Alzubaidi *et al. J Big Data* (2021) 8:53
https://doi.org/10.1186/s40537-021-00444-8

Journal of Big Data

SURVEY PAPER Open Access

Review of deep learning: concepts, CNN architectures, challenges, applications, future directions

Laith Alzubaidi[1,5*], Jinglan Zhang[1], Amjad J. Humaidi[2], Ayad Al-Dujaili[3], Ye Duan[4], Omran Al-Shamma[5], J. Santamaría[6], Mohammed A. Fadhel[7], Muthana Al-Amidie[4] and Laith Farhan[8]

*Correspondence: laith.alzubaidi@hdr.qut.edu.au
[1] School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia
Full list of author information is available at the end of the article

Abstract

In the last few years, the deep learning (DL) computing paradigm has been deemed the Gold Standard in the machine learning (ML) community. Moreover, it has gradually become the most widely used computational approach in the field of ML, thus achieving outstanding results on several complex cognitive tasks, matching or even beating those provided by human performance. One of the benefits of DL is the ability to learn massive amounts of data. The DL field has grown fast in the last few years and it has been extensively used to successfully address a wide range of traditional applications. More importantly, DL has outperformed well-known ML techniques in many domains, e.g., cybersecurity, natural language processing, bioinformatics, robotics and control, and medical information processing, among many others. Despite it has been contributed several works reviewing the State-of-the-Art on DL, all of them only tackled one aspect of the DL, which leads to an overall lack of knowledge about it. Therefore, in this contribution, we propose using a more holistic approach in order to provide a more suitable starting point from which to develop a full understanding of DL. Specifically, this review attempts to provide a more comprehensive survey of the most important aspects of DL and including those enhancements recently added to the field. In particular, this paper outlines the importance of DL, presents the types of DL techniques and networks. It then presents convolutional neural networks (CNNs) which the most utilized DL network type and describes the development of CNNs architectures together with their main features, e.g., starting with the AlexNet network and closing with the High-Resolution network (HR.Net). Finally, we further present the challenges and suggested solutions to help researchers understand the existing research gaps. It is followed by a list of the major DL applications. Computational tools including FPGA, GPU, and CPU are summarized along with a description of their influence on DL. The paper ends with the evolution matrix, benchmark datasets, and summary and conclusion.

Keywords: Deep learning, Machine learning, Convolution neural network (CNN), Deep neural network architectures, Deep learning applications, Image classification, Transfer learning, Medical image analysis, Supervised learning, FPGA, GPU

Springer Open

© The Author(s) 2021. This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Introduction

Recently, machine learning (ML) has become very widespread in research and has been incorporated in a variety of applications, including text mining, spam detection, video recommendation, image classification, and multimedia concept retrieval [1–6]. Among the different ML algorithms, deep learning (DL) is very commonly employed in these applications [7–9]. Another name for DL is representation learning (RL). The continuing appearance of novel studies in the fields of deep and distributed learning is due to both the unpredictable growth in the ability to obtain data and the amazing progress made in the hardware technologies, e.g. High Performance Computing (HPC) [10].

DL is derived from the conventional neural network but considerably outperforms its predecessors. Moreover, DL employs transformations and graph technologies simultaneously in order to build up multi-layer learning models. The most recently developed DL techniques have obtained good outstanding performance across a variety of applications, including audio and speech processing, visual data processing, natural language processing (NLP), among others [11–14].

Usually, the effectiveness of an ML algorithm is highly dependent on the integrity of the input-data representation. It has been shown that a suitable data representation provides an improved performance when compared to a poor data representation. Thus, a significant research trend in ML for many years has been feature engineering, which has informed numerous research studies. This approach aims at constructing features from raw data. In addition, it is extremely field-specific and frequently requires sizable human effort. For instance, several types of features were introduced and compared in the computer vision context, such as, histogram of oriented gradients (HOG) [15], scale-invariant feature transform (SIFT) [16], and bag of words (BoW) [17]. As soon as a novel feature is introduced and is found to perform well, it becomes a new research direction that is pursued over multiple decades.

Relatively speaking, feature extraction is achieved in an automatic way throughout the DL algorithms. This encourages researchers to extract discriminative features using the smallest possible amount of human effort and field knowledge [18]. These algorithms have a multi-layer data representation architecture, in which the first layers extract the low-level features while the last layers extract the high-level features. Note that artificial intelligence (AI) originally inspired this type of architecture, which simulates the process that occurs in core sensorial regions within the human brain. Using different scenes, the human brain can automatically extract data representation. More specifically, the output of this process is the classified objects, while the received scene information represents the input. This process simulates the working methodology of the human brain. Thus, it emphasizes the main benefit of DL.

In the field of ML, DL, due to its considerable success, is currently one of the most prominent research trends. In this paper, an overview of DL is presented that adopts various perspectives such as the main concepts, architectures, challenges, applications, computational tools and evolution matrix. Convolutional neural network (CNN) is one of the most popular and used of DL networks [19, 20]. Because of CNN, DL is very popular nowadays. The main advantage of CNN compared to its predecessors is that it automatically detects the significant features without any human supervision which made it the most used. Therefore, we have dug in deep with CNN by presenting the main

components of it. Furthermore, we have elaborated in detail the most common CNN architectures, starting with the AlexNet network and ending with the High-Resolution network (HR.Net).

Several published DL review papers have been presented in the last few years. However, all of them have only been addressed one side focusing on one application or topic such as the review of CNN architectures [21], DL for classification of plant diseases [22], DL for object detection [23], DL applications in medical image analysis [24], and etc. Although these reviews present good topics, they do not provide a full understanding of DL topics such as concepts, detailed research gaps, computational tools, and DL applications. First, It is required to understand DL aspects including concepts, challenges, and applications then going deep in the applications. To achieve that, it requires extensive time and a large number of research papers to learn about DL including research gaps and applications. Therefore, we propose a deep review of DL to provide a more suitable starting point from which to develop a full understanding of DL from one review paper. The motivation behinds our review was to cover the most important aspect of DL including open challenges, applications, and computational tools perspective. Furthermore, our review can be the first step towards other DL topics.

The main aim of this review is to present the most important aspects of DL to make it easy for researchers and students to have a clear image of DL from single review paper. This review will further advance DL research by helping people discover more about recent developments in the field. Researchers would be allowed to decide the more suitable direction of work to be taken in order to provide more accurate alternatives to the field. Our contributions are outlined as follows:

- This is the first review that almost provides a deep survey of the most important aspects of deep learning. This review helps researchers and students to have a good understanding from one paper.
- We explain CNN in deep which the most popular deep learning algorithm by describing the concepts, theory, and state-of-the-art architectures.
- We review current challenges (limitations) of Deep Learning including lack of training data, Imbalanced Data, Interpretability of data, Uncertainty scaling, Catastrophic forgetting, Model compression, Overfitting, Vanishing gradient problem, Exploding Gradient Problem, and Underspecification. We additionally discuss the proposed solutions tackling these issues.
- We provide an exhaustive list of medical imaging applications with deep learning by categorizing them based on the tasks by starting with classification and ending with registration.
- We discuss the computational approaches (CPU, GPU, FPGA) by comparing the influence of each tool on deep learning algorithms.

The rest of the paper is organized as follows: "Survey methodology" section describes The survey methodology. "Background" section presents the background. "Classification of DL approaches" section defines the classification of DL approaches. "Types of DL networks" section displays types of DL networks. "CNN architectures" section shows CNN Architectures. "Challenges (limitations) of deep learning and alternate solutions" section

details the challenges of DL and alternate solutions. "Applications of deep learning" section outlines the applications of DL. "Computational approaches" section explains the influence of computational approaches (CPU, GPU, FPGA) on DL. "Evaluation metrics" section presents the evaluation metrics. "Frameworks and datasets" section lists frameworks and datasets. "Summary and conclusion" section presents the summary and conclusion.

Survey methodology

We have reviewed the significant research papers in the field published during 2010–2020, mainly from the years of 2020 and 2019 with some papers from 2021. The main focus was papers from the most reputed publishers such as IEEE, Elsevier, MDPI, Nature, ACM, and Springer. Some papers have been selected from ArXiv. We have reviewed more than 300 papers on various DL topics. There are 108 papers from the year 2020, 76 papers from the year 2019, and 48 papers from the year 2018. This indicates that this review focused on the latest publications in the field of DL. The selected papers were analyzed and reviewed to (1) list and define the DL approaches and network types, (2) list and explain CNN architectures, (3) present the challenges of DL and suggest the alternate solutions, (4) assess the applications of DL, (5) assess computational approaches. The most keywords used for search criteria for this review paper are ("Deep Learning"), ("Machine Learning"), ("Convolution Neural Network"), ("Deep Learning" AND "Architectures"), (("Deep Learning") AND ("Image") AND ("detection" OR "classification" OR "segmentation" OR "Localization")), ("Deep Learning" AND "detection" OR "classification" OR "segmentation" OR "Localization"), ("Deep Learning" AND "CPU" OR "GPU" OR "FPGA"), ("Deep Learning" AND "Transfer Learning"), ("Deep Learning" AND "Imbalanced Data"), ("Deep Learning" AND "Interpretability of data"), ("Deep Learning" AND "Overfitting"), ("Deep Learning" AND "Underspecification"). Figure 1 shows our search structure of the survey paper. Table 1 presents the details of some of the journals that have been cited in this review paper.

Background

This section will present a background of DL. We begin with a quick introduction to DL, followed by the difference between DL and ML. We then show the situations that require DL. Finally, we present the reasons for applying DL.

DL, a subset of ML (Fig. 2), is inspired by the information processing patterns found in the human brain. DL does not require any human-designed rules to operate; rather, it uses a large amount of data to map the given input to specific labels. DL is designed using numerous layers of algorithms (artificial neural networks, or ANNs), each of which provides a different interpretation of the data that has been fed to them [18, 25].

Achieving the classification task using conventional ML techniques requires several sequential steps, specifically pre-processing, feature extraction, wise feature selection, learning, and classification. Furthermore, feature selection has a great impact on the performance of ML techniques. Biased feature selection may lead to incorrect discrimination between classes. Conversely, DL has the ability to automate the learning of feature sets for several tasks, unlike conventional ML methods [18, 26]. DL enables learning and classification to be achieved in a single shot (Fig. 3). DL has become an incredibly

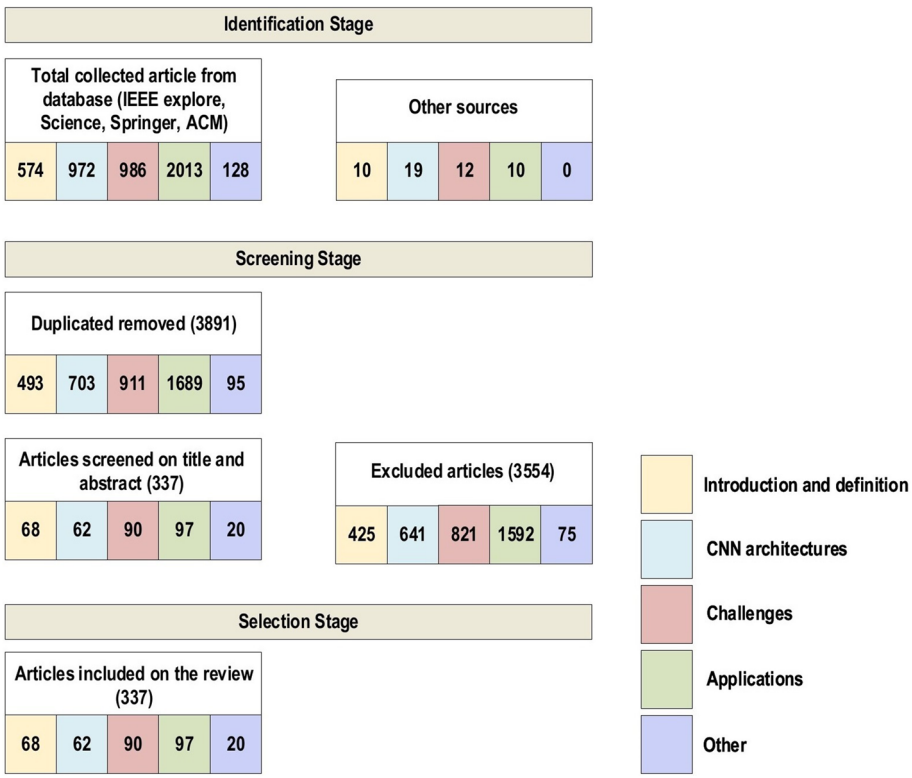

Fig. 1 Search framework

popular type of ML algorithm in recent years due to the huge growth and evolution of the field of big data [27, 28]. It is still in continuous development regarding novel performance for several ML tasks [22, 29–31] and has simplified the improvement of many learning fields [32, 33], such as image super-resolution [34], object detection [35, 36], and image recognition [30, 37]. Recently, DL performance has come to exceed human performance on tasks such as image classification (Fig. 4).

Nearly all scientific fields have felt the impact of this technology. Most industries and businesses have already been disrupted and transformed through the use of DL. The leading technology and economy-focused companies around the world are in a race to improve DL. Even now, human-level performance and capability cannot exceed that the performance of DL in many areas, such as predicting the time taken to make car deliveries, decisions to certify loan requests, and predicting movie ratings [38]. The winners of the 2019 "Nobel Prize" in computing, also known as the Turing Award, were three pioneers in the field of DL (Yann LeCun, Geoffrey Hinton, and Yoshua Bengio) [39]. Although a large number of goals have been achieved, there is further progress to be made in the DL context. In fact, DL has the ability to enhance human lives by providing additional accuracy in diagnosis, including estimating natural disasters [40], the discovery of new drugs [41], and cancer diagnosis [42–44]. Esteva et al. [45] found that a DL network has the same ability to diagnose the disease as twenty-one board-certified dermatologists using 129,450 images of 2032 diseases. Furthermore, in grading prostate

Table 1 Some of the journals have been cited in this review paper

Journal	IF 2019	CiteScore 2019	Publisher	Journal homepage
Pattern Recognition	7.196	13.1	Elsevier	https://www.journals.elsevier.com/pattern-recognition
Pattern Recognition Letter	3.255	6.3	Elsevier	https://www.journals.elsevier.com/pattern-recognition-letters
Artificial Intelligence Review	5.747	9.1	Springer	https://www.springer.com/journal/10462?referer=www.springeronline.com
Expert Systems with Applications	5.452	11	Elsevier	https://www.sciencedirect.com/journal/expert-systems-with-applications
Neurocomputing	4.438	9.5	Elsevier	https://www.journals.elsevier.com/neurocomputing
Nature Medicine	36.130	45.9	Nature	https://www.nature.com/nm/
Nature	42.779	51	Nature	https://www.nature.com/
Journal of Big Data	–	6.1	Springer	https://journalofbigdata.springeropen.com/
Multimedia Tools and Applications	2.313	3.7	Springer	https://www.springer.com/journal/11042
Computer Methods and Programs in Biomedicine	3.632	7.5	Elsevier	https://www.journals.elsevier.com/computer-methods-and-programs-in-biomedicine
Machine Learning	2.672	5.0	Springer	https://www.springer.com/journal/10994
Machine Vision and Applications	1.605	4.2	Springer	https://www.springer.com/journal/138
Medical Image Analysis	11.148	17.2	Elsevier	https://www.sciencedirect.com/journal/medical-image-analysis
IEEE Access	3.745	3.9	IEEE	https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6287639
IEEE Transactions on Knowledge and Data Engineering	4.935	12.0	IEEE	https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=69
Nature Communications	12.121	18.1	Nature	https://www.nature.com/ncomms/
IEEE Transactions on Intelligent Transportation Systems	6.319	12.7	IEEE	https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6979
Methods	3.812	8.0	Elsevier	https://www.journals.elsevier.com/methods
ACM Journal on Emerging Technologies in Computing Systems	1.652	4.3	ACM	https://dl.acm.org/journal/jetc
ACM Computing Surveys	6.319	12.7	ACM	https://dl.acm.org/journal/csur
Applied Soft Computing	5.472	10.2	Elsevier	https://www.journals.elsevier.com/applied-soft-computing
Electronics	2.412	1.9	MDPI	https://www.mdpi.com/journal/electronics
Applied Sciences	2.474	2.4	MDPI	https://www.mdpi.com/journal/applsci
IEEE Transactions on Industrial Informatics	9.112	13.9	IEEE	https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9424

cancer, US board-certified general pathologists achieved an average accuracy of 61%, while the Google AI [44] outperformed these specialists by achieving an average accuracy of 70%. In 2020, DL is playing an increasingly vital role in early diagnosis of the novel coronavirus (COVID-19) [29, 46–48]. DL has become the main tool in many hospitals around the world for automatic COVID-19 classification and detection using chest

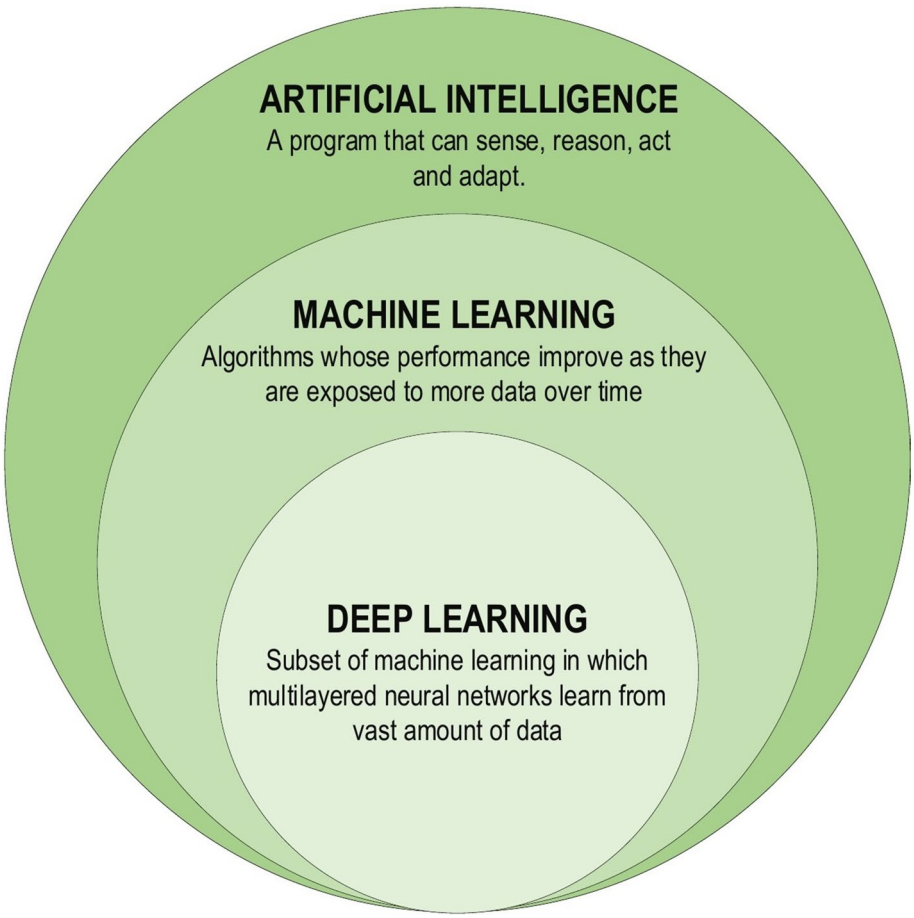

Fig. 2 Deep learning family

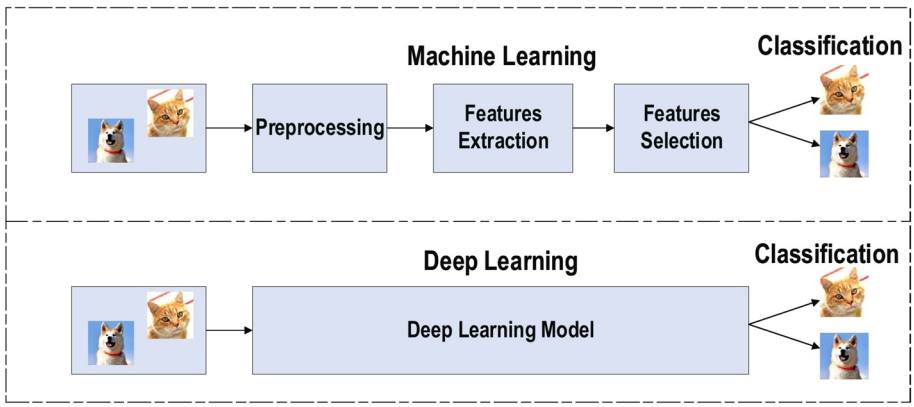

Fig. 3 The difference between deep learning and traditional machine learning

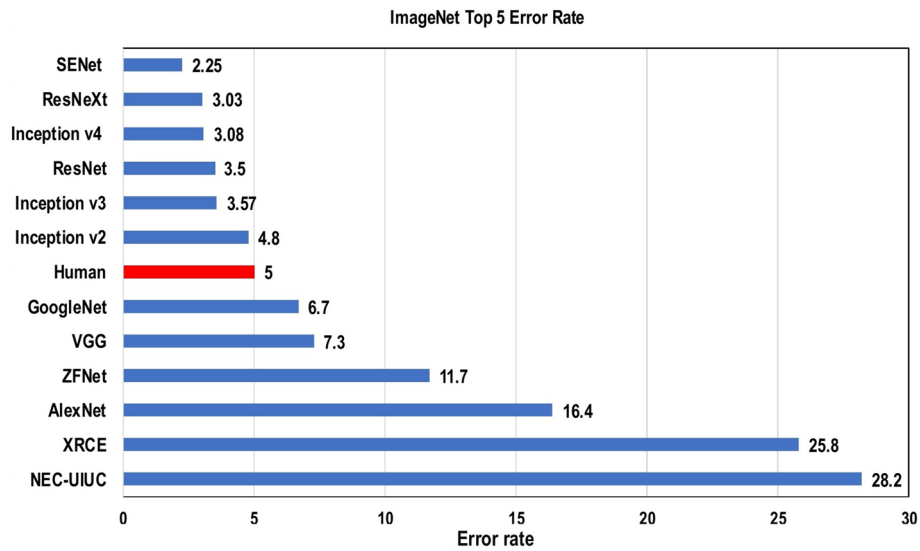

Fig. 4 Deep learning performance compared to human

X-ray images or other types of images. We end this section by the saying of AI pioneer Geoffrey Hinton "Deep learning is going to be able to do everything".

When to apply deep learning

Machine intelligence is useful in many situations which is equal or better than human experts in some cases [49–52], meaning that DL can be a solution to the following problems:

- Cases where human experts are not available.
- Cases where humans are unable to explain decisions made using their expertise (language understanding, medical decisions, and speech recognition).
- Cases where the problem solution updates over time (price prediction, stock preference, weather prediction, and tracking).
- Cases where solutions require adaptation based on specific cases (personalization, biometrics).
- Cases where size of the problem is extremely large and exceeds our inadequate reasoning abilities (sentiment analysis, matching ads to Facebook, calculation webpage ranks).

Why deep learning?

Several performance features may answer this question, e.g

1. Universal Learning Approach: Because DL has the ability to perform in approximately all application domains, it is sometimes referred to as universal learning.
2. Robustness: In general, precisely designed features are not required in DL techniques. Instead, the optimized features are learned in an automated fashion related

to the task under consideration. Thus, robustness to the usual changes of the input data is attained.

3. Generalization: Different data types or different applications can use the same DL technique, an approach frequently referred to as transfer learning (TL) which explained in the latter section. Furthermore, it is a useful approach in problems where data is insufficient.
4. Scalability: DL is highly scalable. ResNet [37], which was invented by Microsoft, comprises 1202 layers and is frequently applied at a supercomputing scale. Lawrence Livermore National Laboratory (LLNL), a large enterprise working on evolving frameworks for networks, adopted a similar approach, where thousands of nodes can be implemented [53].

Classification of DL approaches

DL techniques are classified into three major categories: unsupervised, partially supervised (semi-supervised) and supervised. Furthermore, deep reinforcement learning (DRL), also known as RL, is another type of learning technique, which is mostly considered to fall into the category of partially supervised (and occasionally unsupervised) learning techniques.

