https://doi.org/10.62836/iaet.v1i1.1006

Global Science Publishing

IAET

Innovations in Applied Engineering and Technology

https://ojs.sgsci.org/journals/iaet

Article

Comprehensive Survey of State-of-the-Art Convolutional Neural Network Architectures and Their Applications in Image Classification

Xueting Pan [1,†], Ziqian Luo [2,*,†] and Lisang Zhou [3]

[1] *Oracle, Seattle, WA 98101, USA; xtpan8800@gmail.com*

[2] *Oracle, Seattle, WA 98101, USA*

[3] *Bazaarvoice Inc., Austin, TX 78759, USA; lzhou@berkeley.edu*

† These authors contributed equally to this work.

Abstract: Image classification is a vital research direction in computer vision all over the world. Before the advent of deep learning, image classification relied on manual feature extraction and conventional machine learning algorithms. However, Convolutional Neural Networks (CNNs) revolutionized this field by automatically learning features from data. The article discusses the fundamental principles of convolutional neural networks and compares various CNN architectures. Key layers such as convolutional, pooling, activation, fully connected, and dropout layers are explained in detail, along with techniques like backpropagation and optimization algorithms. Additionally, common CNN models like LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, SENet, and EfficientNet are introduced, highlighting their characteristics and applications.

Keywords: image classification; convolutional neural networks; computer vision; machine learning; deep learning

1. Introduction

Image classification is an important task in the field of computer vision, which refers to the process of assigning digital images to different predefined categories or labels. The improvement of image classification technology is a crucial component driving the development of computer vision, with its main processes including image data preprocessing, feature extraction and representation, and classifier design.

Before the rise of deep learning, image classification relied mainly on manually designed feature extraction methods such as Scale-Invariant Feature Transform and Histogram of Oriented Gradients. These methods involve extracting local features from images and using traditional machine learning algorithms for classification, such as Support Vector Machines (SVM) and Random Forests. In the 1960s, Huber and Wiesel proposed the concept of "receptive fields" while experimenting with the visual cortex cells of cats, which later was introduced into the research of CNNs. In the 1980s, Fukushima and Miyake proposed the "Neocognitron" based on "receptive fields," which can be seen as the first implementation of CNNs. Subsequently, researchers attempted to learn features using multilayer perceptrons and train models using the backpropagation (BP)

Received: 15 February 2022; Accepted: 7 March 2022.

* Corresponding: Ziqian Luo (luoziqian98@gmail.com)

algorithm. LeCun et al. proposed a CNN model called LeNet-5 [1], but it only achieved good results in handwritten digit recognition. The models proposed at that time encountered problems such as vanishing gradients, local optima, and overfitting as the depth of the network increased, leading to a dilemma in the research of deep neural network models.

Hinton et al. proposed effective learning algorithms to solve the difficulty of learning in multilayer neural networks. The learned features more fundamentally represent the data, which helps in visualizing or classifying the data. The problem of vanishing gradients in neural network training can be alleviated by normalized initialization, gradually triggering the boom of deep learning. With the improvement of hardware computing capabilities, such as the emergence of accelerators like GPU and TPU, and the continuous development of deep learning frameworks (such as TensorFlow, PyTorch, etc.), training and deploying deep learning models have become more efficient and convenient. For instance, the integration of techniques such as XGBoost in ultra-wideband radio technology, as mentioned by Liu et al. reflect a trend toward employing sophisticated machine learning methods across various engineering domains [2, 3]. Shen et al., Wang et al., and Deng et al. examine various aspects of lipid metabolism and protein functions, which can be correlated with advances in image recognition and classification techniques to enhance the visualization and analysis of cellular processes in cardiovascular research [4–6]. Qiu et al., explores the estimation of tail risk measures in finance and education through extreme value mixture modeling [7, 8]. Liu et al., discuss using machine learning and robotics for condition assessments, emphasizing their potential in visual recognition and image classification [9,10]. Deng et al., demonstrated a five-beam interference model for enhanced microfabrication control, incorporating deep learning [11,12]. Similarly, advanced research in PCSK9 secretion involving cargo receptor protein interactions indicates the level of detail being explored in biochemical signal pathways, which parallels the depth of study in neural networks for image classification [13–16]. Also, Even the challenges brought by remote learning can be alleviated through the application of computer vision and image recognition technologies to enhance the online learning experience [17]. These works collectively highlight the technological evolution and the integration of machine learning and deep learning techniques in diverse fields, not just in visual computing but across a spectrum that includes civil engineering, biosensing, and even nano photonics [18–21].

In the realm of image classification, particularly against adversarial attacks, the study in ACM Transactions on Embedded Computing Systems provides a novel approach for enhancing the efficiency and security of deep neural networks through non-iterative model pruning [22]. Additionally, Sun et al., offers insights into efficient image data handling, applicable to both telecommunications and large-scale image datasets [23]. Moreover, the transition challenges highlighted by Li et al. in their study on the sudden shift of engineering courses to remote learning shed light on adapting educational strategies in computer vision, a crucial consideration during unforeseen disruptions like public health crises [24,25]. These studies collectively advance the field of computer vision by integrating cross-disciplinary research findings into practical applications and education.

This article will focus on introducing the basic structure and principles of classical neural networks, as well as comparing and analyzing the advantages and disadvantages of various convolutional network architectures.

2. Overview of CNNs

Convolutional Neural Network (CNN) [26, 27] is a type of deep learning model specifically designed for processing data with grid-like structures, such as images and audio. CNNs utilize convolutional and pooling operations to progressively extract features from input data, and through multiple layers of nonlinear transformations, they map the data into higher-dimensional representation spaces. In this way, CNNs can capture both local features and global structures in the data, learning abstract representations of the data, thus enabling tasks such as classification, detection, or regression. CNNs have found wide applications in fields such as image processing, speech recognition, natural language processing, and have achieved state-of-the-art performance in many tasks.

2.1. Key Layers of CNN

2.1.1. Convolutional Layer

Convolutional layers are one of the core components of convolutional neural networks, used for extracting features from input images. It consists of multiple convolutional kernels, each learning specific features such as edges, textures, etc. The convolution operation slides the convolutional kernel over the input image, performing a dot product and summing at each position to generate an output feature map. This operation allows the network to retain spatial structural information of the input image and extract useful features from it.

Additionally, convolutional layers employ parameter sharing and sparse connections, effectively reducing the number of trainable parameters, thus reducing the model's complexity and computational load. This design enables convolutional layers to effectively handle the high-dimensional nature of image data, improving the network's generalization ability and performance.

In general, the size of the input feature map is h×w×c (height h, width w, channels c), and each convolutional kernel is of size k×k×c, where the number of kernels should be equal to the number of input channels. Figure 1 [26] illustrates the convolution process between the input feature map (5×5×3) and the convolutional kernel (3×3×3).

