International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Comparitive analysis of Alexnet, GoogLeNet and EffecientNet using CIFAR-100 dataset

Dr. V. Siva Nagaraju, Saketh Peetha, and K. Druvan Karthik

Dept. Electronics and Communication Engineering
Institute of Aeronautical Engineering, Dundigal, Hyderbad-500043, India

***Abstract*—The CIFAR-100 dataset was implemented to help conduct this effective comparison study across the following models—AlexNet [8], GoogLeNet [8], and EfficientNetB0 [7]—under designs of animal, human, and many other classifications, respectively. To prove that the models differ significantly with respect to the time of training, accuracy, and loss, we ran our tests. Although EfficientNetB0 takes the longest to train, at 47 minutes per epoch, it obtained the lowest final loss and best validation accuracy of 92.3%. Validation accuracy of 66% and a training period of 10 minutes were the well-balanced with GoogLeNet. AlexNet was the fastest with its 3-minute training session, but then the least accurate at 35% together with the most loss. Accuracy is the main factor for which EfficientNetB0 should be used. GoogLeNet should be used where accuracy and time to train need to be balanced, and AlexNet should be the model to use if training more quickly is imperative, even if accuracy suffers as a result. Future research may then look into performance improvement and ways to apply these models. It may include data augmentation, transfer learning, advanced optimization approaches, hybrid approaches, validation on more real-world datasets beyond this bench marking dataset, and deployment in edge computing.**

***Index Terms*—AlexNet, Convolutional neural networks, CIFAR-100 dataset, Deep neural networks, EffecientNetB0, GoogLeNet, Image processing.**

I. INTRODUCTION

Many blocks of recent popularity of Deep Learning Networks have been triggered by the fact that they act effectively to classify information in an image and detect an image's information with regard to certain object classes. On the other side, DLNs have been executed through CNNs in several computer vision tasks like: object tracking, pose estimation, action recognition and Object Counting [6]. Examples range from yield predictions of fruits in vineyards to disease diagnosis, such as Parkinson's, using image data alone as in Heinrich et al. The increase in these applications of DLN results from enhanced computing powers and the need for faster training and inference by extensive RAM and GPU usage, respectively[13].

Beyond computer vision, CNNs have also been applied in speech recognition and natural language processing, where they outperformed the preceding algorithms mainly based on Hidden Markov models and Gaussian mixture models. Requirements for the execution of CNN architecture differ a lot depending on the domain of application. Generally, it aims at reducing inference times, while improving the times for prediction accuracy [3].

Currently, there is no standard guideline on how to effectively design CNN architectures for the solution of domain-specific problems, many of which return apparently random input-output outcomes, like manufacturing fault detection through image classification. To some extent, this challenge arises due to inadequate insight into the nature and performance implications of CNN architectures [4]. Moreover, in many cases, DLNs are considered a black-box model that would make the result hard to predict, thus making the benchmark results and evaluations very necessary. We focus on giving a state-of-the-art re-view of the evolution of architectures of DLN for image classification in this context, which delivers best-practice guidelines [13].

Since it is the most prevalent type of network used for this task, we particularly focus on CNNs. Our goals are to provide several evaluation results and metrics that outline the dynamics of characteristics of CNN architectures, their prediction performance, and computational requirements [5]. We also compare the different views among them over time to underline the technological trends in the design of the CNNs. We use this knowledge to formulate five guidelines to enhance the methodical selection of CNN designs and architectures [14].

II. LITERATURE

A. Convolutional Neural Network

CNN stands for Convolutional Neural Network, a Deep Learning model designed particularly to work with data having grid topology, such as images [16]. Inherent strengths of CNNs in recognizing patterns and structures within input data make them very effective when applied for tasks like image classification, object detection, and segmentation.

These layers convolve the input with filters to detect a set of local patterns, such as edges, textures,and shapes. In doing so, these layers reduce the spatial dimensions of the data and down sample it, reducing computational complexity and improving robustness to spatial variations. These layers are identical to traditional neural network layers, wherein every neuron is connected to all neurons in the previous layer [2]. They are normally used at the end of a network to make final predictions. Other activation functions like ReLU introduce non-linearity into the model to have it pick more complex patterns.