Deep supervised learning

This technique deals with labeled data. When considering such a technique, the environs have a collection of inputs and resultant outputs $(x_t, y_t) \sim \rho$. For instance, the smart agent guesses $y\hat{}_t = f(x_t)$ if the input is xt and will obtain $\imath(y\hat{}_t, y_t)$ as a loss value. Next, the network parameters are repeatedly updated by the agent to obtain an improved estimate for the preferred outputs. Following a positive training outcome, the agent acquires the ability to obtain the right solutions to the queries from the environs. For DL, there are several supervised learning techniques, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and deep neural networks (DNNs). In addition, the RNN category includes gated recurrent units (GRUs) and long short-term memory (LSTM) approaches. The main advantage of this technique is the ability to collect data or generate a data output from the prior knowledge. However, the disadvantage of this technique is that decision boundary might be overstrained when training set doesn't own samples that should be in a class. Overall, this technique is simpler than other techniques in the way of learning with high performance.

Deep semi-supervised learning

In this technique, the learning process is based on semi-labeled datasets. Occasionally, generative adversarial networks (GANs) and DRL are employed in the same way as this technique. In addition, RNNs, which include GRUs and LSTMs, are also employed for partially supervised learning. One of the advantages of this technique is to minimize the amount of labeled data needed. On other the hand, One of the disadvantages of this technique is irrelevant input feature present training data could furnish incorrect decisions. Text document classifier is one of the most popular example of an application of

semi-supervised learning. Due to difficulty of obtaining a large amount of labeled text documents, semi-supervised learning is ideal for text document classification task.

Deep unsupervised learning

This technique makes it possible to implement the learning process in the absence of available labeled data (i.e. no labels are required). Here, the agent learns the significant features or interior representation required to discover the unidentified structure or relationships in the input data. Techniques of generative networks, dimensionality reduction and clustering are frequently counted within the category of unsupervised learning. Several members of the DL family have performed well on non-linear dimensionality reduction and clustering tasks; these include restricted Boltzmann machines, auto-encoders and GANs as the most recently developed techniques. Moreover, RNNs, which include GRUs and LSTM approaches, have also been employed for unsupervised learning in a wide range of applications. The main disadvantages of unsupervised learning are unable to provide accurate information concerning data sorting and computationally complex. One of the most popular unsupervised learning approaches is clustering [54].

Deep reinforcement learning

Reinforcement Learning operates on interacting with the environment, while supervised learning operates on provided sample data. This technique was developed in 2013 with Google Deep Mind [55]. Subsequently, many enhanced techniques dependent on reinforcement learning were constructed. For example, if the input environment samples: $x_t \sim \rho$, agent predict: $y\hat{}_t = x(f_t)$ and the received cost of the agent is $c_t \sim P(c_t \mid x_t, y\hat{}_t)$, P here is the unknown probability distribution, then the environment asks a question to the agent. The answer it gives is a noisy score. This method is sometimes referred to as semi-supervised learning. Based on this concept, several supervised and unsupervised techniques were developed. In comparison with traditional supervised techniques, performing this learning is much more difficult, as no straightforward loss function is available in the reinforcement learning technique. In addition, there are two essential differences between supervised learning and reinforcement learning: first, there is no complete access to the function, which requires optimization, meaning that it should be queried via interaction; second, the state being interacted with is founded on an environment, where the input x_t is based on the preceding actions [9, 56].

For solving a task, the selection of the type of reinforcement learning that needs to be performed is based on the space or the scope of the problem. For example, DRL is the best way for problems involving many parameters to be optimized. By contrast, derivative-free reinforcement learning is a technique that performs well for problems with limited parameters. Some of the applications of reinforcement learning are business strategy planning and robotics for industrial automation. The main drawback of Reinforcement Learning is that parameters may influence the speed of learning. Here are the main motivations for utilizing Reinforcement Learning:

- It assists you to identify which action produces the highest reward over a longer period.
- It assists you to discover which situation requires action.
- It also enables it to figure out the best approach for reaching large rewards.
- Reinforcement Learning also gives the learning agent a reward function.

Reinforcement Learning can't utilize in all the situation such as:

- In case there is sufficient data to resolve the issue with supervised learning techniques.
- Reinforcement Learning is computing-heavy and time-consuming. Specially when the workspace is large.

Types of DL networks

The most famous types of deep learning networks are discussed in this section: these include recursive neural networks (RvNNs), RNNs, and CNNs. RvNNs and RNNs were briefly explained in this section while CNNs were explained in deep due to the importance of this type. Furthermore, it is the most used in several applications among other networks.

Recursive neural networks

RvNN can achieve predictions in a hierarchical structure also classify the outputs utilizing compositional vectors [57]. Recursive auto-associative memory (RAAM) [58] is the primary inspiration for the RvNN development. The RvNN architecture is generated for processing objects, which have randomly shaped structures like graphs or trees. This approach generates a fixed-width distributed representation from a variable-size recursive-data structure. The network is trained using an introduced back-propagation through structure (BTS) learning system [58]. The BTS system tracks the same technique as the general-back propagation algorithm and has the ability to support a treelike structure. Auto-association trains the network to regenerate the input-layer pattern at the output layer. RvNN is highly effective in the NLP context. Socher et al. [59] introduced RvNN architecture designed to process inputs from a variety of modalities. These authors demonstrate two applications for classifying natural language sentences: cases where each sentence is split into words and nature images, and cases where each image is separated into various segments of interest. RvNN computes a likely pair of scores for merging and constructs a syntactic tree. Furthermore, RvNN calculates a score related to the merge plausibility for every pair of units. Next, the pair with the largest score is merged within a composition vector. Following every merge, RvNN generates (a) a larger area of numerous units, (b) a compositional vector of the area, and (c) a label for the class (for instance, a noun phrase will become the class label for the new area if two units are noun words). The compositional vector for the entire area is the root of the RvNN tree structure. An example RvNN tree is shown in Fig. 5. RvNN has been employed in several applications [60–62].

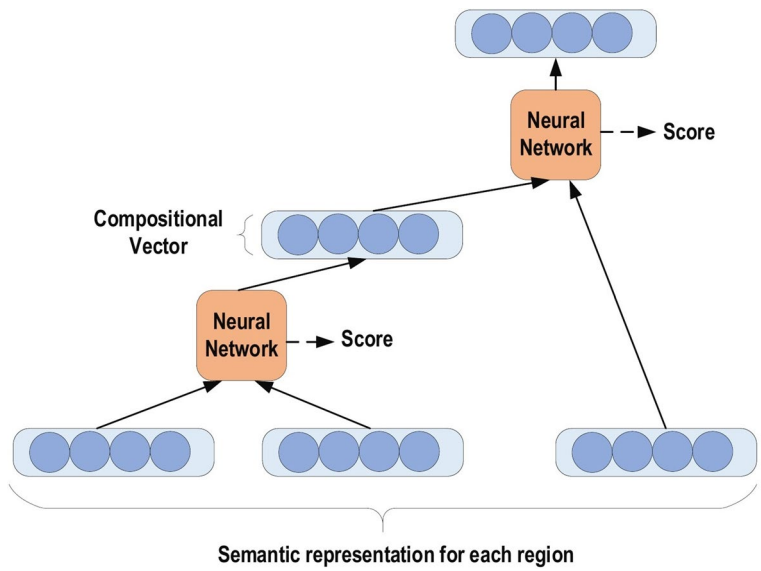

Fig. 5 An example of RvNN tree

Recurrent neural networks

RNNs are a commonly employed and familiar algorithm in the discipline of DL [63–65]. RNN is mainly applied in the area of speech processing and NLP contexts [66, 67]. Unlike conventional networks, RNN uses sequential data in the network. Since the embedded structure in the sequence of the data delivers valuable information, this feature is fundamental to a range of different applications. For instance, it is important to understand the context of the sentence in order to determine the meaning of a specific word in it. Thus, it is possible to consider the RNN as a unit of short-term memory, where x represents the input layer, y is the output layer, and s represents the state (hidden) layer. For a given input sequence, a typical unfolded RNN diagram is illustrated in Fig. 6. Pascanu et al. [68] introduced three different types of deep RNN techniques, namely "Hidden-to-Hidden", "Hidden-to-Output", and "Input-to-Hidden". A deep RNN is introduced that lessens the learning difficulty in the deep network and brings the benefits of a deeper RNN based on these three techniques.

However, RNN's sensitivity to the exploding gradient and vanishing problems represent one of the main issues with this approach [69]. More specifically, during the training process, the reduplications of several large or small derivatives may cause the gradients to exponentially explode or decay. With the entrance of new inputs, the network stops thinking about the initial ones; therefore, this sensitivity decays over time. Furthermore, this issue can be handled using LSTM [70]. This approach offers recurrent connections to memory blocks in the network. Every memory block contains a number of memory cells, which have the ability to store the temporal states of the network. In addition, it contains gated units for controlling the flow of information. In very deep networks [37], residual connections also have the ability to considerably reduce the impact of the vanishing gradient issue which explained in later sections.

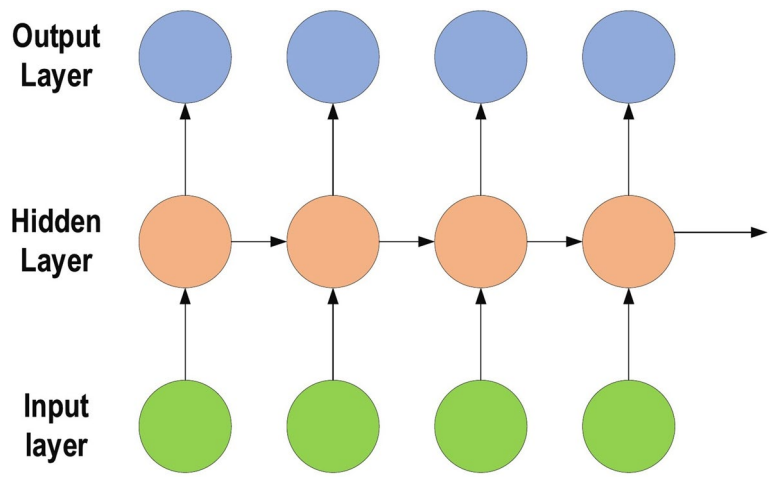

Fig. 6 Typical unfolded RNN diagram

CNN is considered to be more powerful than RNN. RNN includes less feature compatibility when compared to CNN.

Convolutional neural networks

In the field of DL, the CNN is the most famous and commonly employed algorithm [30, 71–75]. The main benefit of CNN compared to its predecessors is that it automatically identifies the relevant features without any human supervision [76]. CNNs have been extensively applied in a range of different fields, including computer vision [77], speech processing [78], Face Recognition [79], etc. The structure of CNNs was inspired by neurons in human and animal brains, similar to a conventional neural network. More specifically, in a cat's brain, a complex sequence of cells forms the visual cortex; this sequence is simulated by the CNN [80]. Goodfellow et al. [28] identified three key benefits of the CNN: equivalent representations, sparse interactions, and parameter sharing. Unlike conventional fully connected (FC) networks, shared weights and local connections in the CNN are employed to make full use of 2D input-data structures like image signals. This operation utilizes an extremely small number of parameters, which both simplifies the training process and speeds up the network. This is the same as in the visual cortex cells. Notably, only small regions of a scene are sensed by these cells rather than the whole scene (i.e., these cells spatially extract the local correlation available in the input, like local filters over the input).

A commonly used type of CNN, which is similar to the multi-layer perceptron (MLP), consists of numerous convolution layers preceding sub-sampling (pooling) layers, while the ending layers are FC layers. An example of CNN architecture for image classification is illustrated in Fig. 7.

The input x of each layer in a CNN model is organized in three dimensions: height, width, and depth, or $m \times m \times r$, where the height (m) is equal to the width. The depth is also referred to as the channel number. For example, in an RGB image, the depth (r) is equal to three. Several kernels (filters) available in each convolutional layer are denoted by k and also have three dimensions ($n \times n \times q$), similar to the input image; here,

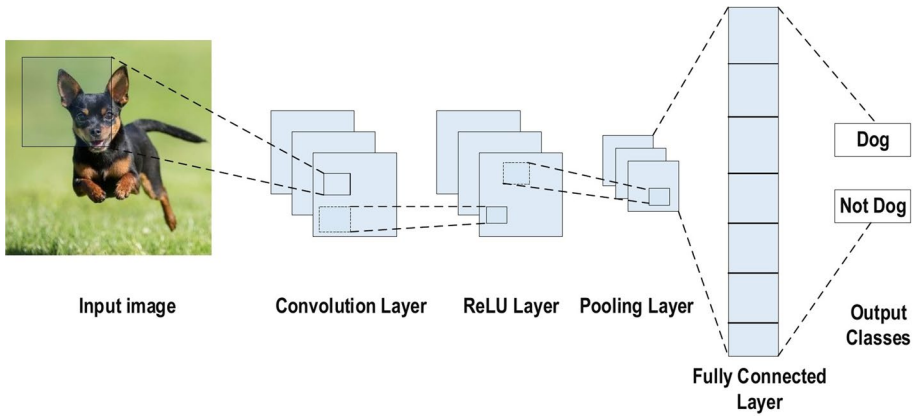

Fig. 7 An example of CNN architecture for image classification

however, n must be smaller than m, while q is either equal to or smaller than r. In addition, the kernels are the basis of the local connections, which share similar parameters (bias b^k and weight W^k) for generating k feature maps h^k with a size of $(m - n - 1)$ each and are convolved with input, as mentioned above. The convolution layer calculates a dot product between its input and the weights as in Eq. 1, similar to NLP, but the inputs are undersized areas of the initial image size. Next, by applying the nonlinearity or an activation function to the convolution-layer output, we obtain the following:

$$h^k = f(W^k * x + b^k) \tag{1}$$

The next step is down-sampling every feature map in the sub-sampling layers. This leads to a reduction in the network parameters, which accelerates the training process and in turn enables handling of the overfitting issue. For all feature maps, the pooling function (e.g. max or average) is applied to an adjacent area of size $p \times p$, where p is the kernel size. Finally, the FC layers receive the mid- and low-level features and create the high-level abstraction, which represents the last-stage layers as in a typical neural network. The classification scores are generated using the ending layer [e.g. support vector machines (SVMs) or softmax]. For a given instance, every score represents the probability of a specific class.

Benefits of employing CNNs

The benefits of using CNNs over other traditional neural networks in the computer vision environment are listed as follows:

1. The main reason to consider CNN is the weight sharing feature, which reduces the number of trainable network parameters and in turn helps the network to enhance generalization and to avoid overfitting.
2. Concurrently learning the feature extraction layers and the classification layer causes the model output to be both highly organized and highly reliant on the extracted features.

3. Large-scale network implementation is much easier with CNN than with other neural networks.

CNN layers

The CNN architecture consists of a number of layers (or so-called multi-building blocks). Each layer in the CNN architecture, including its function, is described in detail below.

1. Convolutional Layer: In CNN architecture, the most significant component is the convolutional layer. It consists of a collection of convolutional filters (so-called kernels). The input image, expressed as N-dimensional metrics, is convolved with these filters to generate the output feature map.

   - Kernel definition: A grid of discrete numbers or values describes the kernel. Each value is called the kernel weight. Random numbers are assigned to act as the weights of the kernel at the beginning of the CNN training process. In addition, there are several different methods used to initialize the weights. Next, these weights are adjusted at each training era; thus, the kernel learns to extract significant features.
   - Convolutional Operation: Initially, the CNN input format is described. The vector format is the input of the traditional neural network, while the multi-channeled image is the input of the CNN. For instance, single-channel is the format of the gray-scale image, while the RGB image format is three-channeled. To understand the convolutional operation, let us take an example of a 4×4 gray-scale image with a 2×2 random weight-initialized kernel. First, the kernel slides over the whole image horizontally and vertically. In addition, the dot product between the input image and the kernel is determined, where their corresponding values are multiplied and then summed up to create a single scalar value, calculated concurrently. The whole process is then repeated until no further sliding is possible. Note that the calculated dot product values represent the feature map of the output. Figure 8 graphically illustrates the primary calculations executed at each step. In this figure, the light green color represents the 2×2 kernel, while the light blue color represents the similar size area of the input image. Both are multiplied; the end result after summing up the resulting product values (marked in a light orange color) represents an entry value to the output feature map.
     However, padding to the input image is not applied in the previous example, while a stride of one (denoted for the selected step-size over all vertical or horizontal locations) is applied to the kernel. Note that it is also possible to use another stride value. In addition, a feature map of lower dimensions is obtained as a result of increasing the stride value.
     On the other hand, padding is highly significant to determining border size information related to the input image. By contrast, the border side-features moves carried away very fast. By applying padding, the size of the input image

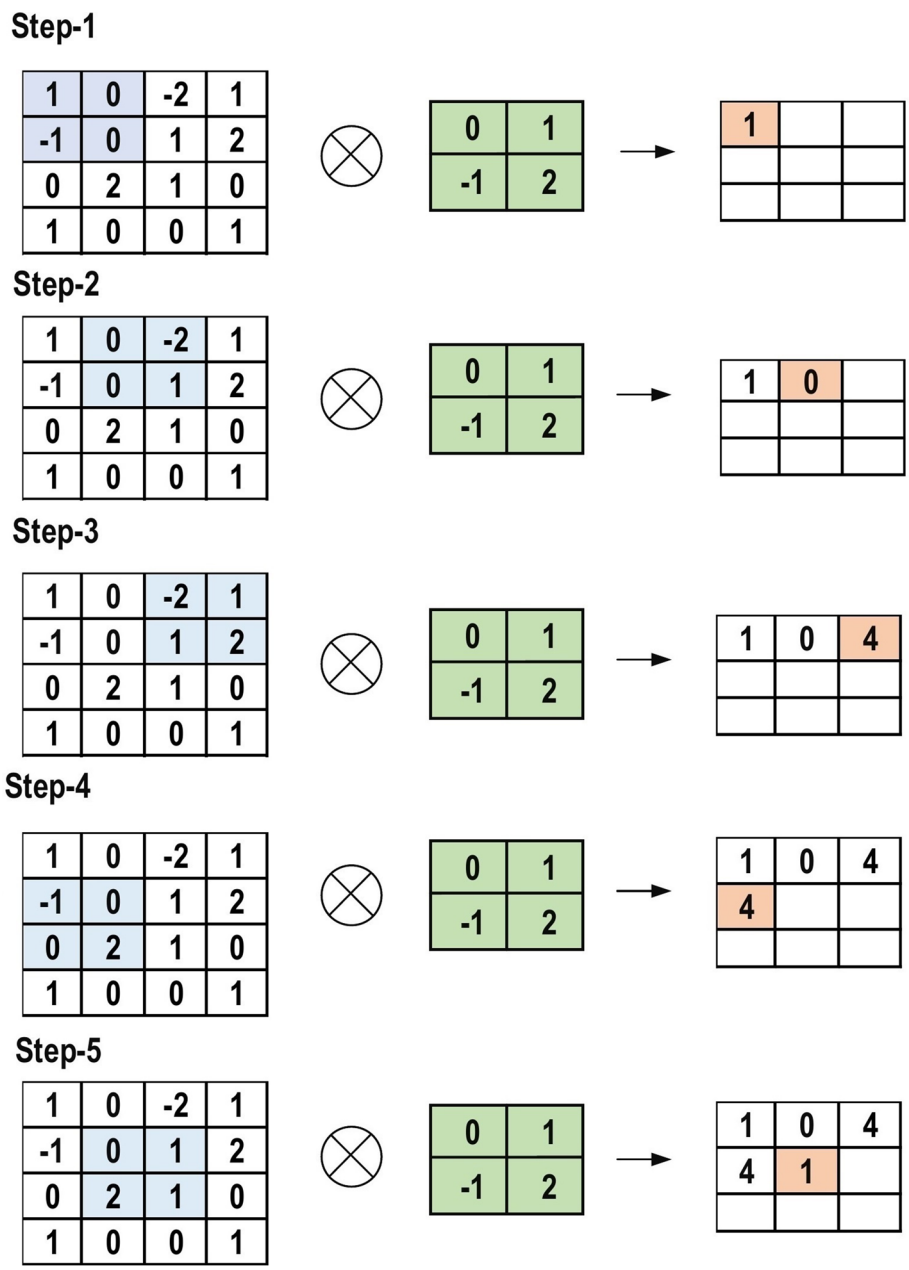

Fig. 8 The primary calculations executed at each step of convolutional layer

will increase, and in turn, the size of the output feature map will also increase. Core Benefits of Convolutional Layers.

- Sparse Connectivity: Each neuron of a layer in FC neural networks links with all neurons in the following layer. By contrast, in CNNs, only a few weights are available between two adjacent layers. Thus, the number of required weights or connections is small, while the memory required to store these weights is also

small; hence, this approach is memory-effective. In addition, matrix operation is computationally much more costly than the dot (.) operation in CNN.

- Weight Sharing: There are no allocated weights between any two neurons of neighboring layers in CNN, as the whole weights operate with one and all pixels of the input matrix. Learning a single group of weights for the whole input will significantly decrease the required training time and various costs, as it is not necessary to learn additional weights for each neuron.

2. Pooling Layer: The main task of the pooling layer is the sub-sampling of the feature maps. These maps are generated by following the convolutional operations. In other words, this approach shrinks large-size feature maps to create smaller feature maps. Concurrently, it maintains the majority of the dominant information (or features) in every step of the pooling stage. In a similar manner to the convolutional operation, both the stride and the kernel are initially size-assigned before the pooling operation is executed. Several types of pooling methods are available for utilization in various pooling layers. These methods include tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling (GAP), and global max pooling. The most familiar and frequently utilized pooling methods are the max, min, and GAP pooling. Figure 9 illustrates these three pooling operations.
   Sometimes, the overall CNN performance is decreased as a result; this represents the main shortfall of the pooling layer, as this layer helps the CNN to determine whether or not a certain feature is available in the particular input image, but focuses exclusively on ascertaining the correct location of that feature. Thus, the CNN model misses the relevant information.
3. Activation Function (non-linearity) Mapping the input to the output is the core function of all types of activation function in all types of neural network. The input value is determined by computing the weighted summation of the neuron input along with its bias (if present). This means that the activation function makes the decision as to whether or not to fire a neuron with reference to a particular input by creating the corresponding output.

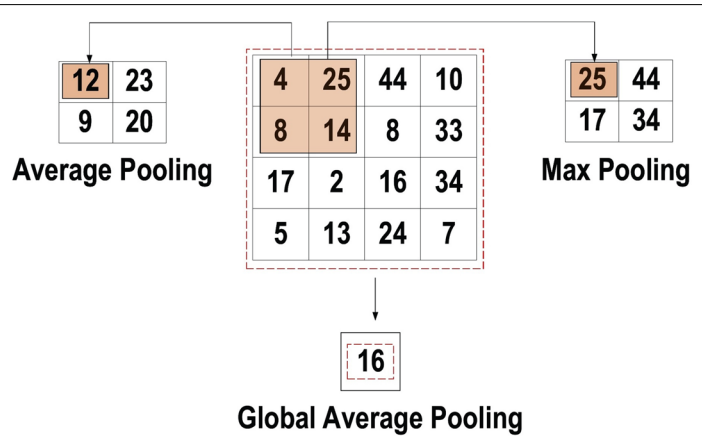

Fig. 9 Three types of pooling operations

Non-linear activation layers are employed after all layers with weights (so-called learnable layers, such as FC layers and convolutional layers) in CNN architecture. This non-linear performance of the activation layers means that the mapping of input to output will be non-linear; moreover, these layers give the CNN the ability to learn extra-complicated things. The activation function must also have the ability to differentiate, which is an extremely significant feature, as it allows error back-propagation to be used to train the network. The following types of activation functions are most commonly used in CNN and other deep neural networks.