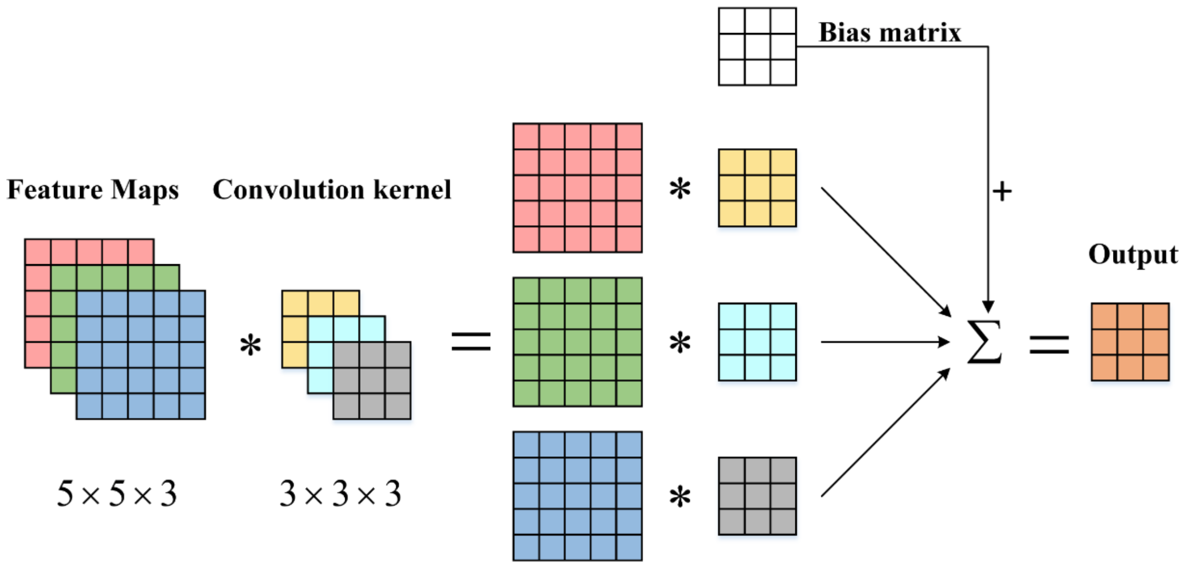


Figure 1. Convolution Process Diagram [26].

The convolution process can be roughly represented by the following formula:

$$f_{out} = f\left(\sum_{i=3}^{3} M_i * W_i + B\right) \quad (1)$$

Here, M_i represents the feature map of the input, W_i is the weight convolutional matrix, B is the bias matrix, $f(\cdot)$ denotes the non-linear activation function, and f_{out} is the output feature map.

The specific calculation in the convolutional layer involves the cross-correlation operation between the convolutional kernel and the feature map. For any input 2-D matrix size i, convolutional kernel size k, stride s, and padding p, the size [28] of the output feature map is:

$$o = \left[\frac{i+2p-k}{s}\right] + 1 \quad (2)$$

As shown in Figure 2 [26], assuming a simple cross-correlation operation between one of the mentioned feature maps and the convolutional kernel, the input is a feature map matrix with both height and width of 3. The convolutional kernel starts from the top left corner of the input matrix and slides over the input array from left to right and top to bottom in sequence.

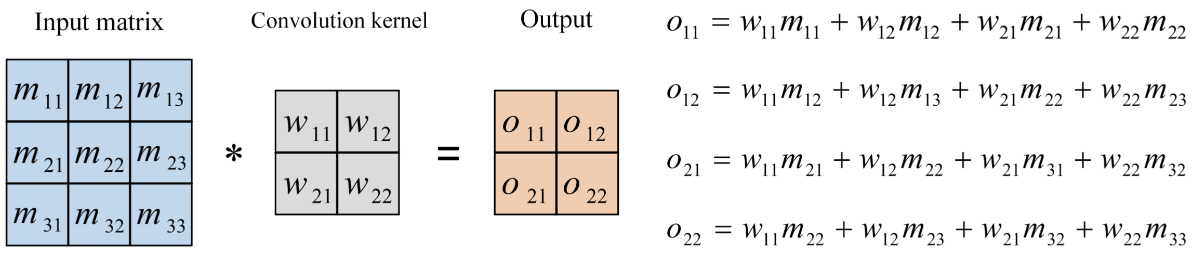

Input matrix
Convolution kernel
Output
m_{11} m_{12} m_{13} m_{21} m_{22} m_{23} m_{31} m_{32} m_{33}
w_{11} w_{12} w_{21} w_{22}
o_{11} o_{12} o_{21} o_{22}
$o_{11} = w_{11}m_{11} + w_{12}m_{12} + w_{21}m_{21} + w_{22}m_{22}$
$o_{12} = w_{11}m_{12} + w_{12}m_{13} + w_{21}m_{22} + w_{22}m_{23}$
$o_{21} = w_{11}m_{21} + w_{12}m_{22} + w_{21}m_{31} + w_{22}m_{32}$
$o_{22} = w_{11}m_{22} + w_{12}m_{23} + w_{21}m_{32} + w_{22}m_{33}$

Figure 2. Convolution operation [26].

2.1.2. Pooling Layer

The pooling layer typically follows the convolutional layer. The main purpose of using the pooling layer is to achieve down sampling and dimensionality reduction of the input image, thereby reducing the number of connections in the convolutional layer and lowering the computational burden of the network. Additionally, it can also achieve scale invariance, translation invariance, and rotation invariance of the input image, while enhancing the robustness of the output feature map to distortions and errors in individual neurons. In pooling operations, a sliding window is usually used to move over the feature map, with the distance of each movement determined by the stride. At each window position, the pooling layer performs pooling operations based on the feature values within the window. Common pooling operations include max pooling and average pooling, with max pooling being more common in practice as it reduces computational burden while retaining important features. Although there are other pooling methods such as Lp pooling, mixed pooling, and random pooling, they can more effectively mitigate overfitting issues in convolutional neural networks. The relationship between the input and output matrix sizes in pooling operations generally follows the following relationship [28]:

$$o = \left[\frac{i-k}{s} + 1 \right] \tag{3}$$

2.1.3. Activation Function

The activation function in Convolutional Neural Networks (CNNs) refers to the non-linear function applied to each neuron in the neural network. The role of the activation function is to introduce non-linearity, enabling the neural network to learn complex patterns and representations. The Table 1 lists common activation functions:

Table 1. Common Activation Functions**.**

Activation Function	Formula	Description
ReLU (Rectified Linear Unit)	$f(x) = \max(0, x)$	Mitigate the vanishing gradient problem and accelerate the convergence speed of the network.
Sigmoid	$f(x) = 1/(1+e^{-x})$	Map inputs to continuous values between 0 and 1, allowing the output to be interpreted as probabilities.
Tanh	$f(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$	Map inputs to continuous values between -1 and 1.
Leaky ReLU	$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \le 0 \end{cases}$	Address the issue of "neuron death" in ReLU function. αα is a small positive number, typically set to 0.01.

Cont.

Activation Function	Formula	Description
ELU (Exponential Linear Unit)	$f(x) = \begin{cases} x, & x > 0 \\ \alpha * (e^x - 1), & x \leq 0 \end{cases}$	Multiply by an exponentially increasing slope when the input is less than 0, enhancing the stability of the network.
Softmax	$f(x) = {e^x}/{\sum e^x}$	Typically used in the output layer of multi-class classification tasks, it transforms the raw output of the network into a form representing the probabilities of each class.