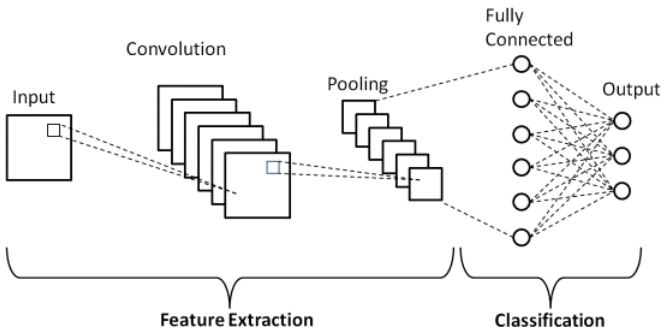

Fig. 1. Overview of CNN Architecture.

Images are high-dimensional data. Even a simple 256 x 256 RGB image has 196,608 values or pixels. Traditional neural networks struggle with handling input sizes this large. CNNs mitigate this problem by using convolutions which are local to receptive fields, greatly reducing the number of parameters and computational load [12]. CNNs can automatically learn to detect hierarchical features in images. Early layers capture simple patterns, including edges and textures, while deeper layers enable the extraction of more complex structures, including shapes and objects [17]. The hierarchical learning that occurs in this process plays a key role in image recognition, where it helps to understand different levels of abstractions.

By the very nature of convolution and pooling operations, a priori translational invariance relative to the input image is guaranteed in CNNs [16]. This means they are able to detect an object regardless of where it may be in the image, which makes them useful during object detection and image classification. Images have a spatial structure, where nearby pixels will tend to be more related than distant ones. CNNs exploit this property through local connections and shared weights, allowing them to effectively capture and process spatial hierarchies within images [16].

Image processing requires the extraction of meaningful features relating to edges, textures, and shapes. Convolutional layers of CNNs extract these features automatically during training; there is no need to engineer features manually [12]. Because CNNs are committed to weight sharing and local connectivity, they have fewer parameters compared to fully connected networks, and their computational power is reduced. This makes them more efficient for large images, reducing the possibility of over fitting.

Up to now, CNNs have been greatly validated and have given state-of-the-art results in nearly all image processing-related tasks. From winning the ImageNet competition with AlexNet in 2012 to modern computer vision applications in facial recognition, autonomous driving, and medical image analysis, it has already been established beyond doubt that CNNs work [19]. CNNs are specialized neural networks for extracting the high dimensionality, hierarchical feature learning, and spatial hierarchies inherent in image data. The ability of automatic and efficient extraction of meaningful features means that they have really extensive applications in image processing.

B. Deep Neural Network

deep neural network is a type of Artificial Neural Network with multiple layers between input and output layers. Multiple layers in DNNs empower them to model complex patterns in data incorporating the learning of hierarchical representations [17].

Every layer in a DNN is composed of neurons, which are the basic processing units. The neurons in one layer are connected to the neurons in the next through weighted connections; these weights start the training and will aim to produce predictions with minimal error.

Activation functions induce non-linearity into the network and thus allow it to learn more complex patterns. Common activation functions are the ReLU—rectified linear unit,the sigmoid and tanh [2]. The traditional training of DNNs involves backpropagation—a means of propagating an error backwards through the network as a means of updating the weights. It is normally combined with optimization algorithms such as stochastic gradient descent [12] . DNNs are very effective in applications like speech recognition, natural language processing, and image recognition due to their capability of learning intricate patterns and representations in data.