- Sigmoid: The input of this activation function is real numbers, while the output is restricted to between zero and one. The sigmoid function curve is S-shaped and can be represented mathematically by Eq. 2.

$$f(x)_{sigm} = \frac{1}{1 + e^{-x}} \tag{2}$$

- Tanh: It is similar to the sigmoid function, as its input is real numbers, but the output is restricted to between -1 and 1. Its mathematical representation is in Eq. 3.

$$f(x)_{tanh} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3}$$

- ReLU: The mostly commonly used function in the CNN context. It converts the whole values of the input to positive numbers. Lower computational load is the main benefit of ReLU over the others. Its mathematical representation is in Eq. 4.

$$f(x)_{ReLU} = max(0, x) \tag{4}$$

  Occasionally, a few significant issues may occur during the use of ReLU. For instance, consider an error back-propagation algorithm with a larger gradient flowing through it. Passing this gradient within the ReLU function will update the weights in a way that makes the neuron certainly not activated once more. This issue is referred to as "Dying ReLU". Some ReLU alternatives exist to solve such issues. The following discusses some of them.
- Leaky ReLU: Instead of ReLU down-scaling the negative inputs, this activation function ensures these inputs are never ignored. It is employed to solve the Dying ReLU problem. Leaky ReLU can be represented mathematically as in Eq. 5.

$$f(x)_{LeakyReLU} = \left\{ \begin{array}{l} x, \quad if \quad x > 0 \\ mx, \ x \leq 0 \end{array} \right\} \tag{5}$$

  Note that the leak factor is denoted by m. It is commonly set to a very small value, such as 0.001.
- Noisy ReLU: This function employs a Gaussian distribution to make ReLU noisy. It can be represented mathematically as in Eq. 6.

$$f(x)_{NoisyReLU} = max(x + Y), with\ Y \sim N(0, \sigma(x)) \tag{6}$$

- Parametric Linear Units: This is mostly the same as Leaky ReLU. The main difference is that the leak factor in this function is updated through the model training process. The parametric linear unit can be represented mathematically as in Eq. 7.

$$f(x)_{ParametricLinear} = \begin{Bmatrix} x, & if\ x > 0 \\ ax, & x \leq 0 \end{Bmatrix} \tag{7}$$

Note that the learnable weight is denoted as a.

4. Fully Connected Layer: Commonly, this layer is located at the end of each CNN architecture. Inside this layer, each neuron is connected to all neurons of the previous layer, the so-called Fully Connected (FC) approach. It is utilized as the CNN classifier. It follows the basic method of the conventional multiple-layer perceptron neural network, as it is a type of feed-forward ANN. The input of the FC layer comes from the last pooling or convolutional layer. This input is in the form of a vector, which is created from the feature maps after flattening. The output of the FC layer represents the final CNN output, as illustrated in Fig. 10.
5. Loss Functions: The previous section has presented various layer-types of CNN architecture. In addition, the final classification is achieved from the output layer, which represents the last layer of the CNN architecture. Some loss functions are utilized in the output layer to calculate the predicted error created across the training samples in the CNN model. This error reveals the difference between the actual output and the predicted one. Next, it will be optimized through the CNN learning process.

   However, two parameters are used by the loss function to calculate the error. The CNN estimated output (referred to as the prediction) is the first parameter. The actual output (referred to as the label) is the second parameter. Several types of loss function are employed in various problem types. The following concisely explains some of the loss function types.

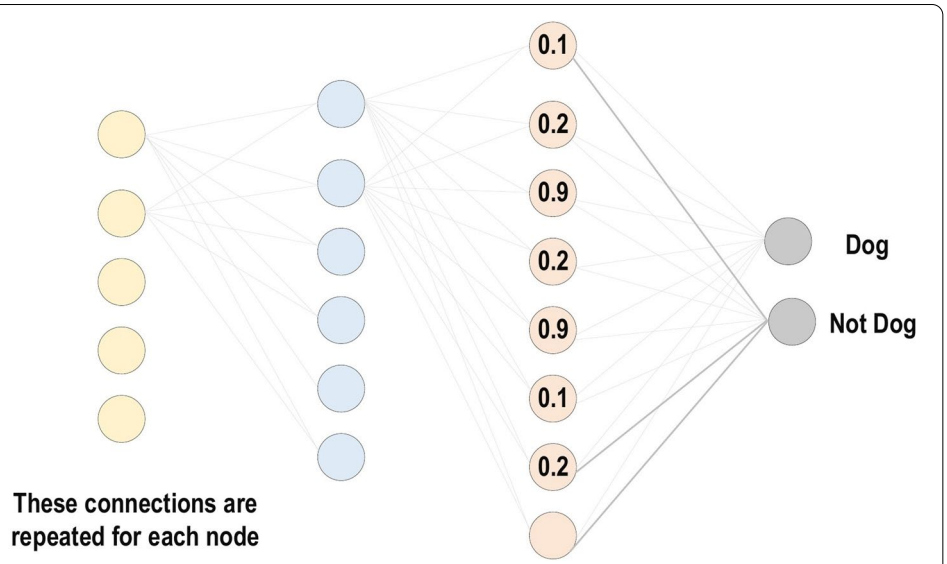

Fig. 10 Fully connected layer

(a) Cross-Entropy or Softmax Loss Function: This function is commonly employed for measuring the CNN model performance. It is also referred to as the log loss function. Its output is the probability $p \in \{0, 1\}$. In addition, it is usually employed as a substitution of the square error loss function in multi-class classification problems. In the output layer, it employs the softmax activations to generate the output within a probability distribution. The mathematical representation of the output class probability is Eq. 8.

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \tag{8}$$

Here, e^{a_i} represents the non-normalized output from the preceding layer, while N represents the number of neurons in the output layer. Finally, the mathematical representation of cross-entropy loss function is Eq. 9.

$$H(p, y) = -\sum_i y_i \log(p_i) \quad where \quad i \in [1, N] \tag{9}$$

(b) Euclidean Loss Function: This function is widely used in regression problems. In addition, it is also the so-called mean square error. The mathematical expression of the estimated Euclidean loss is Eq. 10.

$$H(p, y) = \frac{1}{2N} \sum_{i=1}^{N} (p_i - y_i)^2 \tag{10}$$

(c) Hinge Loss Function: This function is commonly employed in problems related to binary classification. This problem relates to maximum-margin-based classification; this is mostly important for SVMs, which use the hinge loss function, wherein the optimizer attempts to maximize the margin around dual objective classes. Its mathematical formula is Eq. 11.

$$H(p, y) = \sum_{i=1}^{N} max(0, m - (2y_i - 1)p_i) \tag{11}$$

The margin m is commonly set to 1. Moreover, the predicted output is denoted as p_i, while the desired output is denoted as y_i.

Regularization to CNN

For CNN models, over-fitting represents the central issue associated with obtaining well-behaved generalization. The model is entitled over-fitted in cases where the model executes especially well on training data and does not succeed on test data (unseen data) which is more explained in the latter section. An under-fitted model is the opposite; this case occurs when the model does not learn a sufficient amount from the training data. The model is referred to as "just-fitted" if it executes well on both training and testing data. These three types are illustrated in Fig. 11. Various intuitive concepts are used to help the regularization to avoid over-fitting; more details about over-fitting and under-fitting are discussed in latter sections.

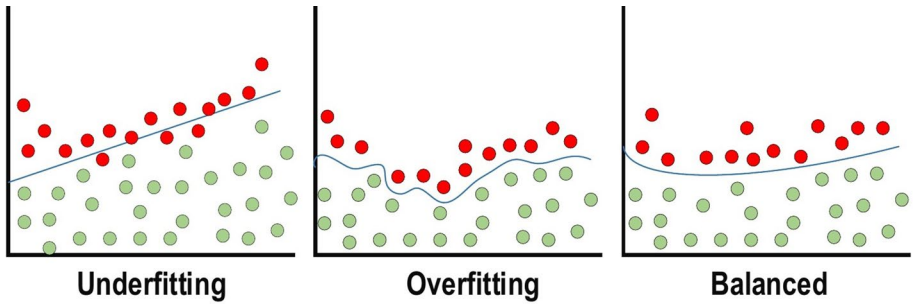

Fig. 11 Over-fitting and under-fitting issues

1. Dropout: This is a widely utilized technique for generalization. During each training epoch, neurons are randomly dropped. In doing this, the feature selection power is distributed equally across the whole group of neurons, as well as forcing the model to learn different independent features. During the training process, the dropped neuron will not be a part of back-propagation or forward-propagation. By contrast, the full-scale network is utilized to perform prediction during the testing process.
2. Drop-Weights: This method is highly similar to dropout. In each training epoch, the connections between neurons (weights) are dropped rather than dropping the neurons; this represents the only difference between drop-weights and dropout.
3. Data Augmentation: Training the model on a sizeable amount of data is the easiest way to avoid over-fitting. To achieve this, data augmentation is used. Several techniques are utilized to artificially expand the size of the training dataset. More details can be found in the latter section, which describes the data augmentation techniques.
4. Batch Normalization: This method ensures the performance of the output activations [81]. This performance follows a unit Gaussian distribution. Subtracting the mean and dividing by the standard deviation will normalize the output at each layer. While it is possible to consider this as a pre-processing task at each layer in the network, it is also possible to differentiate and to integrate it with other networks. In addition, it is employed to reduce the "internal covariance shift" of the activation layers. In each layer, the variation in the activation distribution defines the internal covariance shift. This shift becomes very high due to the continuous weight updating through training, which may occur if the samples of the training data are gathered from numerous dissimilar sources (for example, day and night images). Thus, the model will consume extra time for convergence, and in turn, the time required for training will also increase. To resolve this issue, a layer representing the operation of batch normalization is applied in the CNN architecture.

   The advantages of utilizing batch normalization are as follows:

   - It prevents the problem of vanishing gradient from arising.
   - It can effectively control the poor weight initialization.
   - It significantly reduces the time required for network convergence (for large-scale datasets, this will be extremely useful).
   - It struggles to decrease training dependency across hyper-parameters.

- Chances of over-fitting are reduced, since it has a minor influence on regularization.

Optimizer selection

This section discusses the CNN learning process. Two major issues are included in the learning process: the first issue is the learning algorithm selection (optimizer), while the second issue is the use of many enhancements (such as AdaDelta, Adagrad, and momentum) along with the learning algorithm to enhance the output.

Loss functions, which are founded on numerous learnable parameters (e.g. biases, weights, etc.) or minimizing the error (variation between actual and predicted output), are the core purpose of all supervised learning algorithms. The techniques of gradient-based learning for a CNN network appear as the usual selection. The network parameters should always update though all training epochs, while the network should also look for the locally optimized answer in all training epochs in order to minimize the error.

The learning rate is defined as the step size of the parameter updating. The training epoch represents a complete repetition of the parameter update that involves the complete training dataset at one time. Note that it needs to select the learning rate wisely so that it does not influence the learning process imperfectly, although it is a hyper-parameter.

Gradient Descent or Gradient-based learning algorithm: To minimize the training error, this algorithm repetitively updates the network parameters through every training epoch. More specifically, to update the parameters correctly, it needs to compute the objective function gradient (slope) by applying a first-order derivative with respect to the network parameters. Next, the parameter is updated in the reverse direction of the gradient to reduce the error. The parameter updating process is performed though network back-propagation, in which the gradient at every neuron is back-propagated to all neurons in the preceding layer. The mathematical representation of this operation is as Eq. 12.

$$w_{ij^t} = w_{ij^{t-1}} - \Delta w_{ij^t}, \quad \Delta w_{ij^t} = \eta * \frac{\partial E}{\partial w_{ij}} \tag{12}$$

The final weight in the current training epoch is denoted by w_{ij^t}, while the weight in the preceding $(t-1)$ training epoch is denoted $w_{ij^{t-1}}$. The learning rate is η and the prediction error is E. Different alternatives of the gradient-based learning algorithm are available and commonly employed; these include the following:

1. Batch Gradient Descent: During the execution of this technique [82], the network parameters are updated merely one time behind considering all training datasets via the network. In more depth, it calculates the gradient of the whole training set and subsequently uses this gradient to update the parameters. For a small-sized dataset, the CNN model converges faster and creates an extra-stable gradient using BGD. Since the parameters are changed only once for every training epoch, it requires a substantial amount of resources. By contrast, for a large training dataset, additional

time is required for converging, and it could converge to a local optimum (for non-convex instances).

2. Stochastic Gradient Descent: The parameters are updated at each training sample in this technique [83]. It is preferred to arbitrarily sample the training samples in every epoch in advance of training. For a large-sized training dataset, this technique is both more memory-effective and much faster than BGD. However, because it is frequently updated, it takes extremely noisy steps in the direction of the answer, which in turn causes the convergence behavior to become highly unstable.
3. Mini-batch Gradient Descent: In this approach, the training samples are partitioned into several mini-batches, in which every mini-batch can be considered an under-sized collection of samples with no overlap between them [84]. Next, parameter updating is performed following gradient computation on every mini-batch. The advantage of this method comes from combining the advantages of both BGD and SGD techniques. Thus, it has a steady convergence, more computational efficiency and extra memory effectiveness. The following describes several enhancement techniques in gradient-based learning algorithms (usually in SGD), which further powerfully enhance the CNN training process.
4. Momentum: For neural networks, this technique is employed in the objective function. It enhances both the accuracy and the training speed by summing the computed gradient at the preceding training step, which is weighted via a factor λ (known as the momentum factor). However, it therefore simply becomes stuck in a local minimum rather than a global minimum. This represents the main disadvantage of gradient-based learning algorithms. Issues of this kind frequently occur if the issue has no convex surface (or solution space).

   Together with the learning algorithm, momentum is used to solve this issue, which can be expressed mathematically as in Eq. 13.

$$\Delta w_{ij^t} = \left(\eta * \frac{\partial E}{\partial w_{ij}}\right) + (\lambda * \Delta w_{ij^{t-1}}) \tag{13}$$

   The weight increment in the current t'th training epoch is denoted as Δw_{ij^t}, while η is the learning rate, and the weight increment in the preceding $(t-1)'$th training epoch. The momentum factor value is maintained within the range 0 to 1; in turn, the step size of the weight updating increases in the direction of the bare minimum to minimize the error. As the value of the momentum factor becomes very low, the model loses its ability to avoid the local bare minimum. By contrast, as the momentum factor value becomes high, the model develops the ability to converge much more rapidly. If a high value of momentum factor is used together with LR, then the model could miss the global bare minimum by crossing over it.

   However, when the gradient varies its direction continually throughout the training process, then the suitable value of the momentum factor (which is a hyper-parameter) causes a smoothening of the weight updating variations.
5. Adaptive Moment Estimation (Adam): It is another optimization technique or learning algorithm that is widely used. Adam [85] represents the latest trends in deep

learning optimization. This is represented by the Hessian matrix, which employs a second-order derivative. Adam is a learning strategy that has been designed specifically for training deep neural networks. More memory efficient and less computational power are two advantages of Adam. The mechanism of Adam is to calculate adaptive LR for each parameter in the model. It integrates the pros of both Momentum and RMSprop. It utilizes the squared gradients to scale the learning rate as RMSprop and it is similar to the momentum by using the moving average of the gradient. The equation of Adam is represented in Eq. 14.

$$w_{ij^t} = w_{ij^{t-1}} - \frac{\eta}{\sqrt{\widehat{E[\delta^2]^t}} + \in} * \widehat{E[\delta^2]^t} \tag{14}$$

Design of algorithms (backpropagation)

Let's start with a notation that refers to weights in the network unambiguously. We denote $\boldsymbol{w}_{ij}^{h}$ to be the weight for the connection from ith input or (neuron at (h − 1)th) to the jt neuron in the hth layer. So, Fig. 12 shows the weight on a connection from the neuron in the first layer to another neuron in the next layer in the network.

Where w_{11}^{2} has represented the weight from the first neuron in the first layer to the first neuron in the second layer, based on that the second weight for the same neuron will be w_{21}^{2} which means is the weight comes from the second neuron in the previous layer to the first layer in the next layer which is the second in this net. Regarding the bias, since the bias is not the connection between the neurons for the layers, so it is easily handled each neuron must have its own bias, some network each layer has a certain bias. It can be seen from the above net that each layer has its own bias. Each network has the parameters such as the no of the layer in the net, the number of the neurons in each layer, no of the weight (connection) between the layers, the no of connection can be easily determined based on the no of neurons in each layer, for example, if there are ten input fully connect with two neurons in the next layer then

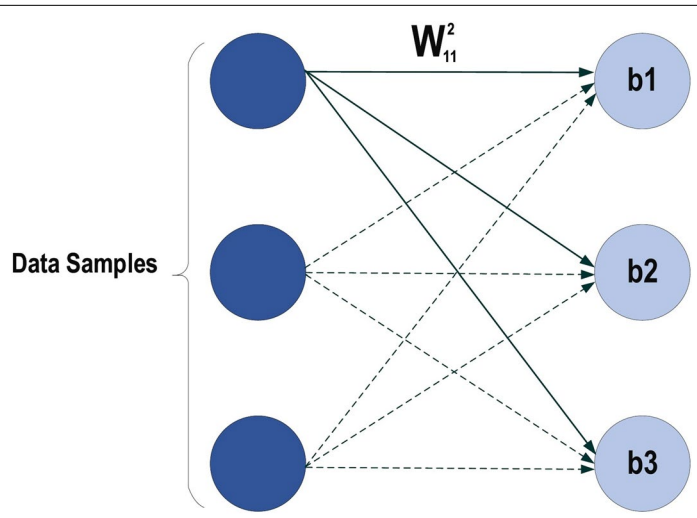

Fig. 12 MLP structure

the number of connection between them is ($10 * 2 = 20$ connection, weights), how the error is defined, and the weight is updated, we will imagine there is there are two layers in our neural network,

$$\text{error } = 1/2\left(\mathbf{d}_i - \mathbf{y}_i\right)^2 \tag{15}$$

where d is the label of induvial input ith and y is the output of the same individual input. Backpropagation is about understanding how to change the weights and biases in a network based on the changes of the cost function (Error). Ultimately, this means computing the partial derivatives $\partial \mathrm{E}/\partial \mathrm{w}_{\mathrm{ij}}^{h}$ and $\partial \mathrm{E}/\partial \mathrm{b}_{\mathrm{j}}^{h}$. But to compute those, a local variable is introduced, δ_j^1 which is called the local error in the jth neuron in the hth layer. Based on that local error Backpropagation will give the procedure to compute $\partial \mathrm{E}/\partial \mathrm{w}_{\mathrm{ij}}^{h}$ and $\partial \mathrm{E}/\partial \mathrm{b}_{\mathrm{j}}^{h}$ how the error is defined, and the weight is updated, we will imagine there is there are two layers in our neural network that is shown in Fig. 13.

Output error for δ_{j}^1 each $1 = 1 : \mathrm{L}$ where L is no. of neuron in output

$$\delta_{\mathrm{j}}^1(\mathbf{k}) = (-1)\mathrm{e}(\mathrm{k})\boldsymbol{\vartheta}'\left(v_j(\boldsymbol{k})\right) \tag{16}$$

where e(k) is the error of the epoch k as shown in Eq. (2) and $\boldsymbol{\vartheta}'\left(\boldsymbol{v}_j(\boldsymbol{k})\right)$ is the derivate of the activation function for v_j at the output.

Backpropagate the error at all the rest layer except the output

$$\delta_{\mathrm{j}}^{\mathrm{h}}(\mathbf{k}) = \boldsymbol{\vartheta}'\left(v_j(\boldsymbol{k})\right) \sum_{l=1}^{L} \delta_{\mathrm{j}}^1 \quad \boldsymbol{w}_{jl}^{h+1}(\boldsymbol{k}) \tag{17}$$

where $\delta_j^1(\mathbf{k})$ is the output error and $w_{jl}^{h+1}(k)$ is represented the weight after the layer where the error need to obtain.

After finding the error at each neuron in each layer, now we can update the weight in each layer based on Eqs. (16) and (17).

Improving performance of CNN

Based on our experiments in different DL applications [86–88]. We can conclude the most active solutions that may improve the performance of CNN are:

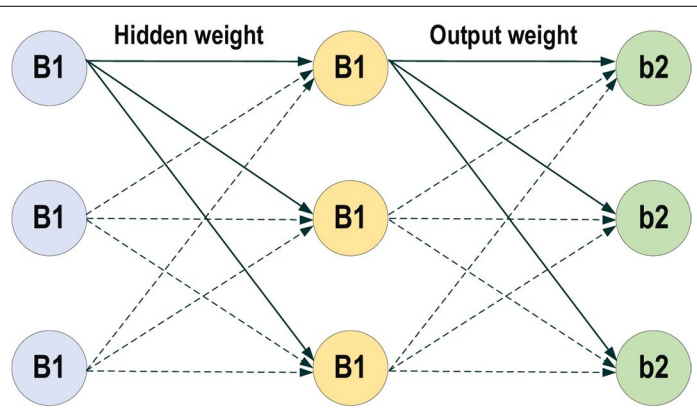

Fig. 13 Neuron activation functions

- Expand the dataset with data augmentation or use transfer learning (explained in latter sections).
- Increase the training time.
- Increase the depth (or width) of the model.
- Add regularization.
- Increase hyperparameters tuning.

CNN architectures

Over the last 10 years, several CNN architectures have been presented [21, 26]. Model architecture is a critical factor in improving the performance of different applications. Various modifications have been achieved in CNN architecture from 1989 until today. Such modifications include structural reformulation, regularization, parameter optimizations, etc. Conversely, it should be noted that the key upgrade in CNN performance occurred largely due to the processing-unit reorganization, as well as the development of novel blocks. In particular, the most novel developments in CNN architectures were performed on the use of network depth. In this section, we review the most popular CNN architectures, beginning from the AlexNet model in 2012 and ending at the High-Resolution (HR) model in 2020. Studying these architectures features (such as input size, depth, and robustness) is the key to help researchers to choose the suitable architecture for the their target task. Table 2 presents the brief overview of CNN architectures.

AlexNet

The history of deep CNNs began with the appearance of LeNet [89] (Fig. 14). At that time, the CNNs were restricted to handwritten digit recognition tasks, which cannot be scaled to all image classes. In deep CNN architecture, AlexNet is highly respected [30], as it achieved innovative results in the fields of image recognition and classification. Krizhevesky et al. [30] first proposed AlexNet and consequently improved the CNN learning ability by increasing its depth and implementing several parameter optimization strategies. Figure 15 illustrates the basic design of the AlexNet architecture.