2.1.4. Fully Connected Layer

In Convolutional Neural Networks (CNNs), the fully connected layer is located at the end of the network, flattening the feature maps outputted by the convolutional and pooling layers into vectors, and connecting them to the output layer to complete classification or regression tasks. Each neuron in the fully connected layer is connected to all neurons in the preceding layer, resulting in many parameters. Non-linear transformations are introduced through activation functions. The fully connected layer plays a crucial role in extracting high-level features and providing effective feature representations. However, it is prone to overfitting, so it is often combined with regularization methods such as Dropout to improve the network's generalization ability.

2.1.5. Dropout

The Dropout layer is a regularization technique used to reduce overfitting in models. During training, Dropout randomly sets a portion of neuron outputs to zero, thereby reducing interdependencies between neurons and making the network more robust.

2.2. Loss Function

The loss function, also known as the cost function or objective function, plays a crucial role in machine learning and deep learning. It is used to measure the difference between the model's predictions and the true labels and serves as the core of optimization algorithms. By minimizing the loss function, the model parameters are optimized to make the model's predictions as close to the true labels as possible. The Table 2 below shows some common loss functions.

Table 2. Common Loss Functions.

Loss Function	Formula	Description
Mean Squared Error (MSE)	$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$	Measures the average squared difference between the predicted values $\hat{y}_i$ and the actual values y_i. Penalizes large errors more heavily.
Cross-Entropy Loss	$CE = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log \left(\hat{y}_i \right) + \left(1 - y_i \right) \log (1 - \hat{y}_i) \right]$	Used for binary classification. Compares the predicted probabilities $\hat{y}_i$ of the binary classes with the true binary labels y_i .
Log Loss	$Log\,Loss = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \, log \left(\hat{y}_i \right) \right]$	Also known as Binary Cross-Entropy, it quantifies the difference between the predicted probabilities $\hat{y}_i$ and the true binary labels y_i .

Cont.

Loss Function	Formula	Description
Hinge Loss	$Hinge = \max(0,\ 1 - y_i \bullet \hat{y}_i)$	Often used for binary classification with Support Vector Machines (SVMs). Penalizes predictions less than one, encouraging correct classification with a margin.
Dice Loss	$Dice = \frac{2 \times Intersection\left(y,\ \hat{y}\right)}{Union\left(y,\ \hat{y}\right) + \epsilon}$	Particularly used for tasks like image segmentation. Measures the overlap between predicted $\hat{y}$ and true y masks. It encourages higher overlap by penalizing lower scores.

2.3. Back Propagation

Backpropagation is an optimization algorithm used to train neural networks. Its basic idea is to compute the gradient of the loss function with respect to the network parameters, and then use gradient descent or other optimization algorithms to update the network parameters, thereby minimizing the loss function. Firstly, the neural network's output is computed through forward propagation, and the value of the loss function is calculated. Then, starting from the output layer, the impact of each parameter on the loss function is calculated layer by layer using the chain rule, and the error signal is propagated back to the input layer. Next, based on the gradient of the loss function with respect to the parameters, gradient descent or other optimization algorithms are used to update the network parameters. Finally, the process of forward propagation, backpropagation, and parameter updating is repeated until a preset stopping condition is met.

2.4. Optimizer

Optimizer is a tool in deep learning used to update model parameters to minimize the loss function. It calculates the gradient of the loss function with respect to the parameters and adjusts the parameter values using gradient information, continuously optimizing the model's performance. The table below contains several common optimizers:

·Gradient Descent: The most fundamental optimization algorithm, which adjusts the parameters based on the gradient direction of the loss function, gradually reducing the loss function. Common gradient descent algorithms include Batch Gradient Descent, Stochastic Gradient Descent, and Mini-batch Gradient Descent.

·Momentum Optimizer: Momentum optimizer is based on the exponential weighted average of gradients, which can accelerate model convergence and reduce oscillations. It introduces a momentum term to track the historical information of gradients, thereby considering the previous update directions when updating parameters.

·AdaGrad: Adjusting the learning rate based on the accumulated historical gradients of parameters, using a smaller learning rate for frequently occurring parameters and a larger learning rate for infrequently occurring parameters, thereby optimizing the model parameters more effectively.

·RMSProp: RMSProp is an improved version of AdaGrad, which introduces a decay coefficient to smooth the accumulated historical gradients, thereby avoiding the issue of the learning rate decreasing too quickly.

·Adam (Adaptive Moment Estimation): Adam combines the advantages of both momentum optimization and RMSProp by simultaneously considering the first and second moment estimates of gradients and correcting them. Adam is generally considered one of the most outstanding optimization algorithms and is widely used in deep learning.

·AdaDelta: It's a further improvement over AdaGrad, enhancing performance by dynamically adjusting the learning rate without the need for manually setting a global learning rate.

·Nadam: It's a variant of Adam that combines Nesterov momentum and the Adam optimization algorithm, using Nesterov momentum to accelerate convergence while leveraging Adam's adaptive learning rate adjustment mechanism.

3. Common CNN models

3.1. LeNet

LeNet [29] is the first successful Convolutional Neural Network (CNN) proposed by Yann LeCun in 1998. Its design inspiration comes from the understanding of the visual cortex in biology. LeNet was initially used to solve handwritten digit recognition problems and achieved success in fields such as postal code recognition and bank check reading. LeNet-5 is a classic CNN architecture, where the combination of convolutional layers, pooling layers, and fully connected layers remains a fundamental component of modern deep CNNs.

The LeNet-5 architecture consists of seven layers, containing about 60k parameters, as shown in Figure 3 [29], and trained on the MNIST dataset. It takes input images of size 32×32. The first layer, C1, and the third layer, C3, are convolutional layers. In the C1 layer, six convolutional kernels are computed, each with fixed weights when convolving with the previous layer. When the input is a single-channel signal, the C1 layer contains six convolutional kernels of size 1×5×5. Considering bias, the C1 layer contains a total of (6×3×5×5 + 6) = 156 parameters. The second layer, S2, and the fourth layer, S4, are subsampling (pooling) layers, where max-pooling layers mainly use spatial windows of size 2×2 and stride 2 for convolution in LeNet-5. The fifth layer, C5, and the sixth layer, F6, are fully connected layers, each with a fixed number of neurons, 120 and 84 respectively. The seventh layer is the output layer, and the outputs are classified through a softmax layer.

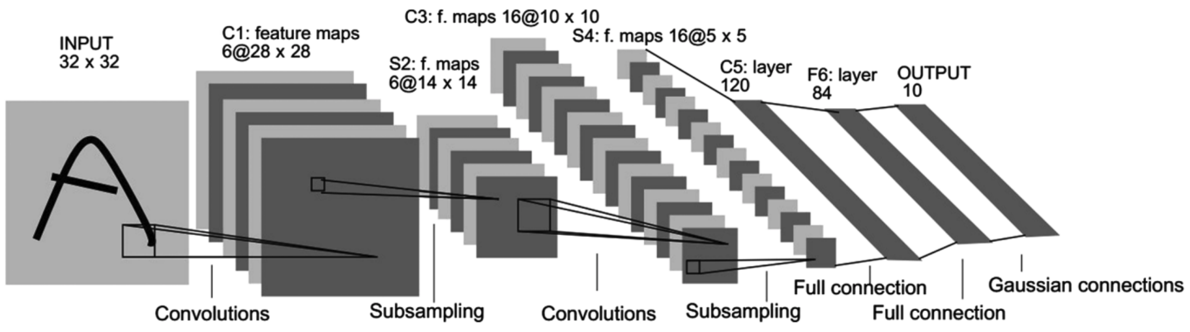


Figure 3. LeNet-5 model diagram [29].