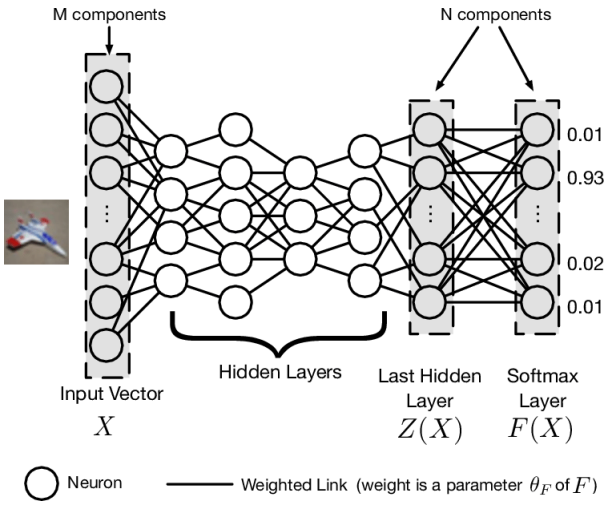

Fig. 2. Overview of DNN Architecture.

Training of DNNs, however, is compute-intensive, calls for large amounts of labeled data, and is model-na¨ıve. Techniques such as transfer learning have shown great potential in reducing these challenges, whereby a model trained on some other task is fine-tuned for a new task [19]. Notably, also successfully applied in various fields including health care, for example, to support diagnosis from data, autonomous systems, where they enable tasks such as self-driving cars, are other endpoints of application. To avoid the over fitting of DNNs and improve their generalization ability on new data, most of

the cases will have them with regularization techniques. This includes dropout and batch normalization [12].

C. CIFAR-100 dataset

Among these fully open datasets, CIFAR-100 represents a widely used benchmark in ML and computer vision. It was created by the Canadian Institute for Advanced Research; it contains 60,000 color images of dimensionality 32x32x3, each assigned one of 100 different classes [20]. Each image in the CIFAR-100 dataset consists of 32x32 pixels. This dataset contains training images and test images. The classes are further divided into 20 superclasses, each containing five subclasses. The CIFAR-100 dataset is designed to be more challenging than the CIFAR-10 dataset, which has only 10 classes [12]. It is, hence, an excellent resource that can be used to benchmark the performance of sophisticated machine learning algorithms.

Fig. 3. Overview of CIFAR-100 dataset.

Each image in the CIFAR-100 dataset is labeled with a ”fine” label, indicating the specific class, and a ”coarse” label indicating the super class; this hierarchical labeling structure allows for experiments to be conducted of fine-grained and coarse-grained classification [21]. This is a very well-balanced test bed for bench marking the skill level of machine learning models in dealing with sophisticated visual tasks: recognizing objects from a very diverse set of categories given small image sizes and limited resolution [20]. It has served as a benchmark in many neural network architectures, including CNN, ResNets, and DenseNets. Given the high complexity and diversity, CIFAR-100 is used in many works today for developing and evaluating new machine-learning algorithms, notably deep learning. The publicly available benchmark dataset gained wide acceptance within the research community, and massive improvements were realized in fields like image classification and recognition [21].

III. Methodology used

A. AlexNet

There is a very serious over fitting problem in conceptual design with regard to a large number of parameters of a convolutional neural network for image categorization [2]. trained their Alex-Net model with a number of different augmentations applied to the training inputs, therefore reducing this impact and ensuring that the model learned more generic characteristics rather than noise. Also, dropout was introduced as a regularization technique where some units are turned off, thus saving only the most important ones. This was incorporated in the fully linked layers to bring in some robustness. Alex-Net has a total of eight layers, three of them fully connected and five convolutional with corresponding pooling layers.

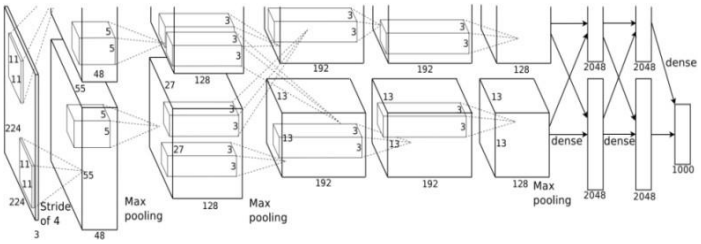

Fig. 4. AlexNet Architecture [2].

