STM JOURNALS

Journal of
Communication Engineering & Systems

ISSN: 2249-8613 (Online)
ISSN: 2321-5151 (Print)
Volume 14, Issue 2, 2024
May–August
DOI (Journal): 10.37591/JoCES

https://journals.stmjournals.com/joces

Review JoCES

Performance Analysis of Deep CNN Architectures

Radhey Shyam*

Abstract

A Convolutional Neural Network (CNN) is an artificial neural network renowned for its remarkable ability to handle large image datasets effectively, particularly excelling in tasks such as image recognition and classification. The fundamental structure of a CNN relies on mathematical convolution operations, comprising essential components such as convolutional layers, activation functions, pooling layers, and fully connected layers. These components work synergistically to extract and learn hierarchical features from input data, enabling CNNs to achieve high accuracy in various machine learning applications. CNNs have demonstrated exceptional performance across a range of fields, including computer vision, natural language processing, medical diagnosis, autonomous driving, and robotics. This study aims to provide an in-depth exploration of CNNs, elucidating their internal mechanisms and highlighting the critical parameters that influence their efficacy and performance. Selecting the optimal CNN architecture is contingent on several factors, including dataset complexity, computational resources, and the specific requirements of the task at hand. While pre-established models like EfficientNet and ResNet have shown considerable versatility and effectiveness, it is often prudent to experiment with and fine-tune architectures to better suit specific applications and achieve optimal results. Through a comprehensive analysis, this study underscores the importance of tailored approaches in the deployment of CNNs for diverse and complex machine learning tasks.

Keywords: Artificial neural network, convolutional neural network, machine learning, computer vision

INTRODUCTION

A convolutional neural network (CNN) is a specific form of artificial neural network developed to replicate the visual processing capabilities of humans, especially in computer vision applications. LeCun and Bengio introduced it in 1995, and its name "convolutional neural network" derives from its utilization of a mathematical process called convolution [1]. Early versions of CNNs found application in a wide array of tasks such as image recognition, text recognition, handwritten digit recognition, banking, anomaly detection, drug discovery, health risk assessment, aging biomarker detection, checkers game, time series forecasting, postal services, and more. Moreover, CNNs can also be employed in video analysis. Training CNNs typically requires large datasets, although they can be adapted to smaller image datasets using techniques like transfer learning [1–5].

Deep learning, including CNN approaches, has proven to be effective for analyzing large datasets, with successful applications in computer vision, pattern recognition, speech recognition, natural language processing, and recommendation systems [6]. Nowadays, CNNs are extensively utilized in cutting-edge applications such as pattern recognition, autonomous driving systems, smart speakers employing deep learning, verification of surgical sites in cataract surgeries, diabetic retinopathy diagnosis, and radiology [7–10].

Convolution denotes a particular kind of linear function. CNNs exhibit resemblances to conventional neural networks, consisting of neurons

***Author for Correspondence**
Radhey Shyam
E-mail: shyam0058@gmail.com

Professor, Department of Information Technology, Shri Ramswaroop Memorial College of Engineering & Management, Lucknow, Uttar Pradesh, India

Received Date: March 10, 2024
Accepted Date: May 22, 2024
Published Date: July 02, 2024

Citation: Radhey Shyam. Performance Analysis of Deep CNN Architectures. Journal of Communication Engineering & Systems. 2024; 14(2): 1–8p.

with modifiable weights and biases. These neurons accept inputs, perform a dot product operation, and may incorporate non-linearities. Frequently used non-linear functions include sigmoid, tanh, ReLU, and softplus [11].

The CNN architecture includes various layers such as convolutional layers, which employ convolutional operations with multiple kernels, non-linear activation layers, pooling layers, flattening, and fully connected layers. Despite their effectiveness in recognizing objects, CNNs can identify items that are imperceptible to human vision. A typical CNN architecture is depicted in Figure 1. However, CNNs face two primary challenges: the need for large amounts of image data and susceptibility to overfitting. To address these challenges, CNNs are typically trained on extensive datasets within a specific domain. Even after convergence of network parameters, additional training steps may be necessary, especially when dealing with small training sets [12–14].

The paper is structured in the following manner: The next Section presents the terminology related to convolutional neural networks. The Section after that delves into a comprehensive understanding of CNN layers. Dominant architectures of deep convolutional neural networks are summarized in the following Section; and the last Section outlines the discussion and conclusion of the paper.