3.2. AlexNet

In 2012, AlexNet [30], constructed by Krizhevsky et al., was introduced. This network won the ILSVR competition with a significant advantage, becoming the first CNN winner of the event. The training of this network utilized two graphics processing units (GPUs), as illustrated in Figure 4 [31], with one GPU dedicated to the upper part and the other to the lower part.

AlexNet consists of 8 layers, containing approximately 60 million parameters. It takes RGB images of size 256×256×3 as input and extracts 224×224 patches for training the first convolutional layer. The first convolutional layer consists of 96 filters of size 11×11×3, and the second convolutional layer, processing the output of the previous layer, has 256 filters of size 5×5×48. The third layer employs 384 filters of size 3×3×256, which are connected to the fourth layer with 384 filters of size 3×3×192. The fifth layer has 256 filters of size 3×3×192. The sixth, seventh, and eighth layers are fully connected layers. The last layer is the output layer, containing 1000 nodes.

Key aspects of AlexNet include the use of dropout and data augmentation to combat overfitting; switching the activation function from sigmoid to ReLU, which accelerated model convergence and reduced gradient vanishing; and having overlapping regions between adjacent pooling windows, which can improve model accuracy and alleviate overfitting.

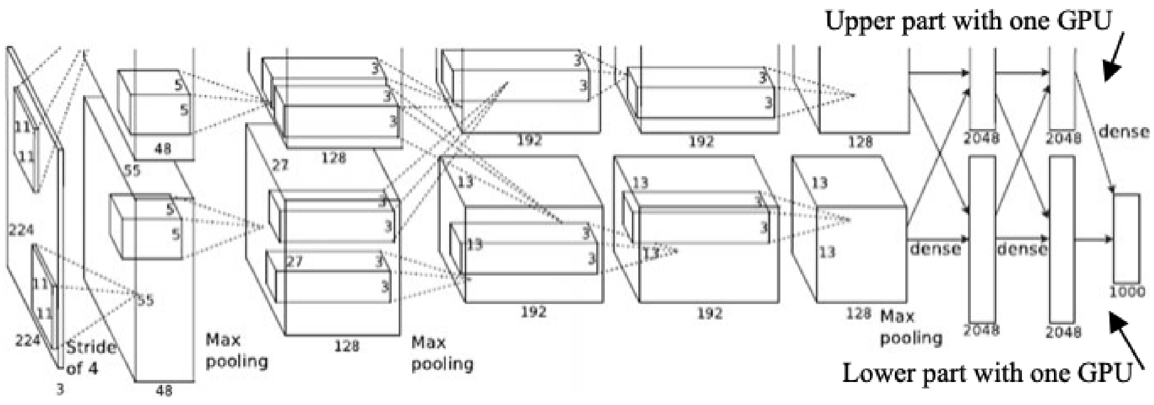


Figure 4. An illustration of the architecture of AlexNet [31].

3.3. VGGNet

In 2014, Simonyan et al. proposed the VGG model [32], which achieved the runner-up position in ILSVR 2014. The distinguishing feature of the VGG model is the use of many basic modules to construct the model, involving the consecutive use of several identical convolutional layers followed by a max-pooling layer. The convolutional layers maintain the height and width of the input unchanged, while the pooling layers halve them. Additionally, VGGNet employs many 3×3 convolutional filters, ensuring an increase in network depth and a reduction in model parameters compared to larger convolutional filters at the same receptive field. Finally, it first scales the input images to different sizes (from 256 to 512), then randomly crops them to a fixed size of 224×224, and trains the data obtained from multiple windows together. This process is considered as scale jittering, which can achieve the effect of data augmentation and prevent model overfitting. VGG network has various layer structure models, such as VGG-16 (as shown in Figure 5 [26]).

VGGNet-16 consists of 13 convolutional layers with filter sizes of 3×3, ranging from 64 to 512 filters of different numbers. There are five pooling layers, all utilizing max-pooling, followed by three fully connected layers with 4096 nodes each, and finally an output layer containing 1000 nodes. ReLU activation function is used in every layer. In the convolutional layers, 3×3 filters are applied with a stride of 1. In places where max pooling is performed, a window size of 2×2 and a stride of 2 are used.

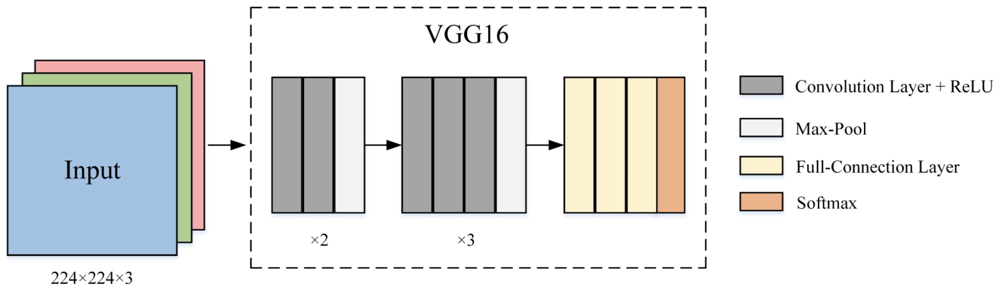


Figure 5. VGG-16 network architecture [26].

3.4. GoogLeNet

GoogLeNet [33] is a deep convolutional neural network structure proposed by Google in 2014. Its core idea is to extract image features through multi-scale convolution operations, and it introduces the Inception module inside the network as shown in Figure 6. The Inception module utilizes multiple convolutional kernels of different sizes to process the input, and then concatenates the results together: it reduces the number of input feature channels using 1x1 convolutional kernels to reduce computational complexity, and then performs 3x3 and 5x5 convolutional operations. This approach can capture features of different scales, greatly increasing the width and depth of the network while reducing the number of parameters.