The learning ability of the deep CNN was limited at this time due to hardware restrictions. To overcome these hardware limitations, two GPUs (NVIDIA GTX 580) were used in parallel to train AlexNet. Moreover, in order to enhance the applicability of the CNN to different image categories, the number of feature extraction stages was increased from five in LeNet to seven in AlexNet. Regardless of the fact that depth enhances generalization for several image resolutions, it was in fact overfitting that represented the main drawback related to the depth. Krizhevesky et al. used Hinton's idea to address this problem [90, 91]. To ensure that the features learned by the algorithm were extra robust, Krizhevesky et al.'s algorithm randomly passes over several transformational units throughout the training stage. Moreover, by reducing the vanishing gradient problem, ReLU [92] could be utilized as a non-saturating activation function to enhance the rate of convergence [93]. Local response normalization and overlapping subsampling were also performed to enhance the generalization by decreasing the overfitting. To improve on the performance of previous networks, other modifications were made by using large-size filters (5×5 and 11×11) in the earlier layers. AlexNet has considerable

Table 2 Brief overview of CNN architectures

Model	Main finding	Depth	Dataset	Error rate	Input size	Year
AlexNet	Utilizes Dropout and ReLU	8	ImageNet	16.4	227 × 227 × 3	2012
NIN	New layer, called 'mlpconv', utilizes GAP	3	CIFAR-10, CIFAR-100, MNIST	10.41, 35.68, 0.45	32 × 32 × 3	2013
ZfNet	Visualization idea of middle layers	8	ImageNet	11.7	224 × 224 × 3	2014
VGG	Increased depth, small filter size	16, 19	ImageNet	7.3	224 × 224 × 3	2014
GoogLeNet	Increased depth,block concept, different filter size, concatenation concept	22	ImageNet	6.7	224 × 224 × 3	2015
Inception-V3	Utilizes small filtersize, better feature representation	48	ImageNet	3.5	229 × 229 × 3	2015
Highway	Presented the multipath concept	19, 32	CIFAR-10	7.76	32 × 32 × 3	2015
Inception-V4	Divided transform and integration concepts	70	ImageNet	3.08	229 × 229 × 3	2016
ResNet	Robust against overfitting due to symmetry mapping-based skip links	152	ImageNet	3.57	224 × 224 × 3	2016
Inception-ResNet-v2	Introduced the concept of residual links	164	ImageNet	3.52	229 × 229 × 3	2016
FractalNet	Introduced the concept of Drop-Path as regularization	40,80	CIFAR-10 CIFAR-100	4.60 18.85	32 × 32 × 3	2016
WideResNet	Decreased the depth and increased the width	28	CIFAR-10 CIFAR-100	3.89 18.85	32 × 32 × 3	2016
Xception	A depthwise convolutionfollowed by a pointwise convolution	71	ImageNet	0.055	229 × 229 × 3	2017
Residual attention neural network	Presented the attention technique	452	CIFAR-10, CIFAR-100	3.90, 20.4	40 × 40 × 3	2017
Squeeze-and-excitation networks	Modeled interdependencies between channels	152	ImageNet	2.25	229 × 229 × 3 224 × 224 × 3 320 × 320 × 3	2017
DenseNet	Blocks of layers; layers connected to each other	201	CIFAR-10, CIFAR-100,ImageNet	3.46, 17.18, 5.54	224 × 224 × 3	2017
Competitive squeeze and excitation network	Both residual and identity mappings utilized to rescale the channel	152	CIFAR-10 CIFAR-100	3.58 18.47	32 × 32 × 3	2018

Table 2 (continued)

Model	Main finding	Depth	Dataset	Error rate	Input size	Year
MobileNet-v2	Inverted residual structure	53	ImageNet	–	224 × 224 × 3	2018
CapsuleNet	Pays attention to special relationships between features	3	MNIST	0.00855	28 × 28 × 1	2018
HRNetV2	High-resolution representations	–	ImageNet	5.4	224 × 224 × 3	2020

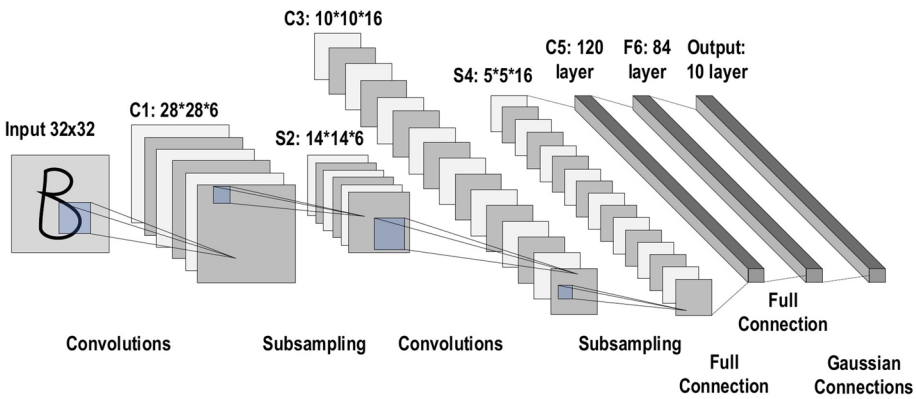

Fig. 14 The architecture of LeNet

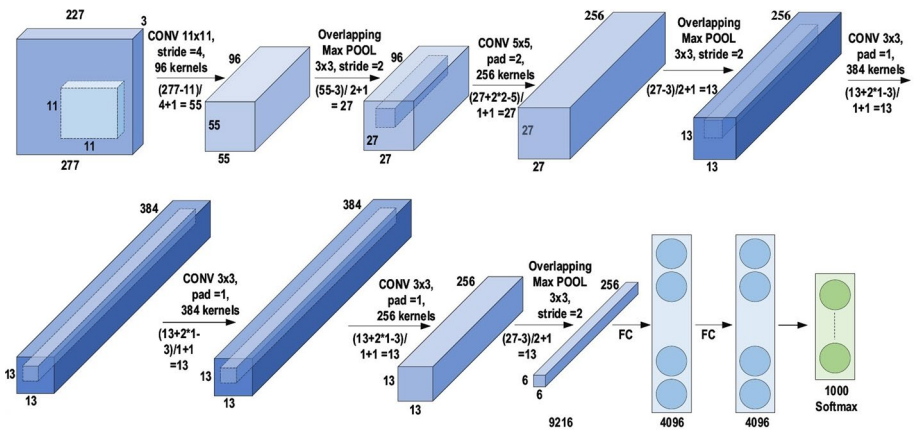

Fig. 15 The architecture of AlexNet

significance in the recent CNN generations, as well as beginning an innovative research era in CNN applications.

Network-in-network

This network model, which has some slight differences from the preceding models, introduced two innovative concepts [94]. The first was employing multiple layers of

perception convolution. These convolutions are executed using a 1×1 filter, which supports the addition of extra nonlinearity in the networks. Moreover, this supports enlarging the network depth, which may later be regularized using dropout. For DL models, this idea is frequently employed in the bottleneck layer. As a substitution for a FC layer, the GAP is also employed, which represents the second novel concept and enables a significant reduction in the number of model parameters. In addition, GAP considerably updates the network architecture. Generating a final low-dimensional feature vector with no reduction in the feature maps dimension is possible when GAP is used on a large feature map [95, 96]. Figure 16 shows the structure of the network.

ZefNet

Before 2013, the CNN learning mechanism was basically constructed on a trial-and-error basis, which precluded an understanding of the precise purpose following the enhancement. This issue restricted the deep CNN performance on convoluted images. In response, Zeiler and Fergus introduced DeconvNet (a multilayer de-convolutional neural network) in 2013 [97]. This method later became known as ZefNet, which was developed in order to quantitively visualize the network. Monitoring the CNN performance via understanding the neuron activation was the purpose of the network activity visualization. However, Erhan et al. utilized this exact concept to optimize deep belief network (DBN) performance by visualizing the features of the hidden layers [98]. Moreover, in addition to this issue, Le et al. assessed the deep unsupervised auto-encoder (AE) performance by visualizing the created classes of the image using the output neurons [99]. By reversing the operation order of the convolutional and pooling layers, DenconvNet operates like a forward-pass CNN. Reverse mapping of this kind launches the convolutional layer output backward to create visually observable image shapes that accordingly give the neural interpretation of the internal feature representation learned at each layer [100]. Monitoring the learning schematic through the training stage was the key concept underlying ZefNet. In addition, it utilized the outcomes to recognize an ability issue coupled with the model. This concept was experimentally proven on AlexNet by applying DeconvNet. This indicated that only certain neurons were working, while the others were out of action in the first two layers of the network. Furthermore, it indicated that the features extracted via the second layer contained aliasing objects. Thus, Zeiler and Fergus changed the CNN topology due to the existence of these outcomes. In addition, they executed parameter optimization, and also exploited the CNN learning by decreasing the stride and the filter sizes in order to retain all features of the initial two convolutional layers. An improvement in performance was accordingly achieved due to this

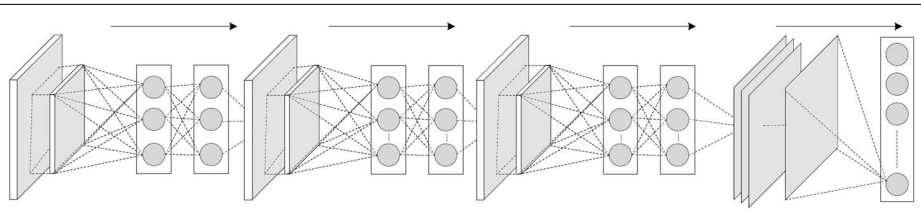

Fig. 16 The architecture of network-in-network

rearrangement in CNN topology. This rearrangement proposed that the visualization of the features could be employed to identify design weaknesses and conduct appropriate parameter alteration. Figure 17 shows the structure of the network.

Visual geometry group (VGG)

After CNN was determined to be effective in the field of image recognition, an easy and efficient design principle for CNN was proposed by Simonyan and Zisserman. This innovative design was called Visual Geometry Group (VGG). A multilayer model [101], it featured nineteen more layers than ZefNet [97] and AlexNet [30] to simulate the relations of the network representational capacity in depth. Conversely, in the 2013-ILSVRC competition, ZefNet was the frontier network, which proposed that filters with small sizes could enhance the CNN performance. With reference to these results, VGG inserted a layer of the heap of 3×3 filters rather than the 5×5 and 11×11 filters in ZefNet. This showed experimentally that the parallel assignment of these small-size filters could produce the same influence as the large-size filters. In other words, these small-size filters made the receptive field similarly efficient to the large-size filters (7×7 and 5×5). By decreasing the number of parameters, an extra advantage of reducing computational complication was achieved by using small-size filters. These outcomes established a novel research trend for working with small-size filters in CNN. In addition, by inserting 1×1 convolutions in the middle of the convolutional layers, VGG regulates the network complexity. It learns a linear grouping of the subsequent feature maps. With respect to network tuning, a max pooling layer [102] is inserted following the convolutional layer, while padding is implemented to maintain the spatial resolution. In general, VGG obtained significant results for localization problems and image classification. While it did not achieve first place in the 2014-ILSVRC competition, it acquired a reputation due to its enlarged depth, homogenous topology, and simplicity. However, VGG's computational cost was excessive due to its utilization of around 140 million parameters, which represented its main shortcoming. Figure 18 shows the structure of the network.

GoogLeNet

In the 2014-ILSVRC competition, GoogleNet (also called Inception-V1) emerged as the winner [103]. Achieving high-level accuracy with decreased computational cost is the core aim of the GoogleNet architecture. It proposed a novel inception block (module) concept in the CNN context, since it combines multiple-scale convolutional transformations by employing merge, transform, and split functions for feature extraction. Figure 19 illustrates the inception block architecture. This architecture incorporates filters

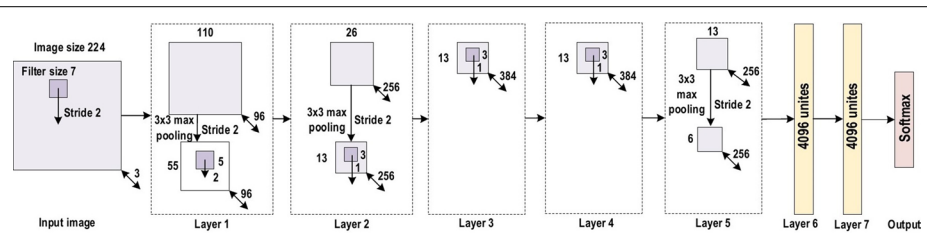

Fig. 17 The architecture of ZefNet

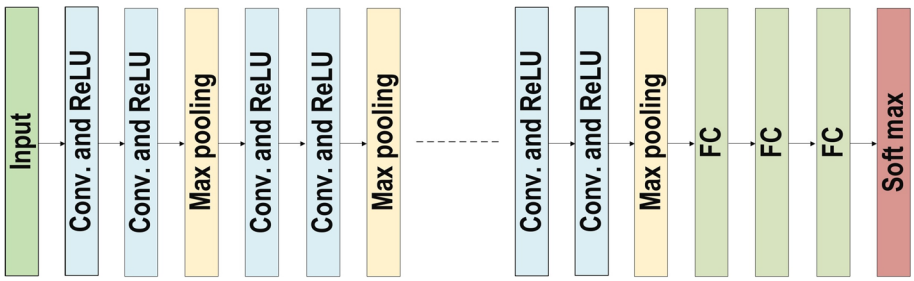

Fig. 18 The architecture of VGG

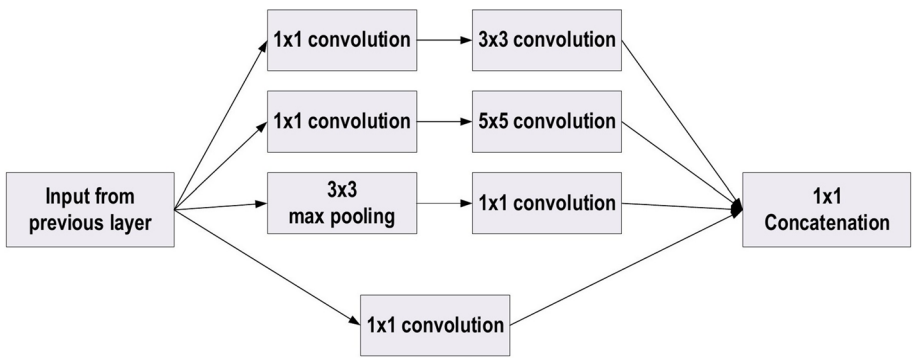

Fig. 19 The basic structure of Google Block

of different sizes ($5 \times 5, 3 \times 3,$ and 1×1) to capture channel information together with spatial information at diverse ranges of spatial resolution. The common convolutional layer of GoogLeNet is substituted by small blocks using the same concept of network-in-network (NIN) architecture [94], which replaced each layer with a micro-neural network. The GoogLeNet concepts of merge, transform, and split were utilized, supported by attending to an issue correlated with different learning types of variants existing in a similar class of several images. The motivation of GoogLeNet was to improve the efficiency of CNN parameters, as well as to enhance the learning capacity. In addition, it regulates the computation by inserting a 1×1 convolutional filter, as a bottleneck layer, ahead of using large-size kernels. GoogleNet employed sparse connections to overcome the redundant information problem. It decreased cost by neglecting the irrelevant channels. It should be noted here that only some of the input channels are connected to some of the output channels. By employing a GAP layer as the end layer, rather than utilizing a FC layer, the density of connections was decreased. The number of parameters was also significantly decreased from 40 to 5 million parameters due to these parameter tunings. The additional regularity factors used included the employment of RmsProp as optimizer and batch normalization [104]. Furthermore, GoogleNet proposed the idea of auxiliary learners to speed up the rate of convergence. Conversely, the main shortcoming of GoogleNet was its heterogeneous topology; this shortcoming requires adaptation from one module to another. Other shortcomings of GoogleNet include the representation

jam, which substantially decreased the feature space in the following layer, and in turn occasionally leads to valuable information loss.

Highway network

Increasing the network depth enhances its performance, mainly for complicated tasks. By contrast, the network training becomes difficult. The presence of several layers in deeper networks may result in small gradient values of the back-propagation of error at lower layers. In 2015, Srivastava et al. [105] suggested a novel CNN architecture, called Highway Network, to overcome this issue. This approach is based on the cross-connectivity concept. The unhindered information flow in Highway Network is empowered by instructing two gating units inside the layer. The gate mechanism concept was motivated by LSTM-based RNN [106, 107]. The information aggregation was conducted by merging the information of the ıth $-k$ layers with the next ıth layer to generate a regularization impact, which makes the gradient-based training of the deeper network very simple. This empowers the training of networks with more than 100 layers, such as a deeper network of 900 layers with the SGD algorithm. A Highway Network with a depth of fifty layers presented an improved rate of convergence, which is better than thin and deep architectures at the same time [108]. By contrast, [69] empirically demonstrated that plain Net performance declines when more than ten hidden layers are inserted. It should be noted that even a Highway Network 900 layers in depth converges much more rapidly than the plain network.

ResNet

He et al. [37] developed ResNet (Residual Network), which was the winner of ILSVRC 2015. Their objective was to design an ultra-deep network free of the vanishing gradient issue, as compared to the previous networks. Several types of ResNet were developed based on the number of layers (starting with 34 layers and going up to 1202 layers). The most common type was ResNet50, which comprised 49 convolutional layers plus a single FC layer. The overall number of network weights was 25.5 M, while the overall number of MACs was 3.9 M. The novel idea of ResNet is its use of the bypass pathway concept, as shown in Fig. 20, which was employed in Highway Nets to address the problem of training a deeper network in 2015. This is illustrated in Fig. 20, which contains the fundamental ResNet block diagram. This is a conventional feed-forward network plus a residual connection. The residual layer output can be identified as the $(l-1)$th outputs, which are delivered from the preceding layer $(x_l - 1)$. After executing different operations [such as convolution using variable-size filters, or batch normalization, before applying an activation function like ReLU on $(x_l - 1)$], the output is $F(x_l - 1)$. The ending residual output is x_l, which can be mathematically represented as in Eq. 18.

$$x_l = F(x_l - 1) + x_l - 1 \tag{18}$$

There are numerous basic residual blocks included in the residual network. Based on the type of the residual network architecture, operations in the residual block are also changed [37].

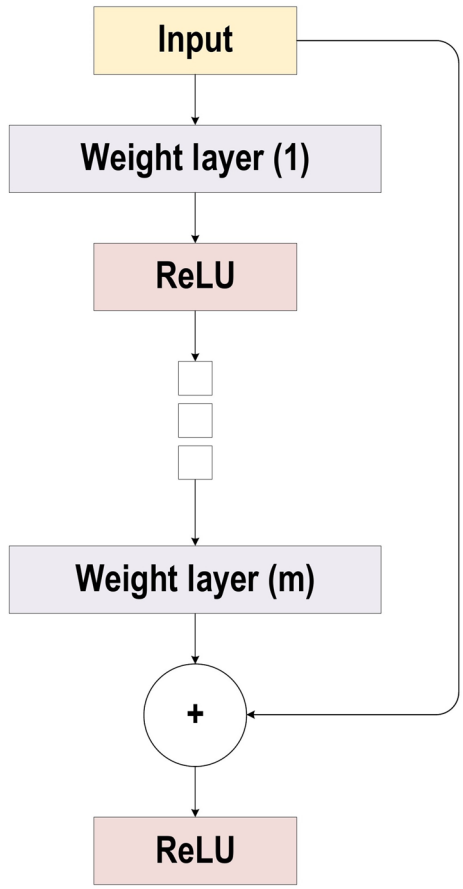

Fig. 20 The block diagram for ResNet

In comparison to the highway network, ResNet presented shortcut connections inside layers to enable cross-layer connectivity, which are parameter-free and data-independent. Note that the layers characterize non-residual functions when a gated shortcut is closed in the highway network. By contrast, the individuality shortcuts are never closed, while the residual information is permanently passed in ResNet. Furthermore, ResNet has the potential to prevent the problems of gradient diminishing, as the shortcut connections (residual links) accelerate the deep network convergence. ResNet was the winner of the 2015-ILSVRC championship with 152 layers of depth; this represents 8 times the depth of VGG and 20 times the depth of AlexNet. In comparison with VGG, it has lower computational complexity, even with enlarged depth.

Inception: ResNet and Inception-V3/4

Szegedy et al. [103, 109, 110] proposed Inception-ResNet and Inception-V3/4 as upgraded types of Inception-V1/2. The concept behind Inception-V3 was to minimize the computational cost with no effect on the deeper network generalization. Thus, Szegedy et al. used asymmetric small-size filters (1×5 and 1×7) rather than large-size filters (7×7 and 5×5); moreover, they utilized a bottleneck of 1×1 convolution prior to the large-size filters [110]. These changes make the operation of the traditional

convolution very similar to cross-channel correlation. Previously, Lin et al. utilized the 1×1 filter potential in NIN architecture [94]. Subsequently, [110] utilized the same idea in an intelligent manner. By using 1×1 convolutional operation in Inception-V3, the input data are mapped into three or four isolated spaces, which are smaller than the initial input spaces. Next, all of these correlations are mapped in these smaller spaces through common 5×5 or 3×3 convolutions. By contrast, in Inception-ResNet, Szegedy et al. bring together the inception block and the residual learning power by replacing the filter concatenation with the residual connection [111]. Szegedy et al. empirically demonstrated that Inception-ResNet (Inception-4 with residual connections) can achieve a similar generalization power to Inception-V4 with enlarged width and depth and without residual connections. Thus, it is clearly illustrated that using residual connections in training will significantly accelerate the Inception network training. Figure 21 shows The basic block diagram for Inception Residual unit.

DenseNet

To solve the problem of the vanishing gradient, DenseNet was presented, following the same direction as ResNet and the Highway network [105, 111, 112]. One of the drawbacks of ResNet is that it clearly conserves information by means of preservative individuality transformations, as several layers contribute extremely little or no information. In addition, ResNet has a large number of weights, since each layer has an isolated group of weights. DenseNet employed cross-layer connectivity in an improved approach to address this problem [112–114]. It connected each layer to all layers in the network using a feed-forward approach. Therefore, the feature maps of each previous layer were employed to input into all of the following layers. In traditional CNNs, there are l connections between the previous layer and the current layer, while in DenseNet, there are $\frac{l(l+1)}{2}$ direct connections. DenseNet demonstrates the influence of cross-layer depth

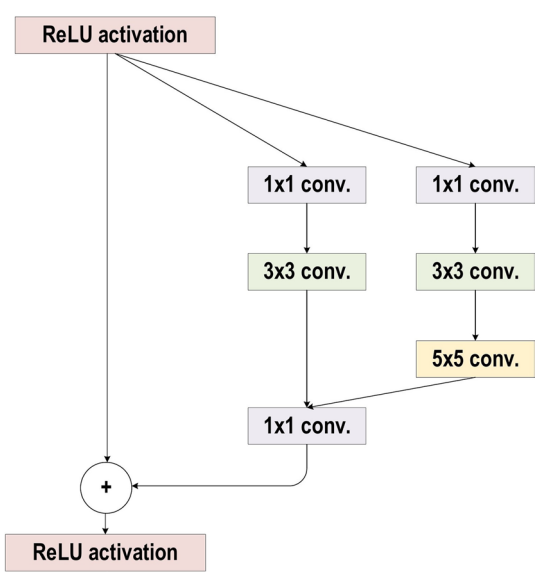

Fig. 21 The basic block diagram for Inception Residual unit

wise-convolutions. Thus, the network gains the ability to discriminate clearly between the added and the preserved information, since DenseNet concatenates the features of the preceding layers rather than adding them. However, due to its narrow layer structure, DenseNet becomes parametrically high-priced in addition to the increased number of feature maps. The direct admission of all layers to the gradients via the loss function enhances the information flow all across the network. In addition, this includes a regularizing impact, which minimizes overfitting on tasks alongside minor training sets. Figure 22 shows the architecture of DenseNet Network.

ResNext

ResNext is an enhanced version of the Inception Network [115]. It is also known as the Aggregated Residual Transform Network. Cardinality, which is a new term presented by [115], utilized the split, transform, and merge topology in an easy and effective way. It denotes the size of the transformation set as an extra dimension [116–118]. However, the Inception network manages network resources more efficiently, as well as enhancing the learning ability of the conventional CNN. In the transformation branch, different spatial embeddings (employing e.g. 5×5, 3×3, and 1×1) are used. Thus, customizing each layer is required separately. By contrast, ResNext derives its characteristic features from ResNet, VGG, and Inception. It employed the VGG deep homogenous topology with the basic architecture of GoogleNet by setting 3×3 filters as spatial resolution inside the blocks of split, transform, and merge. Figure 23 shows the ResNext building blocks. ResNext utilized multi-transformations inside the blocks of split, transform, and merge, as well as outlining such transformations in cardinality terms. The performance is significantly improved by increasing the cardinality, as Xie et al. showed. The complexity

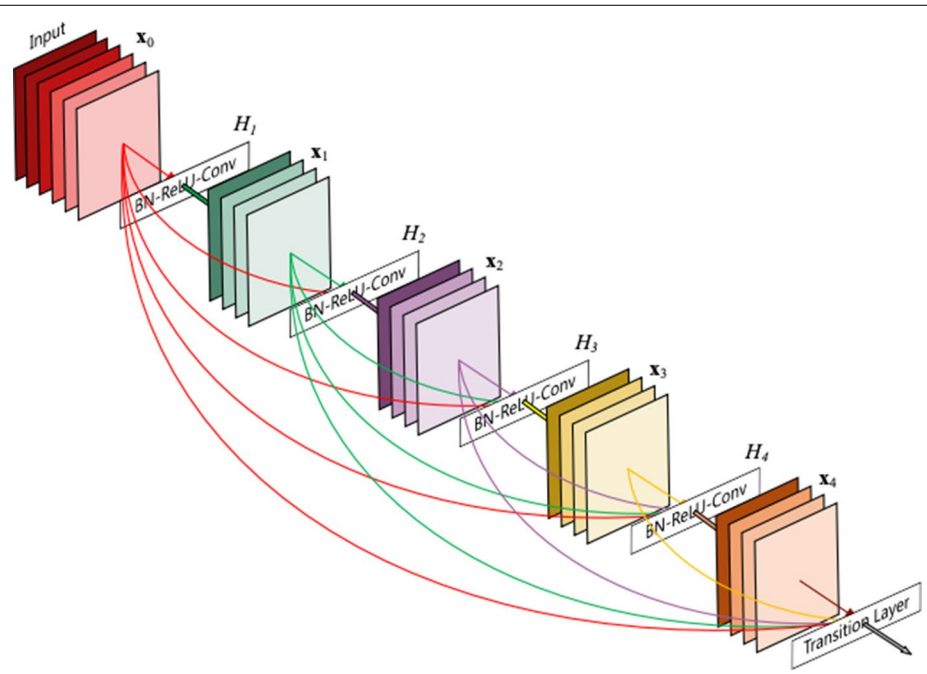

Fig. 22 The architecture of DenseNet Network (adopted from [112])

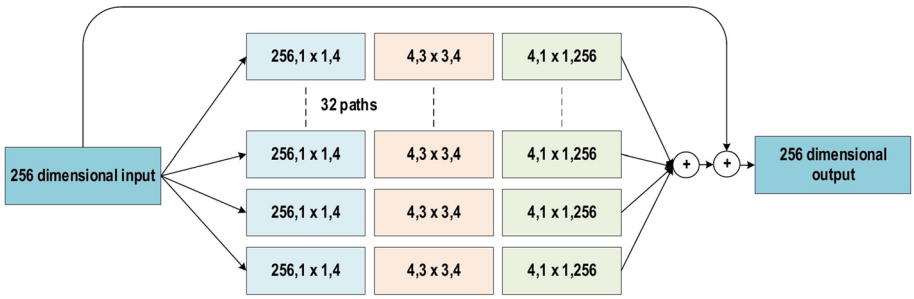

Fig. 23 The basic block diagram for the ResNext building blocks

of ResNext was regulated by employing 1×1 filters (low embeddings) ahead of a 3×3 convolution. By contrast, skipping connections are used for optimized training [115].