TERMINOLOGY ASSOCIATED WITH CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) have specific terms that are important to understand for effective communication and comprehension. Here are some key terms:

- *Convolutional Layer:* The basic unit of a CNN, where input data undergo convolution operations using filters or kernels.
- *Kernel/Filter:* A compact grid utilized within convolutional layers to derive characteristics from input data via convolution operations.
- *Stride:* The increment employed when the kernel is applied to the input data during convolution, dictating the extent to which the kernel traverses the input.
- *Padding:* Including additional pixels surrounding the input data to maintain spatial dimensions post-convolution.
- *Activation Function:* An activation function applied to the output of neurons, introducing non-linear characteristics to the network. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh.
- *Pooling Layer:* A layer utilized to decrease the spatial dimensions of the feature maps produced by convolutional layers, usually achieved through operations such as max pooling or average pooling.
- *Fully Connected Layer:* Also referred to as a dense layer, this layer establishes connections between each neuron and all neurons in both the preceding and succeeding layers, facilitating the amalgamation of high-level features and classification.
- *Flattening:* The process of converting multi-dimensional feature maps into a one-dimensional vector before feeding them into fully connected layers.

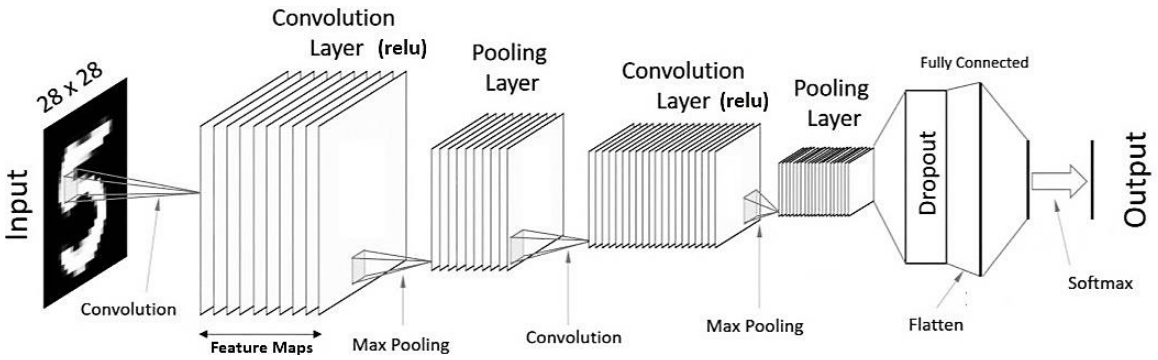

Figure 1. Architecture of a typical convolutional neural network [15].

- *Backpropagation:* The process of updating the neural network weights by backpropagating the error from the output layer to the input layer.
- *Epoch:* A single iteration of the entire dataset passing through the neural network during training.
- *Batch Size:* The quantity of data samples sent through the network before weight updates occur during training.

DESCRIPTION OF CONVOLUTIONAL NEURAL NETWORKS LAYERS

The essence of convolution lies in its mathematical operation involving two functions, denoted as f and g, which results in a third function. This operation illustrates how the shape of one function is altered by the other, as defined in Digital Signal Processing.

The Feature Learning Phases

In a CNN, the feature learning process involves multiple pairs of convolutional and pooling layers. The selection and arrangement of these layers are determined by engineering considerations specific to the problem at hand. However, as the network progresses deeper, these layers tend to capture increasingly abstract features or patterns, akin to the functioning of the human visual cortex as proposed in our assumed model [16]. In RGB images, each color channel is convolved separately with appropriate kernels.

Convolution with Single Kernel in a Convolution Layer

In this example, we examine a 5×5×3 kernel, moving it across the entire input array with a stride of 1. For simplicity, we momentarily disregard the color dimensions. At each step of this movement, we compute the dot product between each element of the kernel and each element of the corresponding subarea of the input array. Each dot product yields a scalar value. With a total of 28×28 unique positions where the kernel can be placed on the image, the resulting feature map is a 28×28×1 array. If the stride is increased beyond 1, the resulting feature map becomes more reduced in size.