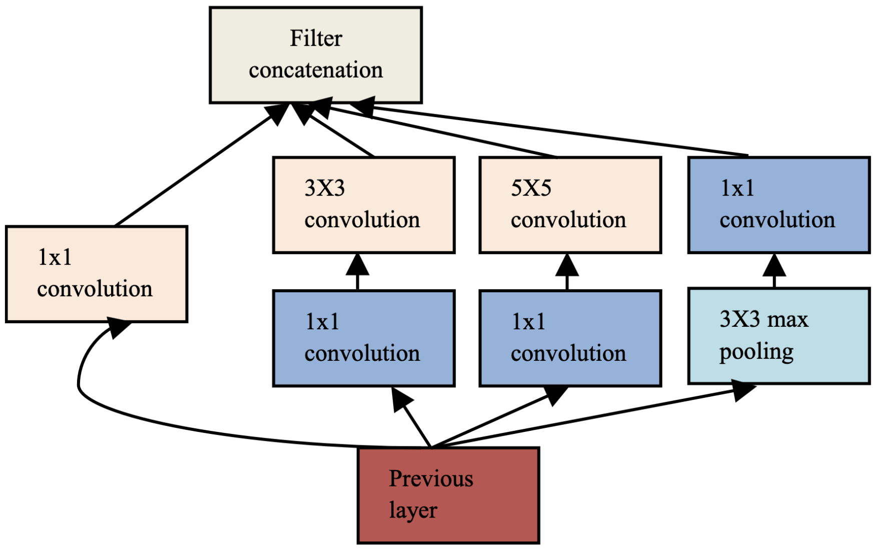


Figure 6. Inception building block used in GoogLeNet [33].

This network consists of 9 linearly stacked Inception modules, totaling 27 layers, with 5 of them being pooling layers. The total number of layers for the network structure is approximately 100 layers. In GoogLeNet, the authors noticed that using average pooling layers instead of fully connected layers can improve accuracy and enhance the model's generalization performance. Even with the removal of fully connected layers, dropout is still necessary to combat overfitting. Additionally, auxiliary classifiers are used to alleviate the vanishing gradient problem, and extra loss functions are introduced during training.

Google also released subsequent versions of the Inception network, namely Inception-v2, Inception-v3, and Inception-v4. Inception-v2 improved upon the original GoogLeNet by introducing the Batch Normalization technique, which applies BN layers to the network, speeding up convergence and reducing sensitivity to learning rates. Inception-v3 further enhanced the network structure based on Inception-v2, introducing more techniques such as decomposed convolution and convolutions with dimensionality reductions, thereby improving the efficiency and accuracy of the network. Additionally, the newly added auxiliary classifiers in Inception-v3 differ from Inception-v1; they are connected to intermediate layers instead of directly to the network's output, aiding gradient propagation throughout the network. Inception-v4 inherits the characteristics of the Inception module and introduces residual connections, a deeper network structure, and advanced optimization and regularization techniques. By improving the design and optimization of the Inception module internals and introducing technologies like residual connections, Inception-v4 achieves significant improvements in both the performance and efficiency of the network. Figure 7 [26] illustrates the overall architecture of InceptionV4.

3.5. ResNet

The Residual Network [34], proposed by Kaiming He et al. in 2015, achieved first place in the ILSVRC 2015 image classification competition. ResNet is a type of deep residual network structure that addresses the issues of gradient vanishing and degradation in training deep neural networks by introducing residual connections. These connections directly add the input to the output, propagating residuals to the deeper layers of the network. The ResNet is primarily composed of the residual learning block, as shown in Figure 8 [34].

The design philosophy of ResNet is to focus the neural network on learning the residuals (i.e., differences) between the input and the target output, rather than directly learning the target output itself. In traditional neural networks, the main task of the network is to attempt to learn to map the input to the target output. This approach may lead to problems such as gradient vanishing or explosion during training, especially when the network becomes very deep. The introduction of ResNet aims to address these issues.

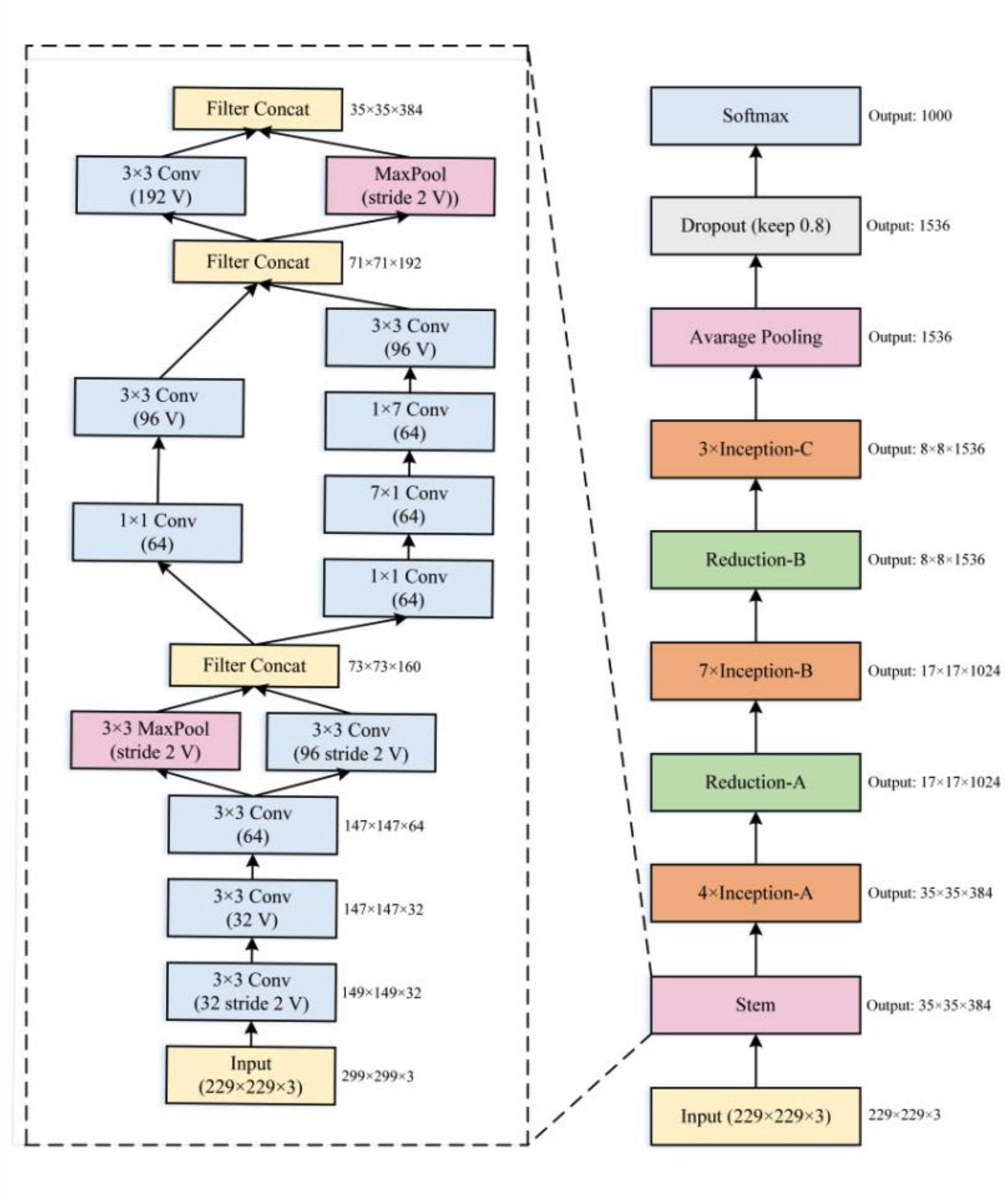


Figure 7. Overall Architecture of InceptionV4 [26].