WideResNet

The feature reuse problem is the core shortcoming related to deep residual networks, since certain feature blocks or transformations contribute a very small amount to learning. Zagoruyko and Komodakis [119] accordingly proposed WideResNet to address this problem. These authors advised that the depth has a supplemental influence, while the residual units convey the core learning ability of deep residual networks. WideResNet utilized the residual block power via making the ResNet wider instead of deeper [37]. It enlarged the width by presenting an extra factor, k, which handles the network width. In other words, it indicated that layer widening is a highly successful method of performance enhancement compared to deepening the residual network. While enhanced representational capacity is achieved by deep residual networks, these networks also have certain drawbacks, such as the exploding and vanishing gradient problems, feature reuse problem (inactivation of several feature maps), and the time-intensive nature of the training. He et al. [37] tackled the feature reuse problem by including a dropout in each residual block to regularize the network in an efficient manner. In a similar manner, utilizing dropouts, Huang et al. [120] presented the stochastic depth concept to solve the slow learning and gradient vanishing problems. Earlier research was focused on increasing the depth; thus, any small enhancement in performance required the addition of several new layers. When comparing the number of parameters, WideResNet has twice that of ResNet, as an experimental study showed. By contrast, WideResNet presents an improved method for training relative to deep networks [119]. Note that most architectures prior to residual networks (including the highly effective VGG and Inception) were wider than ResNet. Thus, wider residual networks were established once this was determined. However, inserting a dropout between the convolutional layers (as opposed to within the residual block) made the learning more effective in WideResNet [121, 122].

Pyramidal Net

The depth of the feature map increases in the succeeding layer due to the deep stacking of multi-convolutional layers, as shown in previous deep CNN architectures such as

ResNet, VGG, and AlexNet. By contrast, the spatial dimension reduces, since a sub-sampling follows each convolutional layer. Thus, augmented feature representation is recompensed by decreasing the size of the feature map. The extreme expansion in the depth of the feature map, alongside the spatial information loss, interferes with the learning ability in the deep CNNs. ResNet obtained notable outcomes for the issue of image classification. Conversely, deleting a convolutional block—in which both the number of channel and spatial dimensions vary (channel depth enlarges, while spatial dimension reduces)—commonly results in decreased classifier performance. Accordingly, the stochastic ResNet enhanced the performance by decreasing the information loss accompanying the residual unit drop. Han et al. [123] proposed Pyramidal Net to address the ResNet learning interference problem. To address the depth enlargement and extreme reduction in spatial width via ResNet, Pyramidal Net slowly enlarges the residual unit width to cover the most feasible places rather than saving the same spatial dimension inside all residual blocks up to the appearance of the down-sampling. It was referred to as Pyramidal Net due to the slow enlargement in the feature map depth based on the up-down method. Factor l, which was determined by Eq. 19, regulates the depth of the feature map.

$$d_l = \left\{ \begin{matrix} 16 & if\ l = 1 \\ \left\lfloor d_{l-1} + \frac{\lambda}{n} \right\rfloor & if\ 2 \le l \le n+1 \end{matrix} \right\} \tag{19}$$

Here, the dimension of the lth residual unit is indicated by d_l; moreover, n indicates the overall number of residual units, the step factor is indicated by λ, and the depth increase is regulated by the factor $\frac{\lambda}{n}$, which uniformly distributes the weight increase across the dimension of the feature map. Zero-padded identity mapping is used to insert the residual connections among the layers. In comparison to the projection-based shortcut connections, zero-padded identity mapping requires fewer parameters, which in turn leads to enhanced generalization [124]. Multiplication- and addition-based widening are two different approaches used in Pyramidal Nets for network widening. More specifically, the first approach (multiplication) enlarges geometrically, while the second one (addition) enlarges linearly [92]. The main problem associated with the width enlargement is the growth in time and space required related to the quadratic time.

Xception

Extreme inception architecture is the main characteristic of Xception. The main idea behind Xception is its depthwise separable convolution [125]. The Xception model adjusted the original inception block by making it wider and exchanging a single dimension (3×3) followed by a 1×1 convolution to reduce computational complexity. Figure 24 shows the Xception block architecture. The Xception network becomes extra computationally effective through the use of the decoupling channel and spatial correspondence. Moreover, it first performs mapping of the convolved output to the embedding short dimension by applying 1×1 convolutions. It then performs k spatial transformations. Note that k here represents the width-defining cardinality, which is obtained via the transformations number in Xception. However, the computations were made simpler in Xception by distinctly convolving each channel around the spatial axes.

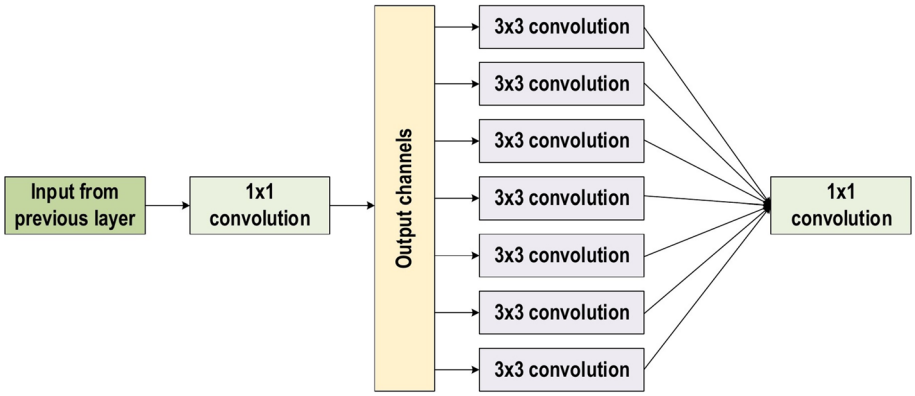

Fig. 24 The basic block diagram for the Xception block architecture

These axes are subsequently used as the 1×1 convolutions (pointwise convolution) for performing cross-channel correspondence. The 1×1 convolution is utilized in Xception to regularize the depth of the channel. The traditional convolutional operation in Xception utilizes a number of transformation segments equivalent to the number of channels; Inception, moreover, utilizes three transformation segments, while traditional CNN architecture utilizes only a single transformation segment. Conversely, the suggested Xception transformation approach achieves extra learning efficiency and better performance but does not minimize the number of parameters [126, 127].

Residual attention neural network

To improve the network feature representation, Wang et al. [128] proposed the Residual Attention Network (RAN). Enabling the network to learn aware features of the object is the main purpose of incorporating attention into the CNN. The RAN consists of stacked residual blocks in addition to the attention module; hence, it is a feed-forward CNN. However, the attention module is divided into two branches, namely the mask branch and trunk branch. These branches adopt a top-down and bottom-up learning strategy respectively. Encapsulating two different strategies in the attention model supports top-down attention feedback and fast feed-forward processing in only one particular feed-forward process. More specifically, the top-down architecture generates dense features to make inferences about every aspect. Moreover, the bottom-up feedforward architecture generates low-resolution feature maps in addition to robust semantic information. Restricted Boltzmann machines employed a top-down bottom-up strategy as in previously proposed studies [129]. During the training reconstruction phase, Goh et al. [130] used the mechanism of top-down attention in deep Boltzmann machines (DBMs) as a regularizing factor. Note that the network can be globally optimized using a top-down learning strategy in a similar manner, where the maps progressively output to the input throughout the learning process [129–132].

Incorporating the attention concept with convolutional blocks in an easy way was used by the transformation network, as obtained in a previous study [133]. Unfortunately, these are inflexible, which represents the main problem, along with their inability to be

used for varying surroundings. By contrast, stacking multi-attention modules has made RAN very effective at recognizing noisy, complex, and cluttered images. RAN's hierarchical organization gives it the capability to adaptively allocate a weight for every feature map depending on its importance within the layers. Furthermore, incorporating three distinct levels of attention (spatial, channel, and mixed) enables the model to use this ability to capture the object-aware features at these distinct levels.

Convolutional block attention module

The importance of the feature map utilization and the attention mechanism is certified via SE-Network and RAN [128, 134, 135]. The convolutional block attention (CBAM) module, which is a novel attention-based CNN, was first developed by Woo et al. [136]. This module is similar to SE-Network and simple in design. SE-Network disregards the object's spatial locality in the image and considers only the channels' contribution during the image classification. Regarding object detection, object spatial location plays a significant role. The convolutional block attention module sequentially infers the attention maps. More specifically, it applies channel attention preceding the spatial attention to obtain the refined feature maps. Spatial attention is performed using 1×1 convolution and pooling functions, as in the literature. Generating an effective feature descriptor can be achieved by using a spatial axis along with the pooling of features. In addition, generating a robust spatial attention map is possible, as CBAM concatenates the max pooling and average pooling operations. In a similar manner, a collection of GAP and max pooling operations is used to model the feature map statistics. Woo et al. [136] demonstrated that utilizing GAP will return a sub-optimized inference of channel attention, whereas max pooling provides an indication of the distinguishing object features. Thus, the utilization of max pooling and average pooling enhances the network's representational power. The feature maps improve the representational power, as well as facilitating a focus on the significant portion of the chosen features. The expression of 3D attention maps through a serial learning procedure assists in decreasing the computational cost and the number of parameters, as Woo et al. [136] experimentally proved. Note that any CNN architecture can be simply integrated with CBAM.

Concurrent spatial and channel excitation mechanism

To make the work valid for segmentation tasks, Roy et al. [137, 138] expanded Hu et al. [134] effort by adding the influence of spatial information to the channel information. Roy et al. [137, 138] presented three types of modules: (1) channel squeeze and excitation with concurrent channels (scSE); (2) exciting spatially and squeezing channel-wise (sSE); (3) exciting channel-wise and squeezing spatially (cSE). For segmentation purposes, they employed auto-encoder-based CNNs. In addition, they suggested inserting modules following the encoder and decoder layers. To specifically highlight the object-specific feature maps, they further allocated attention to every channel by expressing a scaling factor from the channel and spatial information in the first module (scSE). In the second module (sSE), the feature map information has lower importance than the spatial locality, as the spatial information plays a significant role during the segmentation process. Therefore, several channel collections are spatially divided and developed so that

they can be employed in segmentation. In the final module (cSE), a similar SE-block concept is used. Furthermore, the scaling factor is derived founded on the contribution of the feature maps within the object detection [137, 138].

CapsuleNet

CNN is an efficient technique for detecting object features and achieving well-behaved recognition performance in comparison with innovative handcrafted feature detectors. A number of restrictions related to CNN are present, meaning that the CNN does not consider certain relations, orientation, size, and perspectives of features. For instance, when considering a face image, the CNN does not count the various face components (such as mouth, eyes, nose, etc.) positions, and will incorrectly activate the CNN neurons and recognize the face without taking specific relations (such as size, orientation etc.) into account. At this point, consider a neuron that has probability in addition to feature properties such as size, orientation, perspective, etc. A specific neuron/capsule of this type has the ability to effectively detect the face along with different types of information. Thus, many layers of capsule nodes are used to construct the capsule network. An encoding unit, which contains three layers of capsule nodes, forms the CapsuleNet or CapsNet (the initial version of the capsule networks).

For example, the MNIST architecture comprises 28×28 images, applying 256 filters of size 9×9 and with stride 1. The $28 - 9 + 1 = 20$ is the output plus 256 feature maps. Next, these outputs are input to the first capsule layer, while producing an 8D vector rather than a scalar; in fact, this is a modified convolution layer. Note that a stride 2 with 9×9 filters is employed in the first convolution layer. Thus, the dimension of the output is $(20 - 9)/2 + 1 = 6$. The initial capsules employ 8×32 filters, which generate $32 \times 8 \times 6 \times 6$ (32 for groups, 8 for neurons, while 6×6 is the neuron size).

Figure 25 represents the complete CapsNet encoding and decoding processes. In the CNN context, a max-pooling layer is frequently employed to handle the translation change. It can detect the feature moves in the event that the feature is still within the max-pooling window. This approach has the ability to detect the overlapped features;

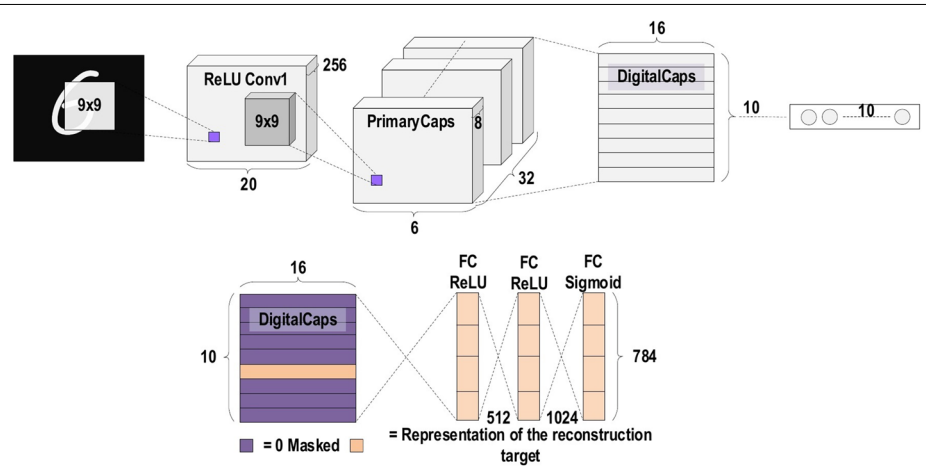

Fig. 25 The complete CapsNet encoding and decoding processes

this is highly significant in detection and segmentation operations, since the capsule involves the weighted features sum from the preceding layer.

In conventional CNNs, a particular cost function is employed to evaluate the global error that grows toward the back throughout the training process. Conversely, in such cases, the activation of a neuron will not grow further once the weight between two neurons turns out to be zero. Instead of a single size being provided with the complete cost function in repetitive dynamic routing alongside the agreement, the signal is directed based on the feature parameters. Sabour et al. [139] provides more details about this architecture. When using MNIST to recognize handwritten digits, this innovative CNN architecture gives superior accuracy. From the application perspective, this architecture has extra suitability for segmentation and detection approaches when compared with classification approaches [140–142].

High-resolution network (HRNet)

High-resolution representations are necessary for position-sensitive vision tasks, such as semantic segmentation, object detection, and human pose estimation. In the present up-to-date frameworks, the input image is encoded as a low-resolution representation using a subnetwork that is constructed as a connected series of high-to-low resolution convolutions such as VGGNet and ResNet. The low-resolution representation is then recovered to become a high-resolution one. Alternatively, high-resolution representations are maintained during the entire process using a novel network, referred to as a High-Resolution Network (HRNet) [143, 144]. This network has two principal features. First, the convolution series of high-to-low resolutions are connected in parallel. Second, the information across the resolutions are repeatedly exchanged. The advantage achieved includes getting a representation that is more accurate in the spatial domain and extra-rich in the semantic domain. Moreover, HRNet has several applications in the fields of object detection, semantic segmentation, and human pose prediction. For computer vision problems, the HRNet represents a more robust backbone. Figure 26 illustrates the general architecture of HRNet.

Challenges (limitations) of deep learning and alternate solutions

When employing DL, several difficulties are often taken into consideration. Those more challenging are listed next and several possible alternatives are accordingly provided.

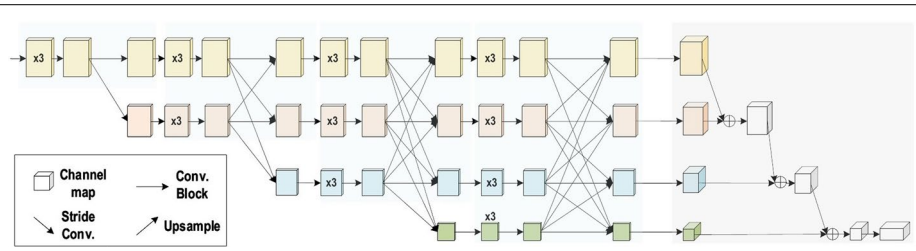

Fig. 26 The general architecture of HRNet

Training data

DL is extremely data-hungry considering it also involves representation learning [145, 146]. DL demands an extensively large amount of data to achieve a well-behaved performance model, i.e. as the data increases, an extra well-behaved performance model can be achieved (Fig. 27). In most cases, the available data are sufficient to obtain a good performance model. However, sometimes there is a shortage of data for using DL directly [87]. To properly address this issue, three suggested methods are available. The first involves the employment of the transfer-learning concept after data is collected from similar tasks. Note that while the transferred data will not directly augment the actual data, it will help in terms of both enhancing the original input representation of data and its mapping function [147]. In this way, the model performance is boosted. Another technique involves employing a well-trained model from a similar task and fine-tuning the ending of two layers or even one layer based on the limited original data. Refer to [148, 149] for a review of different transfer-learning techniques applied in the DL approach. In the second method, data augmentation is performed [150]. This task is very helpful for use in augmenting the image data, since the image translation, mirroring, and rotation commonly do not change the image label. Conversely, it is important to take care when applying this technique in some cases such as with bioinformatics data. For instance, when mirroring an enzyme sequence, the output data may not represent the actual enzyme sequence. In the third method, the simulated data can be considered for increasing the volume of the training set. It is occasionally possible to create simulators based on the physical process if the issue is well understood. Therefore, the result will involve the simulation of as much data as needed. Processing the data requirement for DL-based simulation is obtained as an example in Ref. [151].

Transfer learning

Recent research has revealed a widespread use of deep CNNs, which offer groundbreaking support for answering many classification problems. Generally speaking, deep CNN models require a sizable volume of data to obtain good performance. The

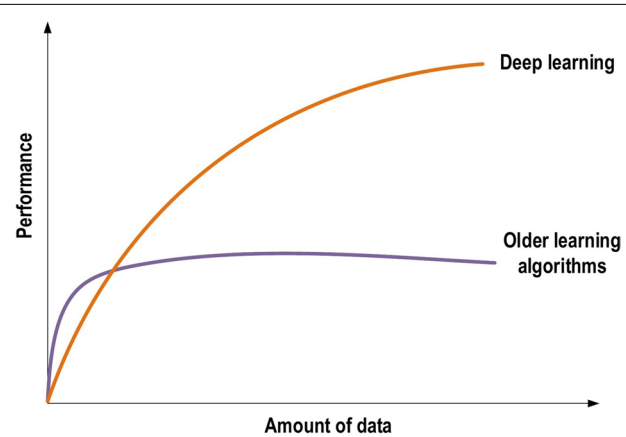

Fig. 27 The performance of DL regarding the amount of data

common challenge associated with using such models concerns the lack of training data. Indeed, gathering a large volume of data is an exhausting job, and no successful solution is available at this time. The undersized dataset problem is therefore currently solved using the TL technique [148, 149], which is highly efficient in addressing the lack of training data issue. The mechanism of TL involves training the CNN model with large volumes of data. In the next step, the model is fine-tuned for training on a small request dataset.

The student-teacher relationship is a suitable approach to clarifying TL. Gathering detailed knowledge of the subject is the first step [152]. Next, the teacher provides a "course" by conveying the information within a "lecture series" over time. Put simply, the teacher transfers the information to the student. In more detail, the expert (teacher) transfers the knowledge (information) to the learner (student). Similarly, the DL network is trained using a vast volume of data, and also learns the bias and the weights during the training process. These weights are then transferred to different networks for retraining or testing a similar novel model. Thus, the novel model is enabled to pre-train weights rather than requiring training from scratch. Figure 28 illustrates the conceptual diagram of the TL technique.

1. Pre-trained models: Many CNN models, e.g. AlexNet [30], GoogleNet [103], and ResNet [37], have been trained on large datasets such as ImageNet for image recognition purposes. These models can then be employed to recognize a different task without the need to train from scratch. Furthermore, the weights remain the same apart from a few learned features. In cases where data samples are lacking, these models are very useful. There are many reasons for employing a pre-trained model. First, training large models on sizeable datasets requires high-priced computational power. Second, training large models can be time-consuming, taking up to multiple weeks. Finally, a pre-trained model can assist with network generalization and speed up the convergence.

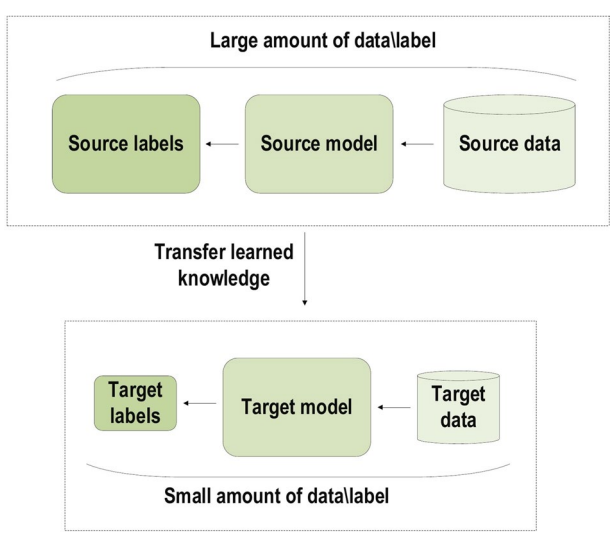

Fig. 28 The conceptual diagram of the TL technique

2. A research problem using pre-trained models: Training a DL approach requires a massive number of images. Thus, obtaining good performance is a challenge under these circumstances. Achieving excellent outcomes in image classification or recognition applications, with performance occasionally superior to that of a human, becomes possible through the use of deep convolutional neural networks (DCNNs) including several layers if a huge amount of data is available [37, 148, 153]. However, avoiding overfitting problems in such applications requires sizable datasets and properly generalizing DCNN models. When training a DCNN model, the dataset size has no lower limit. However, the accuracy of the model becomes insufficient in the case of the utilized model has fewer layers, or if a small dataset is used for training due to over- or under-fitting problems. Due to they have no ability to utilize the hierarchical features of sizable datasets, models with fewer layers have poor accuracy. It is difficult to acquire sufficient training data for DL models. For example, in medical imaging and environmental science, gathering labelled datasets is very costly [148]. Moreover, the majority of the crowdsourcing workers are unable to make accurate notes on medical or biological images due to their lack of medical or biological knowledge. Thus, ML researchers often rely on field experts to label such images; however, this process is costly and time consuming. Therefore, producing the large volume of labels required to develop flourishing deep networks turns out to be unfeasible. Recently, TL has been widely employed to address the later issue. Nevertheless, although TL enhances the accuracy of several tasks in the fields of pattern recognition and computer vision [154, 155], there is an essential issue related to the source data type used by the TL as compared to the target dataset. For instance, enhancing the medical image classification performance of CNN models is achieved by training the models using the ImageNet dataset, which contains natural images [153]. However, such natural images are completely dissimilar from the raw medical images, meaning that the model performance is not enhanced. It has further been proven that TL from different domains does not significantly affect performance on medical imaging tasks, as lightweight models trained from scratch perform nearly as well as standard ImageNet-transferred models [156]. Therefore, there exists scenarios in which using pre-trained models do not become an affordable solution. In 2020, some researchers have utilized same-domain TL and achieved excellent results [86–88, 157]. Same-domain TL is an approach of using images that look similar to the target dataset for training. For example, using X-ray images of different chest diseases to train the model, then fine-tuning and training it on chest X-ray images for COVID-19 diagnosis. More details about same-domain TL and how to implement the fine-tuning process can be found in [87].