Illustration of the Convolution Operation

Let us examine a 7×7 input array consisting of 49 elements. Placed at the center of this array is a 3×3 kernel represented in black and white. The stride is set to 1. This serves as an example of a kernel designed to identify diagonal patterns (represented as "1") along the diagonals. The resulting output is a 5×5 array containing 25 elements. We systematically move the kernel across the input array, covering 25 distinct positions. At each position, we compute the dot product element-wise and store the result in the output matrix.

Padding

One weakness of the convolution step is the potential loss of information at the borders of the image, as these areas may not be fully captured by the kernel as it moves across the image. To address this issue, a straightforward yet effective approach is to utilize zero-padding. Zero-padding refers to the practice of appending additional rows and columns of zeros to the input array. This ensures that information at the borders of the image is preserved and can be fully processed by the kernel during convolution. Additionally, zero-padding helps manage the output size of the convolution operation. The choice of whether to apply padding relies on several factors, including the dimensions of the input array, the size of the kernel, and the stride length. Without padding, the input and output arrays may rapidly shrink in size as the convolution operation progresses. However, systematic use of padding helps maintain the size of the arrays throughout the convolution process.

Repeated Convolution for all Kernels in a Convolution Layer

In each convolutional layer, there exists a collection of distinct kernels. These kernels operate independently, each convolving with the input image separately. For example, within the provided illustration, there are six kernels in the initial convolutional layer. This results in the generation of 6 feature maps, each with a shape of 28×28×1.

Pooling (Subsampling) Layers

Pooling layers are frequently incorporated into convolutional neural networks (CNNs) to gradually reduce the spatial dimensions of the feature maps. This reduction in size helps in controlling the number of features and the computational complexity of the network, ultimately aiding in preventing overfitting.

In the context of image processing, pooling can be likened to reducing the resolution of the images. It is important to note that pooling does not affect the number of kernels present in the network. The selection of kernel size, stride, and potentially padding is relevant for the pooling process.

There are typically three types of pooling operations: max pooling, average pooling, and min pooling. Max pooling selects the maximum element from the region covered by the kernel, while average pooling calculates the average, and min pooling selects the minimum. Max pooling is the most commonly used approach. For instance, in max pooling with a 2×2 window, a kernel of size 2×2 traverses over the entire feature map with a stride of 2, selecting the largest element for the next representation map [12, 17, 18].

In contrary to that, average pooling computes the average value of each cluster to preserve spatial information and avoid excessive downsampling, a stride of one can be employed, although this is less common. It is essential to consider that downsampling may result in the loss of positional information and should be applied judiciously, particularly when preserving the spatial arrangement of information is crucial [19].

Flattening

After passing through the convolutional and pooling layers, and before entering the fully connected layers, the output of the preceding layers undergoes a flattening process. This entails converting the multi-dimensional data into a one-dimensional array, preparing it for input into the subsequent layer. Essentially, the output of the convolutional layers, typically represented as a 3D array with dimensions like 10×10×10, is flattened into a 1D array containing 1000 elements.

Fully Connected Layers

The fully connected layers receive as input a flattened array that represents the activation maps of high-level features obtained from earlier layers. These layers then output an N-dimensional vector, where N corresponds to the number of classes that the model needs to choose from. For example, in digit classification tasks, N would typically be 10 since there are 10 digits (0 to 9). The primary function of the fully connected layers is to discern which features are most strongly associated with a particular class. Different stages of the fully connected layers are depicted in Figure 2.

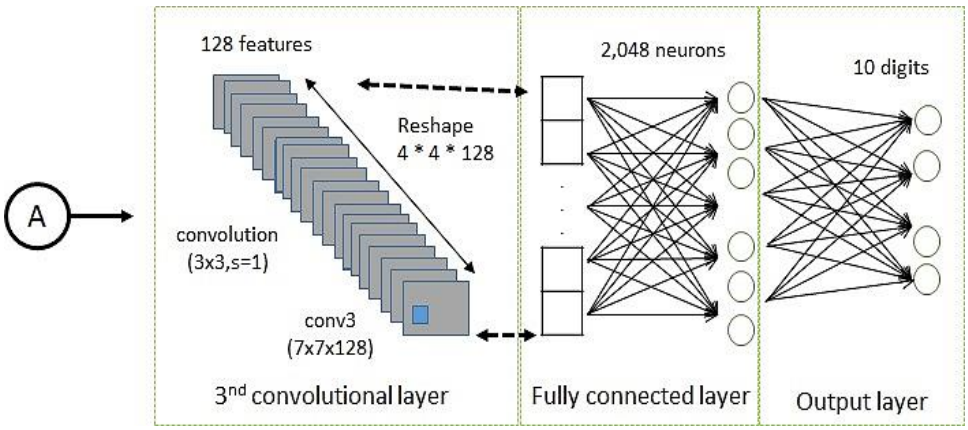

Figure 2. Illustration of different stages of fully connected layers [19].