The inception module is made of many convolutional layers with filters that are different. The 1×1 convolutional layers are applied before the more expensive 3×3 and 5×5 filters are ap-plied. At 64 feature maps, the input image is reduced to 112×112 before the first Convolutional layer with a 7×7 filter and stride 2 is applied [1]. The first inception module is shown following layer C3. Despite these inception modules having different resolutions in their feature maps, they still use the same size of filters. Compared to the traditional three fully connected layers, Google-Net has nine inception modules. The model does not apply a single fully connected layer before the softmax activation but applies an average pooling layer. Apart from this difference, multiple crop sizes are used such as 1, 10 and 144 in case of Google-Net for multi-scale voting instead of using a static size.

B. GoogLeNet

For enhancing inference and model training efficiency, GoogLeNet was proposed as a way of optimizing the use of computing resources inside CNN architecture [1] computing power required for model inference can be reduced while ensuring a deeper architecture with greater feature maps is maintained. This enhancement is considered very important considering how common embedded and mobile computers are becoming. These savings were enabled by limiting the maximum inference time while designing it. This one is

attributed to an architectural building piece, which acts as a network inside a network, and it is the inception module. This module uses 1×1 Convolutional layers to reduce feature map size [8].

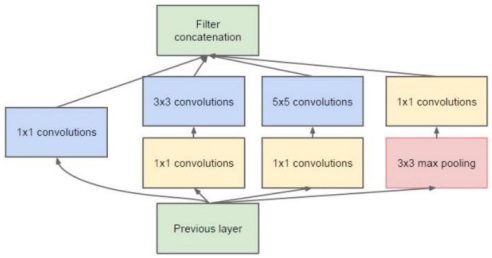

Fig. 5. GoogLeNet Architecture [1].

The inception module is made of many convolutional layers with filters that are different. The 1×1 convolutional layers are applied before the more expensive 3×3 and 5×5 filters are applied. At 64 feature maps, the input image is reduced to 112×112 before the first Convolutional layer with a 7×7 filter and stride 2 is applied [1]. The first inception module is shown following layer C3. Despite these inception modules having different resolutions in their feature maps, they still use the same size of filters. Compared to the traditional three fully connected layers, Google-Net has nine inception modules. The model does not apply a single fully connected layer before the softmax activation but applies an average pooling layer Apart from this difference, multiple crop sizes are used such as 1, 10 and 144 in case of Google-Net for multi-scale voting instead of using a static size.

C. EffecientNet

EfficientNet-B0 is the most revolutionary CNN architecture, acclaimed for its great speed and parameter efficiency. This is because it has a more organized way of scaling up the CNN models through a compound scaling method that grows network parameters like depth, breadth, and resolution equally [7]. In carrying out classification tasks, EfficientNet-B0 is mainly used as a network for spatial feature extraction. The EfficientNet family happens to comprise seven models: EfficientNet-B0 through EfficientNet-B7 models. This can be proved to be the case, as EfficientNet-B0 has more accuracy than ResNet-50 for the same input size with fewer parameters and FLOPs. It can be divided into seven blocks based on the convolutional filter size and stride, and also on the number of channels for each block. The basic building block of EfficientNet-B0 is the mobile inverted bottleneck (MBConv), inspired by the MobileNet idea [11].

MBConv includes a depthwise convolutional layer, followed by a squeeze-and-excitation block, a dropout layer, and two 1x1 convolutional layers. The first convolutional layer will expand the channels, and then the number of parameters will be decreased by the depthwise convolution [12]. It is in this sense that the SE block refines the model by re-accentuating the relationship between stations through the application of

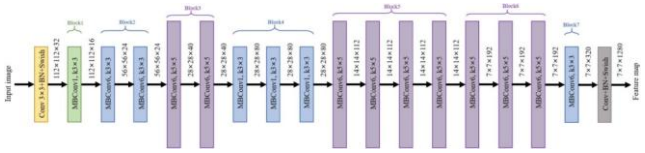

Fig. 6. EffecientNet-B0 Architecture [8].

distinctive weights for each channel, rather than all having equal importance. Once done, these channels are compressed by the second convolution layer [10].