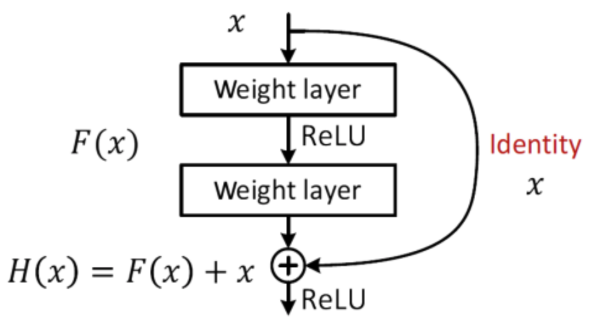
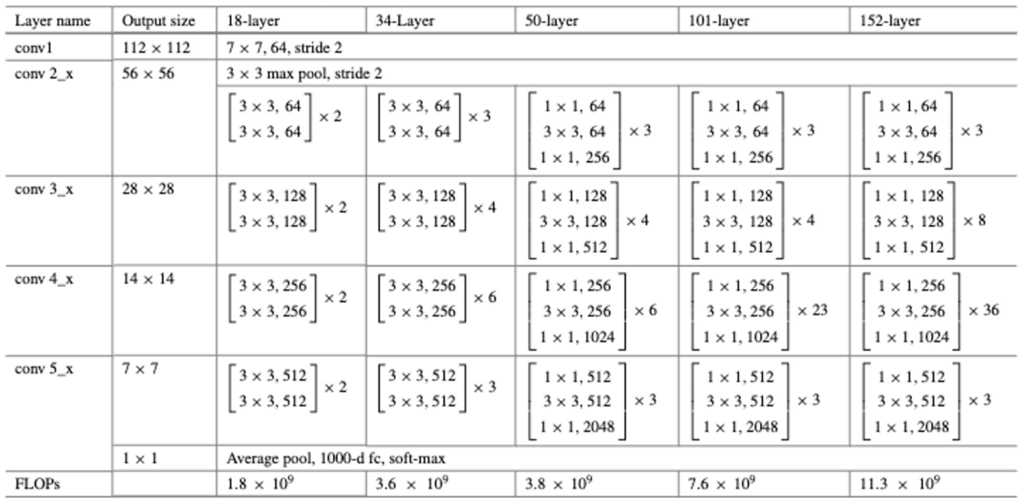


Figure 8. Residual learning block [34].

Specifically, each residual block in ResNet contains a residual mapping, which learns the difference (residual) between the input and its corresponding expected output as the learning target. In this way, the network can more easily learn the identity mapping, which directly maps the input to the target output, thus preserving more information from the original input. This design enables the network to train more effectively and helps to address gradient issues in training deep networks because the network only needs to learn the residuals rather than the complete mapping relationship.

The ResNet-152 architecture consists of 152 layers, utilizing 3×3 and 1×1 filters, and adopts residual learning, as shown in Figure 9. Kaiming He et al. set the number of layers for ResNet to be 18, 34, 50, 101, and 152. In the architectures with 18 and 34 layers, only 3×3 filters are used, while in the architectures with 50, 101, and 152 layers, both 3×3 and 1×1 filters are used. As the number of layers increases, the depth, complexity, FLOPs (floating point operations per second), and accuracy all increase, while the training and validation errors decrease significantly.



Layer name	Output size	18-layer	34-Layer	50-layer	101-layer	152-layer
conv1	112 × 112	7 × 7, 64, stride 2				
conv 2_x	56 × 56	3 × 3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv 3_x	28 × 28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv 4_x	14 × 14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv 5_x	7 × 7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1 × 1	Average pool, 1000-d fc, soft-max				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 9. ResNet architectures vary in depth and computational cost [34].

Pre-activation ResNet [35] is an improved version of ResNet, proposed by Kaiming He et al. in 2016. It is optimized based on the traditional ResNet architecture, introducing a pre-activation structure where the order of batch normalization (BN) and ReLU activation function is swapped, making the network easier to train. In Pre-activation ResNet, the structure of each residual block is transformed into BN-ReLU-Conv, Conv-BN-ReLU-Conv. By placing BN and ReLU activation functions before the convolutional layer, the network can better propagate gradients, alleviating the issues of gradient vanishing and exploding. Additionally, Pre-activation ResNet adopts deeper network structures and utilizes more advanced optimization and regularization techniques, such as the Adam optimizer and Dropout, further improving the performance and generalization ability of the network.

3.6. SENet

SENet (Squeeze-and-Excitation Networks) [36] is a convolutional neural network architecture that incorporates attention mechanisms, proposed by Jie Hu et al. in 2017. The main contribution of SENet is the introduction of attention mechanisms into ordinary convolutional neural networks, enabling the network to adaptively adjust the weights of each feature map to capture important feature information more effectively. The structure of SENet includes two key modules: the Squeeze module and the Excitation module. The Squeeze module is responsible for globally compressing each feature map to transform it into a single feature descriptor, while the Excitation module learns the importance weights of each feature map channel and weights the feature maps. Specifically, a transformation $F_{tr}: X \rightarrow U$ is first performed, then in the squeezing process (F_{tr}), the transformed features are passed to the squeeze process, which collaborates the feature maps and produces channel descriptors, generating embeddings of distributed channel feature responses with dimensions of 1×1×C, as illustrated in Figure 10 [36].

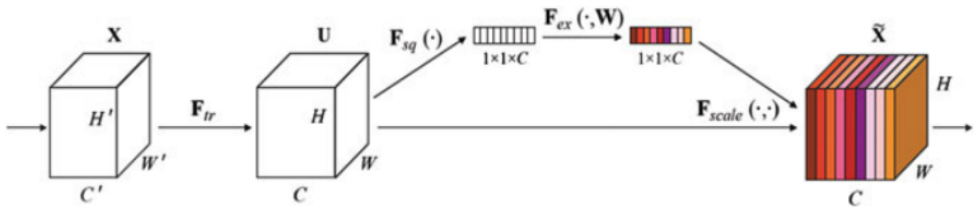


Figure 10. Squeeze-and-excitation network [36].

3.7. EfficientNet

EfficientNet is a series of efficient convolutional neural network architectures proposed by the Google Brain team in 2019, aiming to improve the performance and efficiency of networks by simultaneously adjusting the width, depth, and resolution of the network. The design inspiration for EfficientNet comes from the model scaling theory, which suggests that scaling the model across different dimensions can effectively improve its performance.

In 2019, Tan et al. [37] proposed EfficientNetV1 by combining network structures of different scales to construct an efficient model. It introduces a compound coefficient φ to balance the width (w), depth (d), and resolution (r) of the network, thus improving performance while maintaining efficiency. This model scaling approach uses φ to uniformly scale the network's width, depth, and resolution, as shown below [37]:

$$\begin{gathered} depth: d = \alpha^{\phi} \\ width: w = \beta^{\phi} \\ resolution: r = \gamma^{\phi} \\ s.t \alpha \bullet \beta^2 \bullet \gamma^2 \approx 2 \\ \alpha \geq 1,\ \beta \geq 1,\ \gamma \geq 1 \end{gathered}$$

Where α, β, and γ are constants that can be determined through a small grid search. φ is a user-specified coefficient used to control how much resources are available for model scaling.

