, yet in a predictable way create multiple feature maps.

Conclusions

We analyzed many CNNs with their unique features that significantly reduce model size and computational cost at a cost of small to no accuracy loss. Yet, it is clear that even the most efficient models can be hard to run in real time on low-power resources such as edge devices.

We propose further investigations in adjusting filter sizes for better performance and in searching for an operation that can replace 1x1 point-wise convolutions efficiently.

References

1. Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097-1105.

2. Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.*

3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-9).

4. He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778).

5. Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K. & Ghayvat, H. (2021). CNN variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, 10(20), 2470. https://doi.org/10.3390/electronics10202470

6. Patel, C. I., Patel, R. & Patel, P. (2011, July). Goal detection from unsupervised video surveillance. In *International Conference on Advances in Computing and Information Technology* (pp. 76-88). Berlin, Heidelberg: Springer Berlin Heidelberg.

7. Patel, R. & Patel, C. I. (2013). Robust face recognition using distance matrice. *International Journal of Computer and Electrical Engineering*, 5(4), 401-404.

8. Bosamiya, D. & Fuletra, J. D. (2013). A survey on drivers drowsiness detection techniques. *International Journal of Recent Innovations and Trends in Computing and Communication*, 1, 816-819.

9. Tan, M. & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning* (Vol. 97, pp. 6105-6114). PMLR.

10. Tan, M. & Le, Q. V. (2021). EfficientNetV2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*.

11. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

12. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4510-4520).

13. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. & Le, Q. V. (2019). Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 1314-1324).

14. Zhang, X., Zhou, X., Lin, M. & Sun, J. (2018). ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6848-6856).

15. Ma, N., Zhang, X., Zheng, H. T. & Sun, J. (2018). ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *Proceedings of the European Conference on Computer Vision* (pp. 116-131). Springer.

16. Mehta, S., Rastegari, M., Caspi, A., Shapiro, L. & Hajishirzi, H. (2018). ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. *arXiv preprint arXiv:1803.06815*.

17. Mehta, S., Rastegari, M., Caspi, A., Shapiro, L. & Hajishirzi, H. (2019). ESPNetv2: A light-weight, power efficient, and general purpose convolutional neural network. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9190-9198).

18. Ma, Y., Shao, Y., Wu, X. & Sun, Y. (2020). DiCENet: Dimension-wise convolutions for efficient networks. *arXiv preprint arXiv:2002.10902*.

19. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint arXiv:1602.07360*.

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

21. Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning* (pp. 807-814).

22. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

23. Lin, M., Chen, Q, & Yan, S. (2014). Network in network. *arXiv preprint arXiv:1312.4400*.

24. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1251-1258).

25. He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

26. Ramachandran, P., Zoph, B. & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

27. Elfwing, S., Uchibe, E. & Doya, K. (2017). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *arXiv preprint arXiv:1702.03118*.

28. Tan, M., Chen, B., Pang, R., Vasudevan, V. & Le, Q. V. (2019). MnasNet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*.

29. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3), 211-252.

30. Zhao, H., Shi, J., Qi, X., Wang, X. & Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2881-2890).

31. He, K., Zhang, X., Ren, S. & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proceedings of the European Conference on Computer Vision* (pp. 346-361). Springer.

32. Holschneider, M., Kronland-Martinet, R., Morlet, J. & Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets* (pp. 286–297). Springer.

33. Biloshchytskyi, A., Dikhtiarenko, O. & Paliy, S. (2015). Searching for partial duplicate images in scientific works. *Management of Development of Complex Systems*, 21, 149 – 155.

Received 02.11.2024

Гозак Ярослав Дмитрович
Аспірант кафедри інформаційних систем та технологій,
https://orcid.org/0009-0008-2963-7204
Київський національний університет імені Тараса Шевченка, Київ
Палій Сергій Володимирович
Кандидат технічних наук, доцент, доцент кафедри інформаційних систем та технологій,
https://orcid.org/0000-0001-9742-1116
Київський національний університет імені Тараса Шевченка, Київ

СУЧАСНІ МАЛІ МЕРЕЖІ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ. АНАЛІЗ ОСОБЛИВОСТЕЙ

Анотація. *Зі зростанням попиту на використання моделей глибокого навчання на пристроях з обмеженими ресурсами, таких як смартфони, датчики IoT і периферійні обчислювальні платформи, потреба в ефективних згорткових нейронних мережах (ЗНМ) стала першорядною. У статті запропоновано вичерпний огляд кількох найсучасніших полегшених архітектур ЗНМ, розроблених для вирішення цих проблем шляхом зменшення обчислювальної складності та використання пам'яті, зберігаючи конкурентоспроможність у задачах класифікації зображень. Переглянуті ключові архітектури включають MobileNets, ShuffleNet, DiceNet і ESPNet, кожна з яких використовує різні стратегії для оптимізації ефективності мережі. MobileNets представляє концепцію згорток, що розділяються по глибині, які розкладають стандартну операцію згортки на згортку по глибині та згортку по точках (1x1). Це суттєво зменшує кількість параметрів і обчислень порівняно з традиційними згортками. З іншого боку, ShuffleNet використовує групові згортки і перетасування каналів для підвищення ефективності, уможливлюючи розділяти та рекомбінувати карти ознак, що зменшує витрати на обчислення без суттєвої шкоди для точності. DiceNet спирається на ці концепції, запроваджуючи багаторозгалужену архітектуру з різними темпами розширення для виділення ознак у різних масштабах, підвищуючи як точність, так і ефективність у середовищах із низьким ресурсом. ESPNet використовує ефективні просторові піраміdальні структури разом із поточковими згортками для обробки різноманітних просторових особливостей у різних масштабах, одночасно з високою обчислювальною ефективністю. Незважаючи на ці досягнення, загальним вузьким місцем у цих архітектурах є покладання на поточкові (1x1) згортки, які, хоч і ефективніші, ніж стандартні згортки, все ж роблять значний внесок у загальну вартість обчислень, особливо на більш глибоких рівнях мережі. Крім того, розміри фільтрів часто оптимізовані для продуктивності в хмарних середовищах, але можуть бути не ідеальними для периферійних середовищ, де обчислювальна швидкість й енергоефективність є вирішальною. Ми бачимо потенціал у зміні розмірів фільтрів у деяких шарах до 2x2, що є найменшим можливим фільтром для вилучення просторової інформації. Також слід звернути увагу на те, як інформація поширюється між каналами, а також на те, як кількість каналів формується шляхом заміни згортки 1x1 іншою передбачуваною математичною операцією.*

Ключові слова: згорткові нейронні мережі; класифікація зображень; комп'ютерний зір

Link to publication

APA Hozak, Ya. & Paliy, S. (2024). Modern small networks for image classification. Feature analysis. *Management of Development of Complex Systems*, 60, 221–229, dx.doi.org\10.32347/2412-9933.2024.60.221-229.

ДСТУ Гозак Я. Д., Палій С. В. Сучасні малі мережі для класифікації зображень. Аналіз особливостей. *Управління розвитком складних систем.* Київ, 2024. № 60. С. 221 – 229, dx.doi.org\10.32347/2412-9933.2024.60.221-229.