IV. TABLE I

Table (1): Comparison of EffecientNetB0, GoogLeNet, and AlexNet with validation accuracy and loss

Table 1		
Model	Validation Accuracy	Validation Loss
EffecientNetB0	92%	40%
GoogLeNet	66%	10%
AlexNet	35%	21%

RESULTS AND DISCUSSIONS

In this study, we compared the performance of AlexNet, GoogLeNet, and EfficientNet using the CIFAR-100 dataset. The CIFAR-100 dataset was chosen due to its extensive range of classifications, including various animals and human objects, providing a robust benchmark for evaluating the object detection capabilities of these models. Each model was trained with a learning rate of 0.01 to ensure a fair comparison. The training was carried out over several epochs, and the performance was assessed based on validation accuracy and loss.

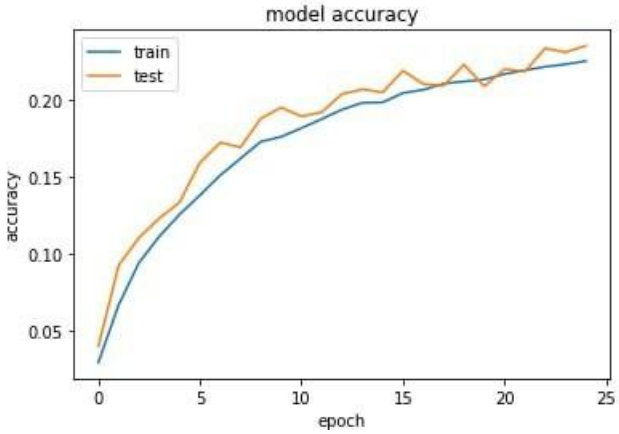

Fig. 7. AlexNet Accuracy vs. epoch's graph.

EfficientNet achieved the highest validation accuracy at 92%, with a validation loss of 40%. GoogLeNet followed with a validation accuracy of 66%, which is 31% lower than EfficientNet but 13% higher than AlexNet, and a validation loss of 10%. AlexNet recorded the lowest validation accuracy

at 35% and a validation loss of 29%. These results indicate that EfficientNet is significantly more accurate in detecting objects in the CIFAR-100 dataset compared to GoogLeNet and AlexNet. The higher accuracy of EfficientNet suggests that it is better suited for applications requiring high precision in object detection.

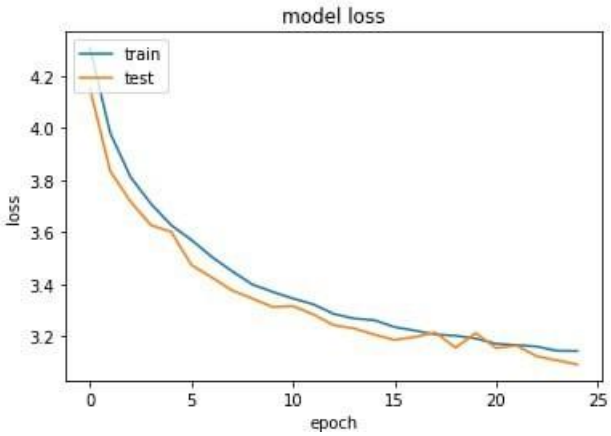

Fig. 8. AlexNet Loss vs. epoch's graph.

The loss values for the models decreased progressively over the training epochs. AlexNet started with a high loss of 43%, which decreased exponentially to 21% by the end of the training. GoogLeNet began with a lower initial loss, which was com-parable to the lowest loss recorded by AlexNet, indicating better initial performance and overall lower loss. EfficientNet started with a loss of 35% and decreased rapidly to 4%, demonstrating efficient learning and the lowest final loss among the models. The rapid decrease in loss for EfficientNet suggests that it learns more effectively than the other models, leading to better overall performance.

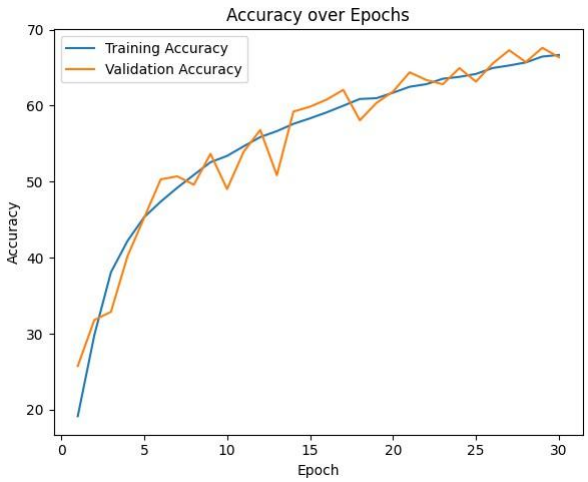

Fig. 9. GoogLeNet Accuracy vs. epoch's graph.