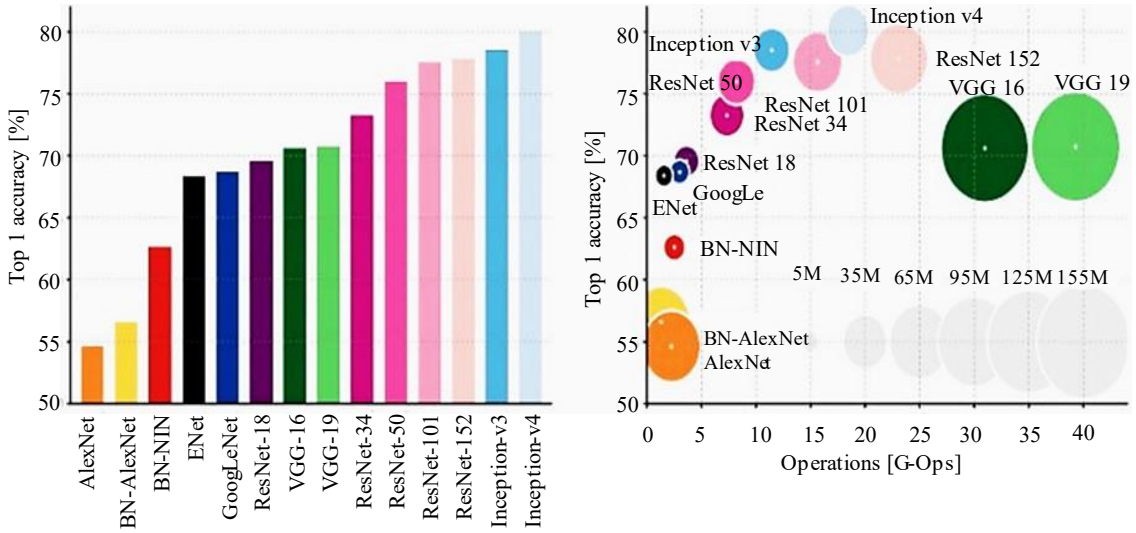

Figure 3. Performance analysis of different deep neural network architectures.

PERFORMANCE ANALYSIS OF DEEP CONVOLUTIONAL NEURAL NETWORK ARCHITECTURES (DCNN)

A comparative analysis of DCNN architectures is conducted, and it is visualized in Figure 3. These DCNN architectures are outlined as follows:

AlexNet

The network comprises around five convolutional layers and two fully connected layers to extract features. Max pooling is utilized following the 1st, 2nd, and 5th convolutional layers. With a total of 650,000 neurons, 60 million parameters, and 630 million convolutions, AlexNet exemplified the significant impact of deep learning in computer vision tasks, marking a pivotal moment in the field. This breakthrough was highlighted in various sources [18, 20].

BN-AlexNet

The BN-AlexNet architecture, short for Batch Normalization-AlexNet, is an adaptation of the original AlexNet architecture that incorporates Batch Normalization layers. These layers are included to standardize the activations within each layer across the network, aiding in the stabilization and speeding up of the training process. The BN-AlexNet architecture generally follows the same structure as AlexNet, which consists of multiple convolutional layers followed by max-pooling layers, fully connected layers, and finally, a softmax output layer for classification. However, in BN-AlexNet, Batch Normalization layers are inserted after the convolutional layers and before the activation functions [21].

BN-NIN

BN-NIN stands for Batch Normalization-Network in Network. It is an extension of the Network in Network (NIN) architecture with the incorporation of Batch Normalization (BN) layers. The NIN architecture, introduces the concept of micro neural networks called “network in network” units. These units are utilized within convolutional layers to capture complex patterns and improve feature representation.