Data augmentation techniques

If the goal is to increase the amount of available data and avoid the overfitting issue, data augmentation techniques are one possible solution [150, 158, 159]. These techniques are data-space solutions for any limited-data problem. Data augmentation incorporates a collection of methods that improve the attributes and size of training datasets. Thus, DL

networks can perform better when these techniques are employed. Next, we list some data augmentation alternate solutions.

1. Flipping: Flipping the vertical axis is a less common practice than flipping the horizontal one. Flipping has been verified as valuable on datasets like ImageNet and CIFAR-10. Moreover, it is highly simple to implement. In addition, it is not a label-conserving transformation on datasets that involve text recognition (such as SVHN and MNIST).
2. Color space: Encoding digital image data is commonly used as a dimension tensor ($height \times width \times colorchannels$). Accomplishing augmentations in the color space of the channels is an alternative technique, which is extremely workable for implementation. A very easy color augmentation involves separating a channel of a particular color, such as Red, Green, or Blue. A simple way to rapidly convert an image using a single-color channel is achieved by separating that matrix and inserting additional double zeros from the remaining two color channels. Furthermore, increasing or decreasing the image brightness is achieved by using straightforward matrix operations to easily manipulate the RGB values. By deriving a color histogram that describes the image, additional improved color augmentations can be obtained. Lighting alterations are also made possible by adjusting the intensity values in histograms similar to those employed in photo-editing applications.
3. Cropping: Cropping a dominant patch of every single image is a technique employed with combined dimensions of height and width as a specific processing step for image data. Furthermore, random cropping may be employed to produce an impact similar to translations. The difference between translations and random cropping is that translations conserve the spatial dimensions of this image, while random cropping reduces the input size [for example from (256, 256) to (224, 224)]. According to the selected reduction threshold for cropping, the label-preserving transformation may not be addressed.
4. Rotation: When rotating an image left or right from within 0 to 360 degrees around the axis, rotation augmentations are obtained. The rotation degree parameter greatly determines the suitability of the rotation augmentations. In digit recognition tasks, small rotations (from 0 to 20 degrees) are very helpful. By contrast, the data label cannot be preserved post-transformation when the rotation degree increases.
5. Translation: To avoid positional bias within the image data, a very useful transformation is to shift the image up, down, left, or right. For instance, it is common that the whole dataset images are centered; moreover, the tested dataset should be entirely made up of centered images to test the model. Note that when translating the initial images in a particular direction, the residual space should be filled with Gaussian or random noise, or a constant value such as 255 s or 0 s. The spatial dimensions of the image post-augmentation are preserved using this padding.
6. Noise injection This approach involves injecting a matrix of arbitrary values. Such a matrix is commonly obtained from a Gaussian distribution. Moreno-Barea et al. [160] employed nine datasets to test the noise injection. These datasets were taken from the UCI repository [161]. Injecting noise within images enables the CNN to learn additional robust features.

However, highly well-behaved solutions for positional biases available within the training data are achieved by means of geometric transformations. To separate the distribution of the testing data from the training data, several prospective sources of bias exist. For instance, when all faces should be completely centered within the frames (as in facial recognition datasets), the problem of positional biases emerges. Thus, geometric translations are the best solution. Geometric translations are helpful due to their simplicity of implementation, as well as their effective capability to disable the positional biases. Several libraries of image processing are available, which enables beginning with simple operations such as rotation or horizontal flipping. Additional training time, higher computational costs, and additional memory are some shortcomings of geometric transformations. Furthermore, a number of geometric transformations (such as arbitrary cropping or translation) should be manually observed to ensure that they do not change the image label. Finally, the biases that separate the test data from the training data are more complicated than transitional and positional changes. Hence, it is not trivial answering to when and where geometric transformations are suitable to be applied.

Imbalanced data

Commonly, biological data tend to be imbalanced, as negative samples are much more numerous than positive ones [162–164]. For example, compared to COVID-19-positive X-ray images, the volume of normal X-ray images is very large. It should be noted that undesirable results may be produced when training a DL model using imbalanced data. The following techniques are used to solve this issue. First, it is necessary to employ the correct criteria for evaluating the loss, as well as the prediction result. In considering the imbalanced data, the model should perform well on small classes as well as larger ones. Thus, the model should employ area under curve (AUC) as the resultant loss as well as the criteria [165]. Second, it should employ the weighted cross-entropy loss, which ensures the model will perform well with small classes if it still prefers to employ the cross-entropy loss. Simultaneously, during model training, it is possible either to down-sample the large classes or up-sample the small classes. Finally, to make the data balanced as in Ref. [166], it is possible to construct models for every hierarchical level, as a biological system frequently has hierarchical label space. However, the effect of the imbalanced data on the performance of the DL model has been comprehensively investigated. In addition, to lessen the problem, the most frequently used techniques were also compared. Nevertheless, note that these techniques are not specified for biological problems.

Interpretability of data

Occasionally, DL techniques are analyzed to act as a black box. In fact, they are interpretable. The need for a method of interpreting DL, which is used to obtain the valuable motifs and patterns recognized by the network, is common in many fields, such as bioinformatics [167]. In the task of disease diagnosis, it is not only required to know the disease diagnosis or prediction results of a trained DL model, but also how to enhance the surety of the prediction outcomes, as the model makes its decisions based on these verifications [168]. To achieve this, it is possible to give a score of importance for every

portion of the particular example. Within this solution, back-propagation-based techniques or perturbation-based approaches are used [169]. In the perturbation-based approaches, a portion of the input is changed and the effect of this change on the model output is observed [170–173]. This concept has high computational complexity, but it is simple to understand. On the other hand, to check the score of the importance of various input portions, the signal from the output propagates back to the input layer in the back-propagation-based techniques. These techniques have been proven valuable in [174]. In different scenarios, various meanings can represent the model interpretability.

Uncertainty scaling

Commonly, the final prediction label is not the only label required when employing DL techniques to achieve the prediction; the score of confidence for every inquiry from the model is also desired. The score of confidence is defined as how confident the model is in its prediction [175]. Since the score of confidence prevents belief in unreliable and misleading predictions, it is a significant attribute, regardless of the application scenario. In biology, the confidence score reduces the resources and time expended in proving the outcomes of the misleading prediction. Generally speaking, in healthcare or similar applications, the uncertainty scaling is frequently very significant; it helps in evaluating automated clinical decisions and the reliability of machine learning-based disease-diagnosis [176, 177]. Because overconfident prediction can be the output of different DL models, the score of probability (achieved from the softmax output of the direct-DL) is often not in the correct scale [178]. Note that the softmax output requires post-scaling to achieve a reliable probability score. For outputting the probability score in the correct scale, several techniques have been introduced, including Bayesian Binning into Quantiles (BBQ) [179], isotonic regression [180], histogram binning [181], and the legendary Platt scaling [182]. More specifically, for DL techniques, temperature scaling was recently introduced, which achieves superior performance compared to the other techniques.

Catastrophic forgetting

This is defined as incorporating new information into a plain DL model, made possible by interfering with the learned information. For instance, consider a case where there are 1000 types of flowers and a model is trained to classify these flowers, after which a new type of flower is introduced; if the model is fine-tuned only with this new class, its performance will become unsuccessful with the older classes [183, 184]. The logical data are continually collected and renewed, which is in fact a highly typical scenario in many fields, e.g. Biology. To address this issue, there is a direct solution that involves employing old and new data to train an entirely new model from scratch. This solution is time-consuming and computationally intensive; furthermore, it leads to an unstable state for the learned representation of the initial data. At this time, three different types of ML techniques, which have not catastrophic forgetting, are made available to solve the human brain problem founded on the neurophysiological theories [185, 186]. Techniques of the first type are founded on regularizations such as EWC [183] Techniques of the second type employ rehearsal training techniques and dynamic neural network

architecture like iCaRL [187, 188]. Finally, techniques of the third type are founded on dual-memory learning systems [189]. Refer to [190–192] in order to gain more details.

Model compression

To obtain well-trained models that can still be employed productively, DL models have intensive memory and computational requirements due to their huge complexity and large numbers of parameters [193, 194]. One of the fields that is characterized as data-intensive is the field of healthcare and environmental science. These needs reduce the deployment of DL in limited computational-power machines, mainly in the healthcare field. The numerous methods of assessing human health and the data heterogeneity have become far more complicated and vastly larger in size [195]; thus, the issue requires additional computation [196]. Furthermore, novel hardware-based parallel processing solutions such as FPGAs and GPUs [197–199] have been developed to solve the computation issues associated with DL. Recently, numerous techniques for compressing the DL models, designed to decrease the computational issues of the models from the starting point, have also been introduced. These techniques can be classified into four classes. In the first class, the redundant parameters (which have no significant impact on model performance) are reduced. This class, which includes the famous deep compression method, is called parameter pruning [200]. In the second class, the larger model uses its distilled knowledge to train a more compact model; thus, it is called knowledge distillation [201, 202]. In the third class, compact convolution filters are used to reduce the number of parameters [203]. In the final class, the information parameters are estimated for preservation using low-rank factorization [204]. For model compression, these classes represent the most representative techniques. In [193], it has been provided a more comprehensive discussion about the topic.

Overfitting

DL models have excessively high possibilities of resulting in data overfitting at the training stage due to the vast number of parameters involved, which are correlated in a complex manner. Such situations reduce the model's ability to achieve good performance on the tested data [90, 205]. This problem is not only limited to a specific field, but involves different tasks. Therefore, when proposing DL techniques, this problem should be fully considered and accurately handled. In DL, the implied bias of the training process enables the model to overcome crucial overfitting problems, as recent studies suggest [205–208]. Even so, it is still necessary to develop techniques that handle the overfitting problem. An investigation of the available DL algorithms that ease the overfitting problem can categorize them into three classes. The first class acts on both the model architecture and model parameters and includes the most familiar approaches, such as weight decay [209], batch normalization [210], and dropout [90]. In DL, the default technique is weight decay [209], which is used extensively in almost all ML algorithms as a universal regularizer. The second class works on model inputs such as data corruption and data augmentation [150, 211]. One reason for the overfitting problem is the lack of training data, which makes the learned distribution not mirror the real distribution. Data augmentation enlarges the training data. By contrast, marginalized data corruption improves the solution exclusive to augmenting the data. The final class works on the

model output. A recently proposed technique penalizes the over-confident outputs for regularizing the model [178]. This technique has demonstrated the ability to regularize RNNs and CNNs.

Vanishing gradient problem

In general, when using backpropagation and gradient-based learning techniques along with ANNs, largely in the training stage, a problem called the vanishing gradient problem arises [212–214]. More specifically, in each training iteration, every weight of the neural network is updated based on the current weight and is proportionally relative to the partial derivative of the error function. However, this weight updating may not occur in some cases due to a vanishingly small gradient, which in the worst case means that no extra training is possible and the neural network will stop completely. Conversely, similarly to other activation functions, the sigmoid function shrinks a large input space to a tiny input space. Thus, the derivative of the sigmoid function will be small due to large variation at the input that produces a small variation at the output. In a shallow network, only some layers use these activations, which is not a significant issue. While using more layers will lead the gradient to become very small in the training stage, in this case, the network works efficiently. The back-propagation technique is used to determine the gradients of the neural networks. Initially, this technique determines the network derivatives of each layer in the reverse direction, starting from the last layer and progressing back to the first layer. The next step involves multiplying the derivatives of each layer down the network in a similar manner to the first step. For instance, multiplying N small derivatives together when there are N hidden layers employs an activation function such as the sigmoid function. Hence, the gradient declines exponentially while propagating back to the first layer. More specifically, the biases and weights of the first layers cannot be updated efficiently during the training stage because the gradient is small. Moreover, this condition decreases the overall network accuracy, as these first layers are frequently critical to recognizing the essential elements of the input data. However, such a problem can be avoided through employing activation functions. These functions lack the squishing property, i.e., the ability to squish the input space to within a small space. By mapping X to max, the ReLU [91] is the most popular selection, as it does not yield a small derivative that is employed in the field. Another solution involves employing the batch normalization layer [81]. As mentioned earlier, the problem occurs once a large input space is squashed into a small space, leading to vanishing the derivative. Employing batch normalization degrades this issue by simply normalizing the input, i.e., the expression $|x|$ does not accomplish the exterior boundaries of the sigmoid function. The normalization process makes the largest part of it come down in the green area, which ensures that the derivative is large enough for further actions. Furthermore, faster hardware can tackle the previous issue, e.g. that provided by GPUs. This makes standard back-propagation possible for many deeper layers of the network compared to the time required to recognize the vanishing gradient problem [215].

Exploding gradient problem

Opposite to the vanishing problem is the one related to gradient. Specifically, large error gradients are accumulated during back-propagation [216–218]. The latter will lead to extremely significant updates to the weights of the network, meaning that the system becomes unsteady. Thus, the model will lose its ability to learn effectively. Grosso modo, moving backward in the network during back-propagation, the gradient grows exponentially by repetitively multiplying gradients. The weight values could thus become incredibly large and may overflow to become a not-a-number (NaN) value. Some potential solutions include:

1. Using different weight regularization techniques.
2. Redesigning the architecture of the network model.

Underspecification

In 2020, a team of computer scientists at Google has identified a new challenge called underspecification [219]. ML models including DL models often show surprisingly poor behavior when they are tested in real-world applications such as computer vision, medical imaging, natural language processing, and medical genomics. The reason behind the weak performance is due to underspecification. It has been shown that small modifications can force a model towards a completely different solution as well as lead to different predictions in deployment domains. There are different techniques of addressing underspecification issue. One of them is to design "stress tests" to examine how good a model works on real-world data and to find out the possible issues. Nevertheless, this demands a reliable understanding of the process the model can work inaccurately. The team stated that "Designing stress tests that are well-matched to applied requirements, and that provide good "coverage" of potential failure modes is a major challenge". Underspecification puts major constraints on the credibility of ML predictions and may require some reconsidering over certain applications. Since ML is linked to human by serving several applications such as medical imaging and self-driving cars, it will require proper attention to this issue.

Applications of deep learning

Presently, various DL applications are widespread around the world. These applications include healthcare, social network analysis, audio and speech processing (like recognition and enhancement), visual data processing methods (such as multimedia data analysis and computer vision), and NLP (translation and sentence classification), among others (Fig. 29) [220–224]. These applications have been classified into five categories: classification, localization, detection, segmentation, and registration. Although each of these tasks has its own target, there is fundamental overlap in the pipeline implementation of these applications as shown in Fig. 30. Classification is a concept that categorizes a set of data into classes. Detection is used to locate interesting objects in an image with consideration given to the background. In detection, multiple objects, which could be from dissimilar classes, are surrounded by bounding boxes. Localization is the concept

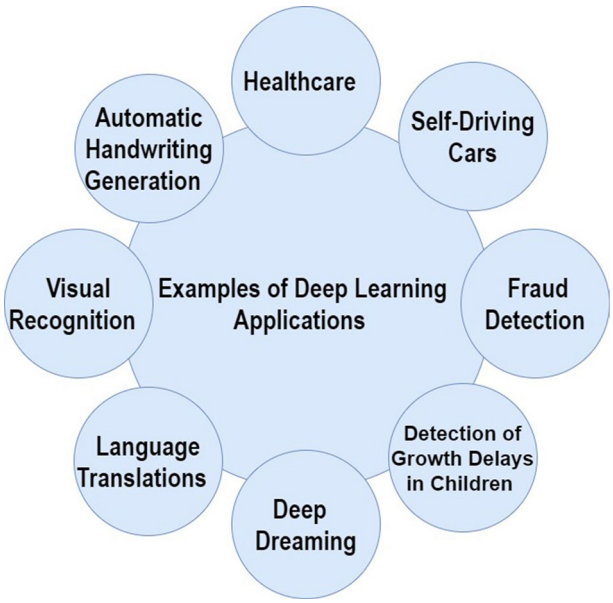

Fig. 29 Examples of DL applications

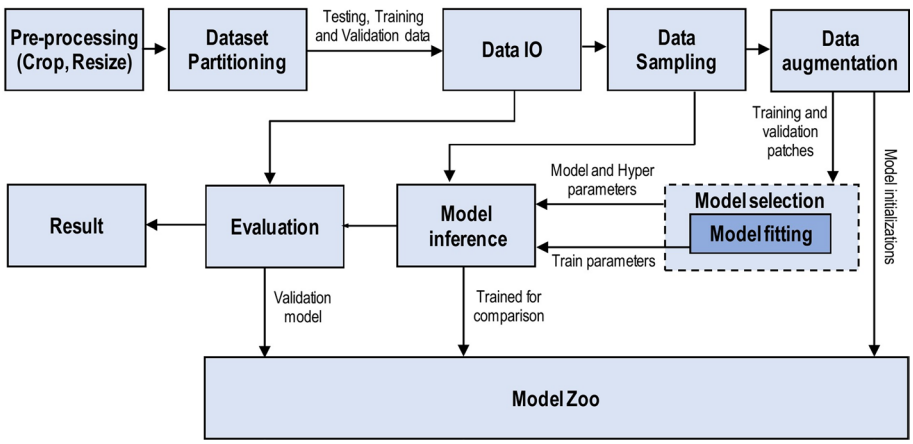

Fig. 30 Workflow of deep learning tasks

used to locate the object, which is surrounded by a single bounding box. In segmentation (semantic segmentation), the target object edges are surrounded by outlines, which also label them; moreover, fitting a single image (which could be 2D or 3D) onto another refers to registration. One of the most important and wide-ranging DL applications are in healthcare [225–230]. This area of research is critical due to its relation to human lives. Moreover, DL has shown tremendous performance in healthcare. Therefore, we take DL applications in the medical image analysis field as an example to describe the DL applications.

Classification

Computer-Aided Diagnosis (CADx) is another title sometimes used for classification. Bharati et al. [231] used a chest X-ray dataset for detecting lung diseases based on a CNN. Another study attempted to read X-ray images by employing CNN [232]. In this modality, the comparative accessibility of these images has likely enhanced the progress of DL. [233] used an improved pre-trained GoogLeNet CNN containing more than 150,000 images for training and testing processes. This dataset was augmented from 1850 chest X-rays. The creators reorganized the image orientation into lateral and frontal views and achieved approximately 100% accuracy. This work of orientation classification has clinically limited use. As a part of an ultimately fully automated diagnosis workflow, it obtained the data augmentation and pre-trained efficiency in learning the metadata of relevant images. Chest infection, commonly referred to as pneumonia, is extremely treatable, as it is a commonly occurring health problem worldwide. Conversely, Rajpurkar et al. [234] utilized CheXNet, which is an improved version of DenseNet [112] with 121 convolution layers, for classifying fourteen types of disease. These authors used the CheXNet14 dataset [235], which comprises 112,000 images. This network achieved an excellent performance in recognizing fourteen different diseases. In particular, pneumonia classification accomplished a 0.7632 AUC score using receiver operating characteristics (ROC) analysis. In addition, the network obtained better than or equal to the performance of both a three-radiologist panel and four individual radiologists. Zuo et al. [236] have adopted CNN for candidate classification in lung nodule. Shen et al. [237] employed both Random Forest (RF) and SVM classifiers with CNNs to classify lung nodules. They employed two convolutional layers with each of the three parallel CNNs. The LIDC-IDRI (Lung Image Database Consortium) dataset, which contained 1010-labeled CT lung scans, was used to classify the two types of lung nodules (malignant and benign). Different scales of the image patches were used by every CNN to extract features, while the output feature vector was constructed using the learned features. Next, these vectors were classified into malignant or benign using either the RF classifier or SVM with radial basis function (RBF) filter. The model was robust to various noisy input levels and achieved an accuracy of 86% in nodule classification. Conversely, the model of [238] interpolates the image data missing between PET and MRI images using 3D CNNs. The Alzheimer Disease Neuroimaging Initiative (ADNI) database, containing 830 PET and MRI patient scans, was utilized in their work. The PET and MRI images are used to train the 3D CNNs, first as input and then as output. Furthermore, for patients who have no PET images, the 3D CNNs utilized the trained images to rebuild the PET images. These rebuilt images approximately fitted the actual disease recognition outcomes. However, this approach did not address the overfitting issues, which in turn restricted their technique in terms of its possible capacity for generalization. Diagnosing normal versus Alzheimer's disease patients has been achieved by several CNN models [239, 240]. Hosseini-Asl et al. [241] attained 99% accuracy for up-to-date outcomes in diagnosing normal versus Alzheimer's disease patients. These authors applied an auto-encoder architecture using 3D CNNs. The generic brain features were pre-trained on the CADDementia dataset. Subsequently, the outcomes of these learned features became inputs to higher layers to differentiate between patient scans of Alzheimer's disease, mild cognitive impairment, or normal brains based on the

ADNI dataset and using fine-tuned deep supervision techniques. The architectures of VGGNet and RNNs, in that order, were the basis of both VOXCNN and ResNet models developed by Korolev et al. [242]. They also discriminated between Alzheimer's disease and normal patients using the ADNI database. Accuracy was 79% for Voxnet and 80% for ResNet. Compared to Hosseini-Asl's work, both models achieved lower accuracies. Conversely, the implementation of the algorithms was simpler and did not require feature hand-crafting, as Korolev declared. In 2020, Mehmood et al. [240] trained a developed CNN-based network called "SCNN" with MRI images for the tasks of classification of Alzheimer's disease. They achieved state-of-the-art results by obtaining an accuracy of 99.05%.

Recently, CNN has taken some medical imaging classification tasks to different level from traditional diagnosis to automated diagnosis with tremendous performance. Examples of these tasks are diabetic foot ulcer (DFU) (as normal and abnormal (DFU) classes) [87, 243–246], sickle cells anemia (SCA) (as normal, abnormal (SCA), and other blood components) [86, 247], breast cancer by classify hematoxylin–eosin-stained breast biopsy images into four classes: invasive carcinoma, in-situ carcinoma, benign tumor and normal tissue [42, 88, 248–252], and multi-class skin cancer classification [253–255].

In 2020, CNNs are playing a vital role in early diagnosis of the novel coronavirus (COVID-2019). CNN has become the primary tool for automatic COVID-19 diagnosis in many hospitals around the world using chest X-ray images [256–260]. More details about the classification of medical imaging applications can be found in [226, 261–265].

Localization

Although applications in anatomy education could increase, the practicing clinician is more likely to be interested in the localization of normal anatomy. Radiological images are independently examined and described outside of human intervention, while localization could be applied in completely automatic end-to-end applications [266–268]. Zhao et al. [269] introduced a new deep learning-based approach to localize pancreatic tumor in projection X-ray images for image-guided radiation therapy without the need for fiducials. Roth et al. [270] constructed and trained a CNN using five convolutional layers to classify around 4000 transverse-axial CT images. These authors used five categories for classification: legs, pelvis, liver, lung, and neck. After data augmentation techniques were applied, they achieved an AUC score of 0.998 and the classification error rate of the model was 5.9%. For detecting the positions of the spleen, kidney, heart, and liver, Shin et al. [271] employed stacked auto-encoders on 78 contrast-improved MRI scans of the stomach area containing the kidneys or liver. Temporal and spatial domains were used to learn the hierarchal features. Based on the organs, these approaches achieved detection accuracies of 62–79%. Sirazitdinov et al. [268] presented an aggregate of two convolutional neural networks, namely RetinaNet and Mask R-CNN for pneumonia detection and localization.

Detection

Computer-Aided Detection (CADe) is another method used for detection. For both the clinician and the patient, overlooking a lesion on a scan may have dire consequences.

Thus, detection is a field of study requiring both accuracy and sensitivity [272–274]. Chouhan et al. [275] introduced an innovative deep learning framework for the detection of pneumonia by adopting the idea of transfer learning. Their approach obtained an accuracy of 96.4% with a recall of 99.62% on unseen data. In the area of COVID-19 and pulmonary disease, several convolutional neural network approaches have been proposed for automatic detection from X-ray images which showed an excellent performance [46, 276–279].

In the area of skin cancer, there several applications were introduced for the detection task [280–282]. Thurnhofer-Hemsi et al. [283] introduced a deep learning approach for skin cancer detection by fine-tuning five state-of-art convolutional neural network models. They addressed the issue of a lack of training data by adopting the ideas of transfer learning and data augmentation techniques. DenseNet201 network has shown superior results compared to other models.