EfficientNet v2 [38] is an improved version of EfficientNet v1, proposed in 2020. It further enhances the performance and speed of the network by optimizing the network architecture and training strategies. EfficientNet v2 adopts a lighter network structure and more efficient training methods, including the use of smaller convolutional kernels, fewer parameters, and faster training strategies, thereby further improving the model's performance and speed while maintaining efficiency. Additionally, EfficientNet v2 introduces a new training method called "Stochastic Depth," which enhances training robustness and generalization by randomly dropping some layers in the network, further boosting the network's performance.

4. Performance

The following Table 3 presents a performance comparison of 7 classic convolutional neural network (CNN) models across several widely used datasets for computer vision tasks.

Table 3. Model Performance Comparison on Benchmark Datasets.

Model	MNIST	CIFAR-10	CIFAR-100	ImageNet (Top-1/5 Err%)	COCO (mAP)
LeNet	99.2%	~90%	~65%	~40% / ~20%	~50%
AlexNet	99.79%	90.76%	~75%	42.6% / 19.6%	~60%
VGGNet	99.8%	92.64%	73.36%	27.3% / 9.0%	~65%
GoogLeNet	99.7%	95.98%	78.09%	27.9% / 9.15%	~68%
ResNet	99.8%	95.83%	79.29%	23.6% / 6.71%	37.7%
SENet	99.8%	~96%	82.55%	22.68% / 6.07%	40.9%

Cont.

Model	MNIST	CIFAR-10	CIFAR-100	ImageNet (Top-1/5 Err%)	COCO (mAP)
EfficientNet	99.8%	~97%	91.7%	23.7% / ~6%	48.0%

MNIST: A handwritten digit recognition dataset. The values indicate the classification accuracy (%) achieved by each model. All the modern CNN models from AlexNet onwards achieve very high accuracy around 99.7-99.8% on this relatively simple dataset. The pioneering LeNet model also performed well with 99.2% accuracy.

CIFAR-10/100: Object classification datasets. Again, the values represent the classification accuracy (%) on these datasets. There is a clear progression, with each new model architecture outperforming the previous ones on these datasets. EfficientNet shows the highest estimated accuracy of around 97% on CIFAR-10 and 91.7% on the more challenging CIFAR-100. The earliest AlexNet model lags with around 90.76% on CIFAR-10.

ImageNet: A large-scale object classification dataset. The two values are the Top-1 and Top-5 error rates (%), which are common evaluation metrics. Again, a pattern of steadily decreasing top-1 and top-5 error rates is observed from AlexNet to EfficientNet. The top-5 error rates have dropped from around 19.6% for AlexNet to an estimated 6% for EfficientNet. ResNet, SENet, and EfficientNet represent the current state-of-the-art on this very competitive benchmark.

COCO: An object detection and instance segmentation dataset. The values denote the mean Average Precision (mAP) score, a standard metric for object detection tasks. Only a few models have reported results on this complex task. EfficientNet currently has the highest mAP of 48%, followed by SENet at 40.9% and ResNet at 37.7%. The performance jump from ResNet to EfficientNet is quite significant.

5. Conclusion

With the development of computer vision, significant progress has been made in image classification technology driven by deep learning. Traditional methods relied on handcrafted feature extraction techniques and conventional machine learning algorithms. However, with the rise of deep learning, particularly the introduction of Convolutional Neural Networks (CNNs), there has been a tremendous breakthrough in image classification.

This article starts by introducing the basic structure and principles of classical neural networks, and then compares and analyzes the advantages and disadvantages of various convolutional network architectures. By providing an overview of key layers and techniques such as convolutional layers, pooling layers, activation functions, fully connected layers, Dropout, loss functions, backpropagation, and optimizers, readers can better understand the working principles and training processes of CNNs. Additionally, by briefly introducing several common CNN models such as LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, SENet, and EfficientNet, readers can learn about the characteristics and applications of different models.

In summary, CNNs, as an important technology in the field of image classification, have been widely applied in practice and have achieved state-of-the-art performance in many tasks [39–44]. With the continuous development and improvement of deep learning technology, it is believed that CNNs will have broader application prospects in the field of image classification.

Funding

Not applicable.

Author Contributions

Conceptualization, X.P. and Z.L.; writ-ing—original draft preparation and writing—review and editing, X.P., Z. L. and L.Z. All authors have read and agreed to the published version of the manuscript.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

Not applicable.

Conflicts of Interest

The authors declare no conflict of interest.

References

1 LeCun Y, Boser B, Denker J, Henderson D, Howard R, Hubbard W, Jackel L. Handwritten Digit Recognition with a Back–Propagation Network. *Advances in Neural Information Processing Systems* 1989; **2**: 396–404.

2 Liu Y, Liu L, Yang L, Hao L, Bao Y. Measuring Distance Using Ultra – Wideband Radio Technology Enhanced by Extreme Gradient Boosting Decision Tree (XGBoost). *Automation in Construction* 2021; **126**: 103678.

3 Liu Y, Bao Y. Review of Electromagnetic Waves – Based Distance Measurement Technologies for Remote Monitoring of Civil Engineering Structures. *Measurement* 2021; **176**: 109193.

4 Shen Y, Gu H – M, Qin S, Zhang D – W. Surf4, Cargo Trafficking, Lipid Metabolism, and Therapeutic Implications. *Journal of Molecular Cell Biology* 2022; **14**(**9**): mjac063.

5 Wang M, Alabi A, Gu H – M, Gill G, Zhang Z, Jarad S, Xia X – D, Shen Y, Wang G – Q, Zhang D – W. Identification of Amino Acid Residues in the MT–Loop of MT1–MMP Critical for Its Ability to Cleave Low –Density Lipoprotein Receptor. *Frontiers in Cardiovascular Medicine* 2022; **9**: 917238.

6 Shen Y, Gu H – m, Zhai L, Wang B, Qin S, Zhang D – w. The Role of Hepatic Surf4 in Lipoprotein Metabolism and the Development of Atherosclerosis in ApoE−/−Mice. *Biochimica et Biophysica Acta (BBA) –Molecular and Cell Biology of Lipids* 2022; **1867**(**10**): 159196.

7 Qiu Y. Estimation of Tail Risk Measures in Finance: Approaches to Extreme Value Mixture Modeling. Master's Thesis, Johns Hopkins University: Baltimore, MD, USA, May 2019.

8 Milord J, Yu F, Orton S, Flores L, Marra R. Impact of COVID Transition to Remote Learning on Engineering Self – Efficacy and Outcome Expectations. In Proceedings of the 2021 ASEE Virtual Annual Conference, Virtual Conference, 26–29 July 2021.

9 Liu Y, Hajj M, Bao Y. Review of Robot – Based Damage Assessment for Offshore Wind Turbines. *Renewable and Sustainable Energy Reviews* 2022; **158**: 112187.

10 Liu Y, Bao Y. Review on Automated Condition Assessment of Pipelines with Machine Learning. *Advanced Engineering Informatics* 2022; **53**: 101687.