ENet

ENet is particularly popular in the field of semantic segmentation and real-time image processing tasks, where it has demonstrated competitive performance with relatively low computational requirements compared to other deep learning architectures. It attains this efficiency by employing methods like factorized convolutions, which decrease the parameters and computational burden while maintaining accuracy. ENet has been used in various applications such as autonomous vehicles,

robotics, augmented reality, and more, where real-time processing is essential. Its efficient design makes it a valuable tool for developers seeking to deploy deep learning models on devices with limited computational resources.

GoogLeNet

In 2014, Google introduced GoogLeNet, also referred to as Inception-v1, a sophisticated convolutional neural network design. It emerged victorious in the ImageNet Large Scale Visual Recognition Challenge during the same year. It features a complex structure with multiple layers and utilizes the "Inception module" which includes various convolutional filters of different sizes in the same layer to efficiently capture features at different scales. GoogLeNet aims for high accuracy in image classification tasks while being computationally efficient, making it suitable for real-time applications like mobile image processing. Subsequent versions like Inception-v2, v3, and v4 have built upon its design, improving both accuracy and efficiency.

ResNet-18

Microsoft Research introduced ResNet-18 in 2015 as a variant of the ResNet architecture. It is known for its 18 layers and uses skip connections to enable training of deep networks without the vanishing gradient problem. These connections improve accuracy and training speed. ResNet-18 is often used in computer vision tasks like image classification and object detection due to its balance between complexity and performance. It is also suitable for deployment on devices with limited resources.

VGG-16

Proposed by K. Simonyan and A. Zisserman in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition," this model achieved a top-5 test accuracy of 92.7% on the ImageNet dataset, which comprises over 14 million images across 1000 classes. AlexNet gained popularity for its architecture, which includes 13 convolutional layers and 3 fully connected layers, adhering to the ReLU activation tradition and utilizing smaller kernel sizes such as 2×2 and 3×3. With approximately 138 million parameters, AlexNet demonstrated its effectiveness in image recognition tasks [22].

VGG-19

VGG-19 shares a similar architectural design with VGG-16 but includes three additional convolutional layers, resulting in a total of 16 convolutional layers and 3 dense layers. In VGG networks, employing 3×3 convolutions with a stride of 1 achieves an effective receptive field equivalent to that of a 7×7 convolution. This approach reduces the number of parameters required for training.

ResNet-34

ResNet-34, similar to ResNet-18, is a type of Residual Neural Network created by Microsoft Research. It is deeper, with 34 layers, and uses skip connections to facilitate training deep networks effectively. This architecture enables higher accuracy in tasks like image classification and object detection. However, due to its increased complexity, ResNet-34 may demand more computational resources compared to smaller variants.

ResNet-50

ResNet50, a modified version of the ResNet architecture, consists of 48 convolutional layers, accompanied by 1 max pooling layer and 1 average pooling layer. It belongs to the category of even deeper CNNs, with approximately 152 layers, making it 8 times deeper than VGG nets, yet it maintains lower complexity [23].

ResNet-101

ResNet-101, which is a component of the ResNet architecture, was created by Microsoft Research. With its 101 layers, it is deeper than previous versions. Like other ResNet models, it uses skip connections to ease training, allowing gradients to flow more efficiently during training for improved

accuracy and convergence. ResNet-101 is powerful, excelling in tasks like image classification and object detection. However, its increased depth means it may demand more computational resources compared to smaller variants like ResNet-18 or ResNet-34.

ResNet-152

ResNet-152, part of the ResNet architecture developed by Microsoft Research, is renowned for its depth with 152 layers. Like other ResNet models, it uses skip connections to ease training by mitigating the vanishing gradient problem. This enhances accuracy and convergence speed. ResNet-152 surpasses ResNet-101 in complexity and capability, excelling in tasks like image classification and object detection. However, its deep structure demands significant computational resources for both training and inference.

Inception-V3

Inception-V3, succeeding Inception-V1, features 24 million parameters. It has become a popular image recognition model known for achieving over 78.1% accuracy on the ImageNet dataset. However, despite its effectiveness, it is not frequently utilized [24].

Inception-V4

This version represents an enhancement over Inception-V3, with a notable difference found in the stem group. The Inception architecture has demonstrated impressive performance with a relatively lower computational cost.