Training time per epoch varied significantly among the models. AlexNet took approximately 3 minutes per epoch, making it the fastest to train. GoogleNet took about 10 minutes per epoch, while EfficientNet took nearly 47 minutes per epoch. Although AlexNet had the shortest training time, it also exhibited the highest learning loss and lowest accuracy. GoogLeNet offered a balance between training time and accuracy, making it a viable option when both factors are critical. EfficientNet, despite its longer training time, provided the highest accuracy and lowest final loss, making it the best performer in terms of accuracy but the most time-consuming to train. These insights highlight the trade-offs between model complexity, training time, and accuracy.

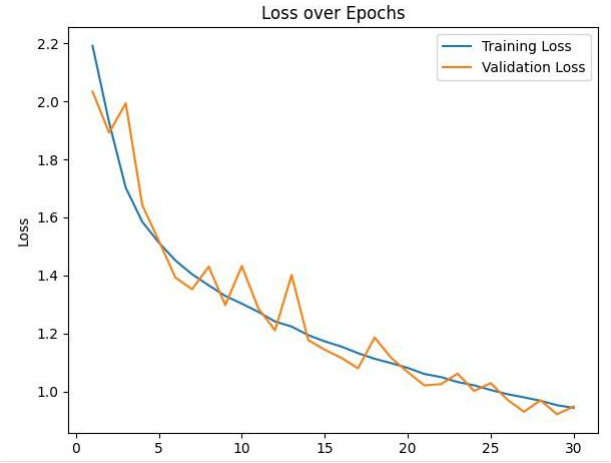

Fig. 10. GoogLeNet Loss vs. epoch's graph.

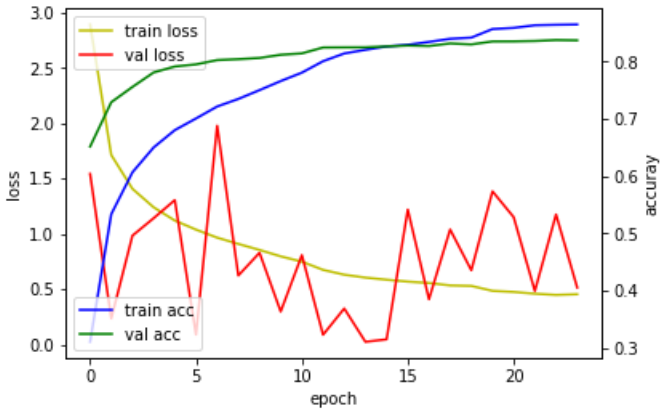

Fig. 11. EffecientNetB0 Accuracy and Loss vs. epoch's graph.

In comparing AlexNet, GoogLeNet, and EfficientNet on the CIFAR-100 dataset, EfficientNet achieved the highest accuracy and lowest loss but required the longest training time. GoogLeNet balanced accuracy and training time well, while AlexNet trained fastest but performed least favor-ably.EfficientNet is recommended for high-accuracy needs, GoogLeNet for balanced performance, and AlexNet when quick training is essential despite lower accuracy.

V. Conclusion

In summary, the CIFAR-100 dataset comparison for AlexNet, GoogLeNet, and EfficientNetB0 underlines the com-

promises between model complexity, time to train, and accuracy. While requiring the longest time per epoch to train at 47 minutes, EfficientNet showed both the best accuracy, at 92.3%, and the lowest final loss. GoogLeNet had an accuracy of 66% and a training period of 10 minutes per epoch, which forms a reasonable balance and so, would be suitable for applications which are equally critical on both counts. Even though AlexNet trained the quickest, three minutes per epoch, it turned out to perform the worst in both accuracy and loss, about 35%. These results set a balance of training limitations and accuracy of the models used, considering the application demands of each.