Another interesting area is that of histopathological images, which are progressively digitized. Several papers have been published in this field [284–290]. Human pathologists read these images laboriously; they search for malignancy markers, such as a high index of cell proliferation, using molecular markers (e.g. Ki-67), cellular necrosis signs, abnormal cellular architecture, enlarged numbers of mitotic figures denoting augmented cell replication, and enlarged nucleus-to-cytoplasm ratios. Note that the histopathological slide may contain a huge number of cells (up to the thousands). Thus, the risk of disregarding abnormal neoplastic regions is high when wading through these cells at excessive levels of magnification. Ciresan et al. [291] employed CNNs of 11–13 layers for identifying mitotic figures. Fifty breast histology images from the MITOS dataset were used. Their technique attained recall and precision scores of 0.7 and 0.88 respectively. Sirinukunwattana et al. [292] utilized 100 histology images of colorectal adenocarcinoma to detect cell nuclei using CNNs. Roughly 30,000 nuclei were hand-labeled for training purposes. The novelty of this approach was in the use of Spatially Constrained CNN. This CNN detects the center of nuclei using the surrounding spatial context and spatial regression. Instead of this CNN, Xu et al. [293] employed a stacked sparse autoencoder (SSAE) to identify nuclei in histological slides of breast cancer, achieving 0.83 and 0.89 recall and precision scores respectively. In this field, they showed that unsupervised learning techniques are also effectively utilized. In medical images, Albarquoni et al. [294] investigated the problem of insufficient labeling. They crowd-sourced the actual mitoses labeling in the histology images of breast cancer (from amateurs online). Solving the recurrent issue of inadequate labeling during the analysis of medical images can be achieved by feeding the crowd-sourced input labels into the CNN. This method signifies a remarkable proof-of-concept effort. In 2020, Lei et al. [285] introduced the employment of deep convolutional neural networks for automatic identification of mitotic candidates from histological sections for mitosis screening. They obtained the state-of-the-art detection results on the dataset of the International Pattern Recognition Conference (ICPR) 2012 Mitosis Detection Competition.

Segmentation

Although MRI and CT image segmentation research includes different organs such as knee cartilage, prostate, and liver, most research work has concentrated on brain

segmentation, particularly tumors [295–300]. This issue is highly significant in surgical preparation to obtain the precise tumor limits for the shortest surgical resection. During surgery, excessive sacrificing of key brain regions may lead to neurological shortfalls including cognitive damage, emotionlessness, and limb difficulty. Conventionally, medical anatomical segmentation was done by hand; more specifically, the clinician draws out lines within the complete stack of the CT or MRI volume slice by slice. Thus, it is perfect for implementing a solution that computerizes this painstaking work. Wadhwa et al. [301] presented a brief overview on brain tumor segmentation of MRI images. Akkus et al. [302] wrote a brilliant review of brain MRI segmentation that addressed the different metrics and CNN architectures employed. Moreover, they explain several competitions in detail, as well as their datasets, which included Ischemic Stroke Lesion Segmentation (ISLES), Mild Traumatic brain injury Outcome Prediction (MTOP), and Brain Tumor Segmentation (BRATS).

Chen et al. [299] proposed convolutional neural networks for precise brain tumor segmentation. The approach that they employed involves several approaches for better features learning including the DeepMedic model, a novel dual-force training scheme, a label distribution-based loss function, and Multi-Layer Perceptron-based post-processing. They conducted their method on the two most modern brain tumor segmentation datasets, i.e., BRATS 2017 and BRATS 2015 datasets. Hu et al. [300] introduced the brain tumor segmentation method by adopting a multi-cascaded convolutional neural network (MCCNN) and fully connected conditional random fields (CRFs). The achieved results were excellent compared with the state-of-the-art methods.

Moeskops et al. [303] employed three parallel-running CNNs, each of which had a 2D input patch of dissimilar size, for segmenting and classifying MRI brain images. These images, which include 35 adults and 22 pre-term infants, were classified into various tissue categories such as cerebrospinal fluid, grey matter, and white matter. Every patch concentrates on capturing various image aspects with the benefit of employing three dissimilar sizes of input patch; here, the bigger sizes incorporated the spatial features, while the lowest patch sizes concentrated on the local textures. In general, the algorithm has Dice coefficients in the range of 0.82–0.87 and achieved a satisfactory accuracy. Although 2D image slices are employed in the majority of segmentation research, Milletrate et al. [304] implemented 3D CNN for segmenting MRI prostate images. Furthermore, they used the PROMISE2012 challenge dataset, from which fifty MRI scans were used for training and thirty for testing. The U-Net architecture of Ronnerberger et al. [305] inspired their V-net. This model attained a 0.869 Dice coefficient score, the same as the winning teams in the competition. To reduce overfitting and create the model of a deeper 11-convolutional layer CNN, Pereira et al. [306] applied intentionally small-sized filters of 3x3. Their model used MRI scans of 274 gliomas (a type of brain tumor) for training. They achieved first place in the 2013 BRATS challenge, as well as second place in the BRATS challenge 2015. Havaei et al. [307] also considered gliomas using the 2013 BRATS dataset. They investigated different 2D CNN architectures. Compared to the winner of BRATS 2013, their algorithm worked better, as it required only 3 min to execute rather than 100 min. The concept of cascaded architecture formed the basis of their model. Thus, it is referred to as an InputCascadeCNN. Employing FC Conditional Random Fields (CRFs), atrous spatial pyramid pooling, and up-sampled filters

were techniques introduced by Chen et al. [308]. These authors aimed to enhance the accuracy of localization and enlarge the field of view of every filter at a multi-scale. Their model, DeepLab, attained 79.7% mIOU (mean Intersection Over Union). In the PASCAL VOC-2012 image segmentation, their model obtained an excellent performance.

Recently, the Automatic segmentation of COVID-19 Lung Infection from CT Images helps to detect the development of COVID-19 infection by employing several deep learning techniques [309–312].

Registration

Usually, given two input images, the four main stages of the canonical procedure of the image registration task are [313, 314]:

- Target Selection: it illustrates the determined input image that the second counterpart input image needs to remain accurately superimposed to.
- Feature Extraction: it computes the set of features extracted from each input image.
- Feature Matching: it allows finding similarities between the previously obtained features.
- Pose Optimization: it is aimed to minimize the distance between both input images.

Then, the result of the registration procedure is the suitable geometric transformation (e.g. translation, rotation, scaling, etc.) that provides both input images within the same coordinate system in a way the distance between them is minimal, i.e. their level of superimposition/overlapping is optimal. It is out of the scope of this work to provide an extensive review of this topic. Nevertheless, a short summary is accordingly introduced next.

Commonly, the input images for the DL-based registration approach could be in various forms, e.g. point clouds, voxel grids, and meshes. Additionally, some techniques allow as inputs the result of the Feature Extraction or Matching steps in the canonical scheme. Specifically, the outcome could be some data in a particular form as well as the result of the steps from the classical pipeline (feature vector, matching vector, and transformation). Nevertheless, with the newest DL-based methods, a novel conceptual type of ecosystem issues. It contains acquired characteristics about the target, materials, and their behavior that can be registered with the input data. Such a conceptual ecosystem is formed by a neural network and its training manner, and it could be counted as an input to the registration approach. Nevertheless, it is not an input that one might adopt in every registration situation since it corresponds to an interior data representation.

From a DL view-point, the interpretation of the conceptual design enables differentiating the input data of a registration approach into defined or non-defined models. In particular, the illustrated phases are models that depict particular spatial data (e.g. 2D or 3D) while a non-defined one is a generalization of a data set created by a learning system. Yumer et al. [315] developed a framework in which the model acquires characteristics of objects, meaning ready to identify what a more sporty car seems like or a more comfy chair is, also adjusting a 3D model to fit those characteristics while maintaining the main characteristics of the primary data. Likewise, a fundamental perspective of the unsupervised learning method introduced by Ding et al. [316] is that there is no target for the

registration approach. In this instance, the network is able of placing each input point cloud in a global space, solving SLAM issues in which many point clouds have to be registered rigidly. On the other hand, Mahadevan [317] proposed the combination of two conceptual models utilizing the growth of Imagination Machines to give flexible artificial intelligence systems and relationships between the learned phases through training schemes that are not inspired on labels and classifications. Another practical application of DL, especially CNNs, to image registration is the 3D reconstruction of objects. Wang et al. [318] applied an adversarial way using CNNs to rebuild a 3D model of an object from its 2D image. The network learns many objects and orally accomplishes the registration between the image and the conceptual model. Similarly, Hermoza et al. [319] also utilize the GAN network for prognosticating the absent geometry of damaged archaeological objects, providing the reconstructed object based on a voxel grid format and a label selecting its class.

DL for medical image registration has numerous applications, which were listed by some review papers [320–322]. Yang et al. [323] implemented stacked convolutional layers as an encoder-decoder approach to predict the morphing of the input pixel into its last formation using MRI brain scans from the OASIS dataset. They employed a registration model known as Large Deformation Diffeomorphic Metric Mapping (LDDMM) and attained remarkable enhancements in computation time. Miao et al. [324] used synthetic X-ray images to train a five-layer CNN to register 3D models of a trans-esophageal probe, a hand implant, and a knee implant onto 2D X-ray images for pose estimation. They determined that their model achieved an execution time of 0.1 s, representing an important enhancement against the conventional registration techniques based on intensity; moreover, it achieved effective registrations 79–99% of the time. Li et al. [325] introduced a neural network-based approach for the non-rigid 2D–3D registration of the lateral cephalogram and the volumetric cone-beam CT (CBCT) images.

Computational approaches

For computationally exhaustive applications, complex ML and DL approaches have rapidly been identified as the most significant techniques and are widely used in different fields. The development and enhancement of algorithms aggregated with capabilities of well-behaved computational performance and large datasets make it possible to effectively execute several applications, as earlier applications were either not possible or difficult to take into consideration.

Currently, several standard DNN configurations are available. The interconnection patterns between layers and the total number of layers represent the main differences between these configurations. The Table 2 illustrates the growth rate of the overall number of layers over time, which seems to be far faster than the "Moore's Law growth rate". In normal DNN, the number of layers grew by around 2.3× each year in the period from 2012 to 2016. Recent investigations of future ResNet versions reveal that the number of layers can be extended up to 1000. However, an SGD technique is employed to fit the weights (or parameters), while different optimization techniques are employed to obtain parameter updating during the DNN training process. Repetitive updates are required to enhance network accuracy in addition to a minorly augmented rate of enhancement. For example, the training process using ImageNet as a large dataset, which contains more

than 14 million images, along with ResNet as a network model, take around 30K to 40K repetitions to converge to a steady solution. In addition, the overall computational load, as an upper-level prediction, may exceed 1020 FLOPS when both the training set size and the DNN complexity increase.

Prior to 2008, boosting the training to a satisfactory extent was achieved by using GPUs. Usually, days or weeks are needed for a training session, even with GPU support. By contrast, several optimization strategies were developed to reduce the extensive learning time. The computational requirements are believed to increase as the DNNs continuously enlarge in both complexity and size.

In addition to the computational load cost, the memory bandwidth and capacity have a significant effect on the entire training performance, and to a lesser extent, deduction. More specifically, the parameters are distributed through every layer of the input data, there is a sizeable amount of reused data, and the computation of several network layers exhibits an excessive computation-to-bandwidth ratio. By contrast, there are no distributed parameters, the amount of reused data is extremely small, and the additional FC layers have an extremely small computation-to-bandwidth ratio. Table 3 presents a comparison between different aspects related to the devices. In addition, the table is established to facilitate familiarity with the tradeoffs by obtaining the optimal approach for configuring a system based on either FPGA, GPU, or CPU devices. It should be noted that each has corresponding weaknesses and strengths; accordingly, there are no clear one-size-fits-all solutions.

Although GPU processing has enhanced the ability to address the computational challenges related to such networks, the maximum GPU (or CPU) performance is not achieved, and several techniques or models have turned out to be strongly linked to bandwidth. In the worst cases, the GPU efficiency is between 15 and 20% of the maximum theoretical performance. This issue is required to enlarge the memory bandwidth using high-bandwidth stacked memory. Next, different approaches based on FPGA, GPU, and CPU are accordingly detailed.

Table 3 A comparison between different aspects related to the devices

Feature	Assessment	Leader
Development	CPU is the easiest to program, then GPU, then FPGA	CPU
Size	Both FPGA and CPU have smaller volume solutions due to their lower power consumption	FPGA-CPU
Customization	Broader flexibility is provided by FPGA	FPGA
Ease of change	Easier way to vary application functionality is provided by GPU and CPU	GPU-CPU
Backward compatibility	Transferring RTL to novel FPGA requires additional work. Furthermore, GPU has less stable architecture than CPU	CPU
Interfaces	Several varieties of interfaces can be implemented using FPGA	FPGA
Processing/$	FPGA configurability assists utilization in wider acceleration space. Due to the considerable processing abilities, GPU wins	FPGA-GPU
Processing/watt	Customized designs can be optimized	FPGA
Timing latency	Implemented FPGA algorithm offers deterministic timing, which is in turn much faster than GPU	FPGA
Large data analysis	FPGA performs well for inline processing, while CPU supports storage capabilities and largest memory	FPGA-CPU
DCNN inference	FPGA has lower latency and can be customized	FPGA
DCNN training	Greater float-point capabilities provided by GPU	GPU

CPU-based approach

The well-behaved performance of the CPU nodes usually assists robust network connectivity, storage abilities, and large memory. Although CPU nodes are more common-purpose than those of FPGA or GPU, they lack the ability to match them in unprocessed computation facilities, since this requires increased network ability and a larger memory capacity.

GPU-based approach

GPUs are extremely effective for several basic DL primitives, which include greatly parallel-computing operations such as activation functions, matrix multiplication, and convolutions [326–330]. Incorporating HBM-stacked memory into the up-to-date GPU models significantly enhances the bandwidth. This enhancement allows numerous primitives to efficiently utilize all computational resources of the available GPUs. The improvement in GPU performance over CPU performance is usually 10-20:1 related to dense linear algebra operations.

Maximizing parallel processing is the base of the initial GPU programming model. For example, a GPU model may involve up to sixty-four computational units. There are four SIMD engines per each computational layer, and each SIMD has sixteen floating-point computation lanes. The peak performance is 25 TFLOPS (fp16) and 10 TFLOPS (fp32) as the percentage of the employment approaches 100%. Additional GPU performance may be achieved if the addition and multiply functions for vectors combine the inner production instructions for matching primitives related to matrix operations.

For DNN training, the GPU is usually considered to be an optimized design, while for inference operations, it may also offer considerable performance improvements.

FPGA-based approach

FPGA is wildly utilized in various tasks including deep learning [199, 247, 331–334]. Inference accelerators are commonly implemented utilizing FPGA. The FPGA can be effectively configured to reduce the unnecessary or overhead functions involved in GPU systems. Compared to GPU, the FPGA is restricted to both weak-behaved floating-point performance and integer inference. The main FPGA aspect is the capability to dynamically reconfigure the array characteristics (at run-time), as well as the capability to configure the array by means of effective design with little or no overhead.

As mentioned earlier, the FPGA offers both performance and latency for every watt it gains over GPU and CPU in DL inference operations. Implementation of custom high-performance hardware, pruned networks, and reduced arithmetic precision are three factors that enable the FPGA to implement DL algorithms and to achieve FPGA with this level of efficiency. In addition, FPGA may be employed to implement CNN overlay engines with over 80% efficiency, eight-bit accuracy, and over 15 TOPs peak performance; this is used for a few conventional CNNs, as Xillinx and partners demonstrated recently. By contrast, pruning techniques are mostly employed in the LSTM context. The sizes of the models can be efficiently minimized by up to 20×, which provides an important benefit during the implementation of the optimal solution, as MLP neural processing demonstrated. A recent study in the field of implementing fixed-point precision and

custom floating-point has revealed that lowering the 8-bit is extremely promising; moreover, it aids in supplying additional advancements to implementing peak performance FPGA related to the DNN models.

Evaluation metrics

Evaluation metrics adopted within DL tasks play a crucial role in achieving the optimized classifier [335]. They are utilized within a usual data classification procedure through two main stages: training and testing. It is utilized to optimize the classification algorithm during the training stage. This means that the evaluation metric is utilized to discriminate and select the optimized solution, e.g., as a discriminator, which can generate an extra-accurate forecast of upcoming evaluations related to a specific classifier. For the time being, the evaluation metric is utilized to measure the efficiency of the created classifier, e.g. as an evaluator, within the model testing stage using hidden data. As given in Eq. 20, TN and TP are defined as the number of negative and positive instances, respectively, which are successfully classified. In addition, FN and FP are defined as the number of misclassified positive and negative instances respectively. Next, some of the most well-known evaluation metrics are listed below.

1. Accuracy: Calculates the ratio of correct predicted classes to the total number of samples evaluated (Eq. 20).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{20}$$

2. Sensitivity or Recall: Utilized to calculate the fraction of positive patterns that are correctly classified (Eq. 21).

$$Sensitivity = \frac{TP}{TP + FN} \tag{21}$$

3. Specificity: Utilized to calculate the fraction of negative patterns that are correctly classified (Eq. 22).

$$Specificity = \frac{TN}{FP + TN} \tag{22}$$

4. Precision: Utilized to calculate the positive patterns that are correctly predicted by all predicted patterns in a positive class (Eq. 23).

$$Precision = \frac{TP}{TP + FP} \tag{23}$$

5. F1-Score: Calculates the harmonic average between recall and precision rates (Eq. 24).

$$F1_{score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{24}$$

6. J Score: This metric is also called Youdens J statistic. Eq. 25 represents the metric.

$$J_{score} = Sensitivity + Specificity - 1 \tag{25}$$

7. False Positive Rate (FPR): This metric refers to the possibility of a false alarm ratio as calculated in Eq. 26

$$FPR = 1 - Specificity \tag{26}$$

8. Area Under the ROC Curve: AUC is a common ranking type metric. It is utilized to conduct comparisons between learning algorithms [336–338], as well as to construct an optimal learning model [339, 340]. In contrast to probability and threshold metrics, the AUC value exposes the entire classifier ranking performance. The following formula is used to calculate the AUC value for two-class problem [341] (Eq. 27)

$$AUC = \frac{S_p - n_p(n_n + 1)/2}{n_p n_n} \tag{27}$$

Here, S_p represents the sum of all positive ranked samples. The number of negative and positive samples is denoted as n_n and n_p, respectively. Compared to the accuracy metrics, the AUC value was verified empirically and theoretically, making it very helpful for identifying an optimized solution and evaluating the classifier performance through classification training.

When considering the discrimination and evaluation processes, the AUC performance was brilliant. However, for multiclass issues, the AUC computation is primarily cost-effective when discriminating a large number of created solutions. In addition, the time complexity for computing the AUC is $O\left(|C|^2\ n \log n\right)$ with respect to the Hand and Till AUC model [341] and $O\left(|C|\ n \log n\right)$ according to Provost and Domingo's AUC model [336].

Frameworks and datasets

Several DL frameworks and datasets have been developed in the last few years. various frameworks and libraries have also been used in order to expedite the work with good results. Through their use, the training process has become easier. Table 4 lists the most utilized frameworks and libraries.

Based on the star ratings on Github, as well as our own background in the field, TensorFlow is deemed the most effective and easy to use. It has the ability to work on several platforms. (Github is one of the biggest software hosting sites, while Github stars refer to how well-regarded a project is on the site). Moreover, there are several other benchmark datasets employed for different DL tasks. Some of these are listed in Table 5.

Summary and conclusion

Finally, it is mandatory the inclusion of a brief discussion by gathering all the relevant data provided along this extensive research. Next, an itemized analysis is presented in order to conclude our review and exhibit the future directions.

Table 4 List of the most common frameworks and libraries

Framework	License	Core language	Year of release	Homepages
TensorFlow	Apache 2.0	C++ & Python	2015	https://www.tensorflow.org/
Keras	MIT	Python	2015	https://keras.io/
Caffe	BSD	C++	2015	http://caffe.berkeleyvision.org/
MatConvNet	Oxford	MATLAB	2014	http://www.vlfeat.org/matconvnet/
MXNet	Apache 2.0	C++	2015	https://github.com/dmlc/mxnet
CNTK	MIT	C++	2016	https://github.com/Microsoft/CNTK
Theano	BSD	Python	2008	http://deeplearning.net/software/theano/
Torch	BSD	C & Lua	2002	http://torch.ch/
DL4j	Apache 2.0	Java	2014	https://deeplearning4j.org/
Gluon	AWS Microsoft	C++	2017	https://github.com/gluon-api/gluon-api/
OpenDeep	MIT	Python	2017	http://www.opendeep.org/

Table 5 Benchmark datasets

Dataset	Num. of classes	Applications	Link to dataset
ImageNet	1000	Image classification, object localization, object detection, etc.	http://www.image-net.org/
CIFAR10/100	10/100	Image classification	https://www.cs.toronto.edu/~kriz/cifar.html
MNIST	10	Classification of handwritten digits	http://yann.lecun.com/exdb/mnist/
Pascal VOC	20	Image classification, segmentation, object detection	http://host.robots.ox.ac.uk/pascal/VOC/voc2012/
Microsoft COCO	80	Object detection, semantic segmentation	https://cocodataset.org/#home
YFCC100M	8M	Video and image understanding	http://projects.dfki.unikl.de/yfcc100m/
YouTube-8M	4716	Video classification	https://research.google.com/youtube8m/
UCF-101	101	Human action detection	https://www.crcv.ucf.edu/data/UCF101.php
Kinetics	400	Human action detection	https://deepmind.com/research/open-source/kinetics
Google Open Images	350	Image classification, segmentation, object detection	https://storage.googleapis.com/openimages/web/index.html
CalTech101	101	Classification	http://www.vision.caltech.edu/Image_Datasets/Caltech101/
Labeled Faces in the Wild	–	Face recognition	http://vis-www.cs.umass.edu/lfw/
MIT-67 scene dataset	67	Indoor scene recognition	http://web.mit.edu/torralba/www/indoor.htm

- DL already experiences difficulties in simultaneously modeling multi-complex modalities of data. In recent DL developments, another common approach is that of multimodal DL.
- DL requires sizeable datasets (labeled data preferred) to predict unseen data and to train the models. This challenge turns out to be particularly difficult when real-time data processing is required or when the provided datasets are limited (such as in the

case of healthcare data). To alleviate this issue, TL and data augmentation have been researched over the last few years.

- Although ML slowly transitions to semi-supervised and unsupervised learning to manage practical data without the need for manual human labeling, many of the current deep-learning models utilize supervised learning.
- The CNN performance is greatly influenced by hyper-parameter selection. Any small change in the hyper-parameter values will affect the general CNN performance. Therefore, careful parameter selection is an extremely significant issue that should be considered during optimization scheme development.
- Impressive and robust hardware resources like GPUs are required for effective CNN training. Moreover, they are also required for exploring the efficiency of using CNN in smart and embedded systems.
- In the CNN context, ensemble learning [342, 343] represents a prospective research area. The collection of different and multiple architectures will support the model in improving its generalizability across different image categories through extracting several levels of semantic image representation. Similarly, ideas such as new activation functions, dropout, and batch normalization also merit further investigation.
- The exploitation of depth and different structural adaptations is significantly improved in the CNN learning capacity. Substituting the traditional layer configuration with blocks results in significant advances in CNN performance, as has been shown in the recent literature. Currently, developing novel and efficient block architectures is the main trend in new research models of CNN architectures. HRNet is only one example that shows there are always ways to improve the architecture.
- It is expected that cloud-based platforms will play an essential role in the future development of computational DL applications. Utilizing cloud computing offers a solution to handling the enormous amount of data. It also helps to increase efficiency and reduce costs. Furthermore, it offers the flexibility to train DL architectures.
- With the recent development in computational tools including a chip for neural networks and a mobile GPU, we will see more DL applications on mobile devices. It will be easier for users to use DL.
- Regarding the issue of lack of training data, It is expected that various techniques of transfer learning will be considered such as training the DL model on large unlabeled image datasets and next transferring the knowledge to train the DL model on a small number of labeled images for the same task.
- Last, this overview provides a starting point for the community of DL being interested in the field of DL. Furthermore, researchers would be allowed to decide the more suitable direction of work to be taken in order to provide more accurate alternatives to the field.