DISCUSSION AND CONCLUSION

This study offers a thorough exploration of convolutional neural networks (CNNs) and sheds light on how different parameters impact network performance. The convolutional layer emerges as a crucial element in CNNs, contributing significantly to processing time. Network effectiveness is also influenced by its depth, although deeper networks tend to require longer training and testing times. CNNs have become a powerful tool in contemporary machine learning, finding applications in various domains such as face detection, image and video recognition, and more. The study also discusses popular deep CNN architectures, providing insights into their design and performance. Selecting the most suitable CNN architecture depends on factors like dataset complexity, computational resources, and task requirements. While models like EfficientNet and ResNet have shown versatility, it is important to experiment and customize architectures to meet specific needs.

REFERENCES

1. LeCun Y, Bengio Y. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks. 1995 Apr.
2. Abdel-Hamid O, Deng L, Yu D. Exploring convolutional neural network structures and optimization techniques for speech recognition. In Interspeech. 2013 Aug 5; 2013: 1173–5.
3. Singh S, Jaiswal T, Shyam R, Khanna S. Evaluation of Tweet Sentiments Using NLP. In International Symposium on Intelligent Informatics 2022 Aug 31. In: Thampi SM, Mukhopadhyay J, Paprzycki M, Li K-C, editors. Singapore: Springer Nature Singapore; 2023; 225–238.
4. Simran, S. Tandon, S. Khanna, and R. Shyam. Detection of traffic sign using CNN. Recent Trends in Parallel Computing. 2022;9(1):14–23.
5. Shyam R, Mishra A, Kumar A, Chowdhary A, Srivastava AK. Recording of Class Attendance Using DL-Based Face Recognition Method. In International Conference on Data Science and Applications. Singapore: Springer Nature Singapore; 2023 Jul 14; 249–260.
6. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE. A survey of deep neural network architectures and their applications. Neurocomputing. 2017 Apr 19; 234: 11–26.
7. Yoo TK, Oh E, Kim HK, Ryu IH, Lee IS, Kim JS, Kim JK. Deep learning-based smart speaker to confirm surgical sites for cataract surgeries: A pilot study. PLoS one. 2020 Apr 9; 15(4): e0231322.
8. Shyam R, Singh R. A taxonomy of machine learning techniques. J Adv Robot. 2021 Dec; 8(3): 18–25.

9. Pratt H, Coenen F, Broadbent DM, Harding SP, Zheng Y. Convolutional neural networks for diabetic retinopathy. Procedia Comput Sci. 2016 Jan 1; 90: 200–5.
10. Ni J, Chen Y, Chen Y, Zhu J, Ali D, Cao W. A survey on theories and applications for self-driving cars based on deep learning methods. Appl Sci. 2020 Apr 16; 10(8): 2749.
11. Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285. 2016 Mar 23.
12. Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. In 2017 IEEE international conference on engineering and technology (ICET). 2017 Aug 21; 1–6.
13. Maitra DS, Bhattacharya U, Parui SK. CNN based common approach to handwritten character recognition of multiple scripts. In 2015 13th IEEE International Conference on Document Analysis and Recognition (ICDAR). 2015 Aug 23; 1021–1025.
14. Srivastava V, Shyam R. Enhanced object detection with deep convolutional neural networks. Int J All Res Educ Sci Methods (IJARESM). 2021; 9(7): 27–36.
15. Shyam R. Convolutional neural network and its architectures. J Comput Technol Appl. 2021; 12(2): 6–14.
16. Ivry RB, Mangun GR. Cognitive Neuroscience: The biology of the mind. Norton; New York City; 2014.
17. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 2017 My 24; Volume 60, Issue 6 Pages 84–90.
18. Explore AI. (2015). TensorFlow - Convolutional Neural Network (CNN). [Online]. Available from: https://exploreai.org/p/tensorflow-cnn.
19. Deep learning book. (2024). ConvNets. [Online]. Available from: https://www.deeplearning book.org/contents/convnets.html
20. Zhou B, Chen D, Wang X. Seat belt detection using convolutional neural network bn-alexnet. In Intelligent Computing Theories and Application: 13th International Conference, ICIC 2017, Liverpool, UK, August 7–10, 2017, Proceedings, Part I 13; Cham: Springer International Publishing. 2017; 384–395.
21. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.
22. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2016; 770–778.
23. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2016; 2818–2826.
24. Szegedy C, Ioffe S, Vanhoucke V, Alemi A. Inception-v4, inception-resnet and the impact of residual connections on learning. In 31st Proceedings of the AAAI conference on artificial intelligence. 2017 Feb 12; 4278–4284.