VI. Future Work

For high accuracy applications, EfficientNetB0 should be used; in situations that require a balance between—, Other methods can be followed to enhance the performance of these models further and make them more useful for object detecting tasks in the future. Implementing Rotation, Flipping, random Cropping, and Color jittering may enable model generalization. The fine-tuning of the already pretrained weights on the CIFAR-100 dataset available in GoogLeNet, EfficientNetB0, and AlexNet can achieve all accuracy and time efficiency goals. Other miscellaneous areas include sophisticated optimization techniques to train efficiently and to gain final accuracy based on features like Cosine Annealing, with adjustable learning rates like Ad-am/RMSprop, and learning rate Warm up. Moreover, it can also be improved upon by investigating hybrid models and group learning techniques that give both models better accuracy and resilience. It will be good to test on more challenged datasets, like COCO and Pascal VOC, and fine-tune these models on higher resolution images.

References

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, 2015. “Going Deeper with Convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, 2012. “Imagenet Classification with Deep Convolutional Neural Networks,” in Advances in Neural Information Processing Systems, pp. 1097–1105. (http://papers.nips.cc/paper/4824-imagenet-classification-w).

[3] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Ben-gio, 2007. “An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation,” in Proceedings of the 24th International Conference on Machine Learning, ACM, pp. 473–480.

[4] D. Mishkin, N. Sergievskiy, and J. Matas, 2017. “Systematic Evaluation of CNN Advances on the ImageNet,” Computer Vision and Image Understanding (161),pp.11–19.

[5] X. Cao, 2015. “A Practical Theory for Designing Very Deep Convolutional Neural Networks,” Technical Report.

[6] S. Ji, W. Xu, M. Yang, K. Yu: 3D convolutional neural net-works for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. 35(1), 221–231 (2012)

[7] Multi head attention based two stream EfcientNet for action recognition 1. Aihua Zhou1, 2· Yujun Ma, 3· Wanting Ji, 4· Ming Zong, 5· Pei Yang1, 6· Min Wu, 7. Mingzhe Liu. Received: 7 January 2022 / Accepted: 29 May 2022 / Published online: 24 June 2022 © The Author(s) 2022.

[8] Is Bigger Always Better? Lessons Learnt from the Evolution of Deep Learning Architectures for Image Classification Completed Research Paper Kai Heinrich TU-Dresden kai.heinrich@tu-dresden.de Christian Janiesch TU-Dresden christian.janiesch@tu-dresden.de Bjo¨rn Mo¨ller TU-Dresden bjoernmdot@web.de Patrick Zschech TU-Dresden patrick.zschech@tu-dresden.de

[9] M. Tan, Q. Le, (2019). Efcientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105–6114).PMLR.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L. C. Chen, (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4510–4520).

[11] N. Yudistira, T. Kurita : Correlation net: spatiotemporal multimodal deep learning for action recognition. Signal Process. Image Commun. 82, 115731 (2020).

[12] K. He, X. Zhang, S. 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).

[13] J. Cheng, P. Wang, G. Li, Q. Hu, and H. Lu, 2018. “Recent Advances in Efficient Computation of Deep Convolutional Neural Networks,” Frontiers of Information Technology and Electronic Engineering (19:1), pp. 64–77.

[14] Y. LeCun, 1989. “Generalization and Network Design Strategies in Connectionism in Perspective”, R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels (eds.), Elsevier.

[15] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R.Fergus and Y. LeCun, 2013. “OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks,” ArXiv:1312.6229 [Cs].

[16] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE.

[17] I. Goodfellow, Y. Bengio, A. Courville, (2016). Deep Learning. MIT Press.

[18] V. Nair, G. E. Hinton, (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on Machine Learning (ICML).

[19] K. Simonyan,A. Zisserman, (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

[20] Y. Li, L. Zhang, (2021). A comprehensive review of the CIFAR-100 dataset. Journal of Machine Learning Research.

[21] A. Krizhevsky, (2012). Learning Multiple Layers of Features from Tiny Images. Technical report.