11 Deng X, Dong Z, Ma X, Wu H, Wang B, Du X. Exploration on Mechanics Design for Scanning Tunneling Microscope. In Proceedings of the 2009 Symposium on Photonics and Optoelectronics, Wuhan, China, 14–16 August 2009.

12 Deng X, Hu Z, Xiu G, Li D, Yue Y, Song Z, Weng ZK, Xu J, Wang Z. Five – Beam Interference Pattern Model for Laser Interference Lithography. In Proceedings of the The 2010 IEEE International Conference on Information and Automation, Harbin, China, 20–23 June 2010.

13 Wang B, Shen Y, Zhai L, Xia X, Gu H-M, Wang M, Zhao Y, Chang X, Alabi A, Xing S, Deng S, Liu B, Wang G, Qin S, Zhang D-W. Atherosclerosis – Associated Hepatic Secretion of VLDL but Not PCSK9 Is Dependent on Cargo Receptor Protein Surf4. *Journal of Lipid Research* 2021; **62**: 100091.

14 Deng S–j, Shen Y, Gu H–M, Guo S, Wu S–R, Zhang D–w. The Role of the C–Terminal Domain of PCSK9

and SEC24 Isoforms in PCSK9 Secretion. *Biochimica et Biophysica Acta (BBA) – Molecular and Cell Biology of Lipids* 2020; **1865**(**6**): 158660.

15 Shen Y, Wang B, Deng S, Zhai L, Gu H-M, Alabi A, Xia X, Zhao Y, Chang X, Qin S, Zhang D-W. Surf4 Regulates Expression of Proprotein Convertase Subtilisin/Kexin Type 9 (PCSK9) but Is Not Required for PCSK9 Secretion in Cultured Human Hepatocytes. *Biochimica et Biophysica Acta (BBA)* – *Molecular and Cell Biology of Lipids* 2020; **1865**(**2**): 158555.

16 Mock MB, Zhang S, Pniak B, Belt N, Witherspoon M, Summers RM. Substrate Promiscuity of the NdmCDE N7–Demethylase Enzyme Complex. *Biotechnology Notes* 2021; **2**: 18–25.

17 Yu F, Milord J, Orton SL, Flores L, Marra R. The Concerns and Perceived Challenges Students Faced When Traditional in – Person Engineering Courses Suddenly Transitioned to Remote Learning. In Proceedings of the 2022 ASEE Annual Conference, Minneapolis, MN, USA, 26–29 June 2022.

18 Deng X, Li L, Enomoto M, Kawano Y. Continuously Frequency – Tuneable Plasmonic Structures for Terahertz Bio–Sensing and Spectroscopy. *Scientific Reports* 2019; **9**(**1**): 3498.

19 Deng X, Simanullang M, Kawano Y. Ge – Core/a – Si – Shell Nanowire – Based Field – Effect Transistor for Sensitive Terahertz Detection. *Photonics* 2018; **5**(**2**): 13.

20 Deng X, Kawano Y. Surface Plasmon Polariton Graphene Midinfrared Photodetector with Multifrequency Resonance. *Journal of Nanophotonics* 2018; **12**(**2**): 026017–026017.

21 Xia D, Alexander AK, Isbell A, Zhang S, Ou J, Liu XM. Establishing a Co–Culture System for Clostridium Cellulovorans and Clostridium Aceticum for High Efficiency Biomass Transformation. *J Sci Heal Univ Ala* 2017; **14**: 8–13.

22 Kundu S, Fu Y, Ye B, Beerel PA, Pedram M. Toward Adversary – Aware Non – Iterative Model Pruning Through D Ynamic N Etwork R Ewiring of DNNs. *ACM Transactions on Embedded Computing Systems* 2022; **21**(**5**): 1–24.

23 Sun G, Zhan T, Owusu BG, Daniel A – M, Liu G, Jiang W. Revised Reinforcement Learning Based on Anchor Graph Hashing for Autonomous Cell Activation in Cloud – RANs. *Future Generation Computer Systems* 2020; **104**: 60–73.

24 Li S, Singh K, Riedel N, Yu F, Jahnke I. Digital Learning Experience Design and Research of a Self–Paced Online Course for Risk–Based Inspection of Food Imports. *Food Control* 2022; **135**: 108698.

25 Horne J, Beddingfield E, Knapp M, Mitchell S, Crawford L, Mills SB, Wrist A, Zhang S, Summers RM. Caffeine and Theophylline Inhibit β–Galactosidase Activity and Reduce Expression in Escherichia coli. *ACS Omega* 2020; **5**(**50**): 32250–32255.

26 Chen L, Li S, Bai Q, Yang J, Jiang S, Miao Y. Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sensing* 2021; **13**(**22**): 4712.

27 Wang W, Yang Y, Wang X, Wang W, Li J. Development of Convolutional Neural Network and Its Application in Image Classification: a Survey. *Optical Engineering* 2019; **58**(**4**): 040901–040901.

28 Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning. 2016; arXiv:1603.07285.

29 LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient – Based Learning Applied to Document Recognition. *Proceedings of the IEEE* 1998; **86**(**11**): 2278–2324.

30 Krizhevsky A, Sutskever I, Hinton GE. Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems* 2017; **60**(**6**): 84–90.

31 Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural, NIPS ' 12. In Proceedings of the Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.

32 Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large–Scale Image Recognition. 2014. arXiv:1409.1556.

33 Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going Deeper with Convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.

34 He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.

35 He K, Zhang X, Ren S, Sun J. Identity Mappings in Deep Residual Networks. In Proceedings of the Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11 – 14 October 2016.

36 Hu J, Shen L, Sun G. Squeeze – and – Excitation Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.

37 Tan M, Le Q. Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the ICML 2019: 36th International Conference on Machine Learning, Long Beach, CA, USA, 10 – 15 June 2019.

38 Tan M, Le Q. Efficientnetv2: Smaller Models and Faster Training. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Virtual Event, 18–24 July 2021.

39 Luo Z, Xu H, Chen F. Utterance – Based Audio Sentiment Analysis Learned by a Parallel Combination of CNN and LSTM. 2018. arXiv:1811.08065.

40 Chen F, Luo Z. Learning Robust Heterogeneous Signal Features from Parallel Neural Network for Audio Sentiment Analysis. 2018. arXiv:1811.08065.

41 Chen F, Luo Z. Sentiment Analysis Using Deep Robust Complementary Fusion of Multi–Features and Multi –Modalities. 2019. CoRR abs/1904.08138.

42 Luo Z, Xu H, Chen F. Audio Sentiment Analysis by Heterogeneous Signal Features Learned from Utterance – Based Parallel Neural Network. In Proceedings of the AffCon@ AAAI 2019, Honolulu, HI, USA, 27 January 2019..

43 Luo Z, Zeng X, Bao Z, Xu M. Deep Learning – Based Strategy for Macromolecules Classification with Imbalanced Data from Cellular Electron Cryotomography. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019.

44 Chen F, Luo Z, Xu Y, Ke D. Complementary Fusion of Multi–Features and Multi–Modalities in Sentiment Analysis. 2019. arXiv:1904.08138.