Acknowledgements

We would like to thank the professors from the Queensland University of Technology and the University of Information Technology and Communications who gave their feedback on the paper.

Authors' contributions

Conceptualization: LA, and JZ; methodology: LA, JZ, and JS; software: LA, and MAF; validation: LA, JZ, MA, and LF; formal analysis: LA, JZ, YD, and JS; investigation: LA, and JZ; resources: LA, JZ, and MAF; data curation: LA, and OA.; writing–original draft preparation: LA, and OA; writing—review and editing: LA, JZ, AJH, AA, YD, OA, JS, MAF, MA, and LF; visualization:

LA, and MAF; supervision: JZ, and YD; project administration: JZ, YD, and JS; funding acquisition: LA, AJH, AA, and YD. All authors read and approved the final manuscript.

Funding
This research received no external funding.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia. [2] Control and Systems Engineering Department, University of Technology, Baghdad 10001, Iraq. [3] Electrical Engineering Technical College, Middle Technical University, Baghdad 10001, Iraq. [4] Faculty of Electrical Engineering & Computer Science, University of Missouri, Columbia, MO 65211, USA. [5] AlNidhal Campus, University of Information Technology & Communications, Baghdad 10001, Iraq. [6] Department of Computer Science, University of Jaén, 23071 Jaén, Spain. [7] College of Computer Science and Information Technology, University of Sumer, Thi Qar 64005, Iraq. [8] School of Engineering, Manchester Metropolitan University, Manchester M1 5GD, UK.

Received: 21 January 2021 Accepted: 22 March 2021
Published online: 31 March 2021

References

1. Rozenwald MB, Galitsyna AA, Sapunov GV, Khrameeva EE, Gelfand MS. A machine learning framework for the prediction of chromatin folding in *Drosophila* using epigenetic features. PeerJ Comput Sci. 2020;6:307.
2. Amrit C, Paauw T, Aly R, Lavric M. Identifying child abuse through text mining and machine learning. Expert Syst Appl. 2017;88:402–18.
3. Hossain E, Khan I, Un-Noor F, Sikander SS, Sunny MSH. Application of big data and machine learning in smart grid, and associated security concerns: a review. IEEE Access. 2019;7:13960–88.
4. Crawford M, Khoshgoftaar TM, Prusa JD, Richter AN, Al Najada H. Survey of review spam detection using machine learning techniques. J Big Data. 2015;2(1):23.
5. Deldjoo Y, Elahi M, Cremonesi P, Garzotto F, Piazzolla P, Quadrana M. Content-based video recommendation system based on stylistic visual features. J Data Semant. 2016;5(2):99–113.
6. Al-Dulaimi K, Chandran V, Nguyen K, Banks J, Tomeo-Reyes I. Benchmarking hep-2 specimen cells classification using linear discriminant analysis on higher order spectra features of cell shape. Pattern Recogn Lett. 2019;125:534–41.
7. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE. A survey of deep neural network architectures and their applications. Neurocomputing. 2017;234:11–26.
8. Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes MP, Shyu ML, Chen SC, Iyengar S. A survey on deep learning: algorithms, techniques, and applications. ACM Comput Surv (CSUR). 2018;51(5):1–36.
9. Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, Hasan M, Van Essen BC, Awwal AA, Asari VK. A state-of-the-art survey on deep learning theory and architectures. Electronics. 2019;8(3):292.
10. Potok TE, Schuman C, Young S, Patton R, Spedalieri F, Liu J, Yao KT, Rose G, Chakma G. A study of complex deep learning networks on high-performance, neuromorphic, and quantum computers. ACM J Emerg Technol Comput Syst (JETC). 2018;14(2):1–21.
11. Adeel A, Gogate M, Hussain A. Contextual deep learning-based audio-visual switching for speech enhancement in real-world environments. Inf Fusion. 2020;59:163–70.
12. Tian H, Chen SC, Shyu ML. Evolutionary programming based deep learning feature selection and network construction for visual data classification. Inf Syst Front. 2020;22(5):1053–66.
13. Young T, Hazarika D, Poria S, Cambria E. Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag. 2018;13(3):55–75.
14. Koppe G, Meyer-Lindenberg A, Durstewitz D. Deep learning for small and big data in psychiatry. Neuropsychopharmacology. 2021;46(1):176–90.
15. Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol. 1. IEEE; 2005. p. 886–93.
16. Lowe DG. Object recognition from local scale-invariant features. In: Proceedings of the seventh IEEE international conference on computer vision, vol. 2. IEEE; 1999. p. 1150–7.
17. Wu L, Hoi SC, Yu N. Semantics-preserving bag-of-words models and applications. IEEE Trans Image Process. 2010;19(7):1908–20.

18. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–44.
19. Yao G, Lei T, Zhong J. A review of convolutional-neural-network-based action recognition. Pattern Recogn Lett. 2019;118:14–22.
20. Dhillon A, Verma GK. Convolutional neural network: a review of models, methodologies and applications to object detection. Prog Artif Intell. 2020;9(2):85–112.
21. Khan A, Sohail A, Zahoora U, Qureshi AS. A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev. 2020;53(8):5455–516.
22. Hasan RI, Yusuf SM, Alzubaidi L. Review of the state of the art of deep learning for plant diseases: a broad analysis and discussion. Plants. 2020;9(10):1302.
23. Xiao Y, Tian Z, Yu J, Zhang Y, Liu S, Du S, Lan X. A review of object detection based on deep learning. Multimed Tools Appl. 2020;79(33):23729–91.
24. Ker J, Wang L, Rao J, Lim T. Deep learning applications in medical image analysis. IEEE Access. 2017;6:9375–89.
25. Zhang Z, Cui P, Zhu W. Deep learning on graphs: a survey. IEEE Trans Knowl Data Eng. 2020. https://doi.org/10.1109/TKDE.2020.2981333.
26. Shrestha A, Mahmood A. Review of deep learning algorithms and architectures. IEEE Access. 2019;7:53040–65.
27. Najafabadi MM, Villanustre F, Khoshgoftaar TM, Seliya N, Wald R, Muharemagic E. Deep learning applications and challenges in big data analytics. J Big Data. 2015;2(1):1.
28. Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning, vol. 1. Cambridge: MIT press; 2016.
29. Shorten C, Khoshgoftaar TM, Furht B. Deep learning applications for COVID-19. J Big Data. 2021;8(1):1–54.
30. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Commun ACM. 2017;60(6):84–90.
31. Bhowmick S, Nagarajaiah S, Veeraraghavan A. Vision and deep learning-based algorithms to detect and quantify cracks on concrete surfaces from uav videos. Sensors. 2020;20(21):6299.
32. Goh GB, Hodas NO, Vishnu A. Deep learning for computational chemistry. J Comput Chem. 2017;38(16):1291–307.
33. Li Y, Zhang T, Sun S, Gao X. Accelerating flash calculation through deep learning methods. J Comput Phys. 2019;394:153–65.
34. Yang W, Zhang X, Tian Y, Wang W, Xue JH, Liao Q. Deep learning for single image super-resolution: a brief review. IEEE Trans Multimed. 2019;21(12):3106–21.
35. Tang J, Li S, Liu P. A review of lane detection methods based on deep learning. Pattern Recogn. 2020;111:107623.
36. Zhao ZQ, Zheng P, Xu ST, Wu X. Object detection with deep learning: a review. IEEE Trans Neural Netw Learn Syst. 2019;30(11):3212–32.
37. 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. p. 770–8.
38. Ng A. Machine learning yearning: technical strategy for AI engineers in the era of deep learning. 2019. https://www.mlyearning.org.
39. Metz C. Turing award won by 3 pioneers in artificial intelligence. The New York Times. 2019;27.
40. Nevo S, Anisimov V, Elidan G, El-Yaniv R, Giencke P, Gigi Y, Hassidim A, Moshe Z, Schlesinger M, Shalev G, et al. Ml for flood forecasting at scale; 2019. arXiv preprint arXiv:1901.09583.
41. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T. The rise of deep learning in drug discovery. Drug Discov Today. 2018;23(6):1241–50.
42. Benhammou Y, Achchab B, Herrera F, Tabik S. Breakhis based breast cancer automatic diagnosis using deep learning: taxonomy, survey and insights. Neurocomputing. 2020;375:9–24.
43. Wulczyn E, Steiner DF, Xu Z, Sadhwani A, Wang H, Flament-Auvigne I, Mermel CH, Chen PHC, Liu Y, Stumpe MC. Deep learning-based survival prediction for multiple cancer types using histopathology images. PLoS ONE. 2020;15(6):e0233678.
44. Nagpal K, Foote D, Liu Y, Chen PHC, Wulczyn E, Tan F, Olson N, Smith JL, Mohtashamian A, Wren JH, et al. Development and validation of a deep learning algorithm for improving Gleason scoring of prostate cancer. NPJ Digit Med. 2019;2(1):1–10.
45. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S. Dermatologist-level classification of skin cancer with deep neural networks. Nature. 2017;542(7639):115–8.
46. Brunese L, Mercaldo F, Reginelli A, Santone A. Explainable deep learning for pulmonary disease and coronavirus COVID-19 detection from X-rays. Comput Methods Programs Biomed. 2020;196(105):608.
47. Jamshidi M, Lalbakhsh A, Talla J, Peroutka Z, Hadjilooei F, Lalbakhsh P, Jamshidi M, La Spada L, Mirmozafari M, Dehghani M, et al. Artificial intelligence and COVID-19: deep learning approaches for diagnosis and treatment. IEEE Access. 2020;8:109581–95.
48. Shorfuzzaman M, Hossain MS. Metacovid: a siamese neural network framework with contrastive loss for n-shot diagnosis of COVID-19 patients. Pattern Recogn. 2020;113:107700.
49. Carvelli L, Olesen AN, Brink-Kjær A, Leary EB, Peppard PE, Mignot E, Sørensen HB, Jennum P. Design of a deep learning model for automatic scoring of periodic and non-periodic leg movements during sleep validated against multiple human experts. Sleep Med. 2020;69:109–19.
50. De Fauw J, Ledsam JR, Romera-Paredes B, Nikolov S, Tomasev N, Blackwell S, Askham H, Glorot X, O'Donoghue B, Visentin D, et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. Nat Med. 2018;24(9):1342–50.
51. Topol EJ. High-performance medicine: the convergence of human and artificial intelligence. Nat Med. 2019;25(1):44–56.
52. Kermany DS, Goldbaum M, Cai W, Valentim CC, Liang H, Baxter SL, McKeown A, Yang G, Wu X, Yan F, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. Cell. 2018;172(5):1122–31.
53. Van Essen B, Kim H, Pearce R, Boakye K, Chen B. Lbann: livermore big artificial neural network HPC toolkit. In: Proceedings of the workshop on machine learning in high-performance computing environments; 2015. p. 1–6.
54. Saeed MM, Al Aghbari Z, Alsharidah M. Big data clustering techniques based on spark: a literature review. PeerJ Comput Sci. 2020;6:321.

55. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529–33.
56. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. Deep reinforcement learning: a brief survey. IEEE Signal Process Mag. 2017;34(6):26–38.
57. Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, Potts C. Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 conference on empirical methods in natural language processing; 2013. p. 1631–42.
58. Goller C, Kuchler A. Learning task-dependent distributed representations by backpropagation through structure. In: Proceedings of international conference on neural networks (ICNN'96), vol 1. IEEE; 1996. p. 347–52.
59. Socher R, Lin CCY, Ng AY, Manning CD. Parsing natural scenes and natural language with recursive neural networks. In: ICML; 2011.
60. Louppe G, Cho K, Becot C, Cranmer K. QCD-aware recursive neural networks for jet physics. J High Energy Phys. 2019;2019(1):57.
61. Sadr H, Pedram MM, Teshnehlab M. A robust sentiment analysis method based on sequential combination of convolutional and recursive neural networks. Neural Process Lett. 2019;50(3):2745–61.
62. Urban G, Subrahmanya N, Baldi P. Inner and outer recursive neural networks for chemoinformatics applications. J Chem Inf Model. 2018;58(2):207–11.
63. Hewamalage H, Bergmeir C, Bandara K. Recurrent neural networks for time series forecasting: current status and future directions. Int J Forecast. 2020;37(1):388–427.
64. Jiang Y, Kim H, Asnani H, Kannan S, Oh S, Viswanath P. Learn codes: inventing low-latency codes via recurrent neural networks. IEEE J Sel Areas Inf Theory. 2020;1(1):207–16.
65. John RA, Acharya J, Zhu C, Surendran A, Bose SK, Chaturvedi A, Tiwari N, Gao Y, He Y, Zhang KK, et al. Optogenetics inspired transition metal dichalcogenide neuristors for in-memory deep recurrent neural networks. Nat Commun. 2020;11(1):1–9.
66. Batur Dinler Ö, Aydin N. An optimal feature parameter set based on gated recurrent unit recurrent neural networks for speech segment detection. Appl Sci. 2020;10(4):1273.
67. Jagannatha AN, Yu H. Structured prediction models for RNN based sequence labeling in clinical text. In: Proceedings of the conference on empirical methods in natural language processing. conference on empirical methods in natural language processing, vol. 2016, NIH Public Access; 2016. p. 856.
68. Pascanu R, Gulcehre C, Cho K, Bengio Y. How to construct deep recurrent neural networks. In: Proceedings of the second international conference on learning representations (ICLR 2014); 2014.
69. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics; 2010. p. 249–56.
70. Gao C, Yan J, Zhou S, Varshney PK, Liu H. Long short-term memory-based deep recurrent neural networks for target tracking. Inf Sci. 2019;502:279–96.
71. Zhou DX. Theory of deep convolutional neural networks: downsampling. Neural Netw. 2020;124:319–27.
72. Jhong SY, Tseng PY, Siriphockpirom N, Hsia CH, Huang MS, Hua KL, Chen YY. An automated biometric identification system using CNN-based palm vein recognition. In: 2020 international conference on advanced robotics and intelligent systems (ARIS). IEEE; 2020. p. 1–6.
73. Al-Azzawi A, Ouadou A, Max H, Duan Y, Tanner JJ, Cheng J. Deepcryopicker: fully automated deep neural network for single protein particle picking in cryo-EM. BMC Bioinform. 2020;21(1):1–38.
74. Wang T, Lu C, Yang M, Hong F, Liu C. A hybrid method for heartbeat classification via convolutional neural networks, multilayer perceptrons and focal loss. PeerJ Comput Sci. 2020;6:324.
75. Li G, Zhang M, Li J, Lv F, Tong G. Efficient densely connected convolutional neural networks. Pattern Recogn. 2021;109:107610.
76. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, et al. Recent advances in convolutional neural networks. Pattern Recogn. 2018;77:354–77.
77. Fang W, Love PE, Luo H, Ding L. Computer vision for behaviour-based safety in construction: a review and future directions. Adv Eng Inform. 2020;43:100980.
78. Palaz D, Magimai-Doss M, Collobert R. End-to-end acoustic modeling using convolutional neural networks for hmm-based automatic speech recognition. Speech Commun. 2019;108:15–32.
79. Li HC, Deng ZY, Chiang HH. Lightweight and resource-constrained learning network for face recognition with performance optimization. Sensors. 2020;20(21):6114.
80. Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol. 1962;160(1):106.
81. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift; 2015. arXiv preprint arXiv:1502.03167.
82. Ruder S. An overview of gradient descent optimization algorithms; 2016. arXiv preprint arXiv:1609.04747.
83. Bottou L. Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer; 2010. p. 177–86.
84. Hinton G, Srivastava N, Swersky K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on. 2012;14(8).
85. Zhang Z. Improved Adam optimizer for deep neural networks. In: 2018 IEEE/ACM 26th international symposium on quality of service (IWQoS). IEEE; 2018. p. 1–2.
86. Alzubaidi L, Fadhel MA, Al-Shamma O, Zhang J, Duan Y. Deep learning models for classification of red blood cells in microscopy images to aid in sickle cell anemia diagnosis. Electronics. 2020;9(3):427.
87. Alzubaidi L, Fadhel MA, Al-Shamma O, Zhang J, Santamaría J, Duan Y, Oleiwi SR. Towards a better understanding of transfer learning for medical imaging: a case study. Appl Sci. 2020;10(13):4523.
88. Alzubaidi L, Al-Shamma O, Fadhel MA, Farhan L, Zhang J, Duan Y. Optimizing the performance of breast cancer classification by employing the same domain transfer learning from hybrid deep convolutional neural network model. Electronics. 2020;9(3):445.

89. LeCun Y, Jackel LD, Bottou L, Cortes C, Denker JS, Drucker H, Guyon I, Muller UA, Sackinger E, Simard P, et al. Learning algorithms for classification: a comparison on handwritten digit recognition. Neural Netw Stat Mech Perspect. 1995;261:276.
90. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
91. Dahl GE, Sainath TN, Hinton GE. Improving deep neural networks for LVCSR using rectified linear units and dropout. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE; 2013. p. 8609–13.
92. Xu B, Wang N, Chen T, Li M. Empirical evaluation of rectified activations in convolutional network; 2015. arXiv preprint arXiv:1505.00853.
93. Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int J Uncertain Fuzziness Knowl Based Syst. 1998;6(02):107–16.
94. Lin M, Chen Q, Yan S. Network in network; 2013. arXiv preprint arXiv:1312.4400.
95. Hsiao TY, Chang YC, Chou HH, Chiu CT. Filter-based deep-compression with global average pooling for convolutional networks. J Syst Arch. 2019;95:9–18.
96. Li Z, Wang SH, Fan RR, Cao G, Zhang YD, Guo T. Teeth category classification via seven-layer deep convolutional neural network with max pooling and global average pooling. Int J Imaging Syst Technol. 2019;29(4):577–83.
97. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer; 2014. p. 818–33.
98. Erhan D, Bengio Y, Courville A, Vincent P. Visualizing higher-layer features of a deep network. Univ Montreal. 2009;1341(3):1.
99. Le QV. Building high-level features using large scale unsupervised learning. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE; 2013. p. 8595–8.
100. Grün F, Rupprecht C, Navab N, Tombari F. A taxonomy and library for visualizing learned features in convolutional neural networks; 2016. arXiv preprint arXiv:1606.07757.
101. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition; 2014. arXiv preprint arXiv:1409.1556.
102. Ranzato M, Huang FJ, Boureau YL, LeCun Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: 2007 IEEE conference on computer vision and pattern recognition. IEEE; 2007. p. 1–8.
103. 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 IEEE conference on computer vision and pattern recognition; 2015. p. 1–9.
104. Bengio Y, et al. Rmsprop and equilibrated adaptive learning rates for nonconvex optimization; 2015. arXiv:1502.04390corr abs/1502.04390
105. Srivastava RK, Greff K, Schmidhuber J. Highway networks; 2015. arXiv preprint arXiv:1505.00387.
106. Kong W, Dong ZY, Jia Y, Hill DJ, Xu Y, Zhang Y. Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Trans Smart Grid. 2017;10(1):841–51.
107. Ordóñez FJ, Roggen D. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors. 2016;16(1):115.
108. CireşAn D, Meier U, Masci J, Schmidhuber J. Multi-column deep neural network for traffic sign classification. Neural Netw. 2012;32:333–8.
109. Szegedy C, Ioffe S, Vanhoucke V, Alemi A. Inception-v4, inception-resnet and the impact of residual connections on learning; 2016. arXiv preprint arXiv:1602.07261.
110. 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. p. 2818–26.
111. Wu S, Zhong S, Liu Y. Deep residual learning for image steganalysis. Multimed Tools Appl. 2018;77(9):10437–53.
112. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 4700–08.
113. Rubin J, Parvaneh S, Rahman A, Conroy B, Babaeizadeh S. Densely connected convolutional networks for detection of atrial fibrillation from short single-lead ECG recordings. J Electrocardiol. 2018;51(6):S18-21.
114. Kuang P, Ma T, Chen Z, Li F. Image super-resolution with densely connected convolutional networks. Appl Intell. 2019;49(1):125–36.
115. Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 1492–500.
116. Su A, He X, Zhao X. Jpeg steganalysis based on ResNeXt with gauss partial derivative filters. Multimed Tools Appl. 2020;80(3):3349–66.
117. Yadav D, Jalal A, Garlapati D, Hossain K, Goyal A, Pant G. Deep learning-based ResNeXt model in phycological studies for future. Algal Res. 2020;50:102018.
118. Han W, Feng R, Wang L, Gao L. Adaptive spatial-scale-aware deep convolutional neural network for high-resolution remote sensing imagery scene classification. In: IGARSS 2018-2018 IEEE international geoscience and remote sensing symposium. IEEE; 2018. p. 4736–9.
119. Zagoruyko S, Komodakis N. Wide residual networks; 2016. arXiv preprint arXiv:1605.07146.
120. Huang G, Sun Y, Liu Z, Sedra D, Weinberger KQ. Deep networks with stochastic depth. In: European conference on computer vision. Springer; 2016. p. 646–61.
121. Huynh HT, Nguyen H. Joint age estimation and gender classification of Asian faces using wide ResNet. SN Comput Sci. 2020;1(5):1–9.
122. Takahashi R, Matsubara T, Uehara K. Data augmentation using random image cropping and patching for deep cnns. IEEE Trans Circuits Syst Video Technol. 2019;30(9):2917–31.
123. Han D, Kim J, Kim J. Deep pyramidal residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 5927–35.
124. Wang Y, Wang L, Wang H, Li P. End-to-end image super-resolution via deep and shallow convolutional networks. IEEE Access. 2019;7:31959–70.

125. Chollet F. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 1251–8.
126. Lo WW, Yang X, Wang Y. An xception convolutional neural network for malware classification with transfer learning. In: 2019 10th IFIP international conference on new technologies, mobility and security (NTMS). IEEE; 2019. p. 1–5.
127. Rahimzadeh M, Attar A. A modified deep convolutional neural network for detecting COVID-19 and pneumonia from chest X-ray images based on the concatenation of xception and resnet50v2. Inform Med Unlocked. 2020;19:100360.
128. Wang F, Jiang M, Qian C, Yang S, Li C, Zhang H, Wang X, Tang X. Residual attention network for image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 3156–64.
129. Salakhutdinov R, Larochelle H. Efficient learning of deep boltzmann machines. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics; 2010. p. 693–700.
130. Goh H, Thome N, Cord M, Lim JH. Top-down regularization of deep belief networks. Adv Neural Inf Process Syst. 2013;26:1878–86.
131. Guan J, Lai R, Xiong A, Liu Z, Gu L. Fixed pattern noise reduction for infrared images based on cascade residual attention CNN. Neurocomputing. 2020;377:301–13.
132. Bi Q, Qin K, Zhang H, Li Z, Xu K. RADC-Net: a residual attention based convolution network for aerial scene classification. Neurocomputing. 2020;377:345–59.
133. Jaderberg M, Simonyan K, Zisserman A, et al. Spatial transformer networks. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2015. p. 2017–25.
134. Hu J, Shen L, Sun G. Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 7132–41.
135. Mou L, Zhu XX. Learning to pay attention on spectral domain: a spectral attention module-based convolutional network for hyperspectral image classification. IEEE Trans Geosci Remote Sens. 2019;58(1):110–22.
136. Woo S, Park J, Lee JY, So Kweon I. CBAM: Convolutional block attention module. In: Proceedings of the European conference on computer vision (ECCV); 2018. p. 3–19.
137. Roy AG, Navab N, Wachinger C. Concurrent spatial and channel 'squeeze & excitation' in fully convolutional networks. In: International conference on medical image computing and computer-assisted intervention. Springer; 2018. p. 421–9.
138. Roy AG, Navab N, Wachinger C. Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks. IEEE Trans Med Imaging. 2018;38(2):540–9.
139. Sabour S, Frosst N, Hinton GE. Dynamic routing between capsules. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2017. p. 3856–66.
140. Arun P, Buddhiraju KM, Porwal A. Capsulenet-based spatial-spectral classifier for hyperspectral images. IEEE J Sel Topics Appl Earth Obs Remote Sens. 2019;12(6):1849–65.
141. Xinwei L, Lianghao X, Yi Y. Compact video fingerprinting via an improved capsule net. Syst Sci Control Eng. 2020;9:1–9.
142. Ma B, Li X, Xia Y, Zhang Y. Autonomous deep learning: a genetic DCNN designer for image classification. Neurocomputing. 2020;379:152–61.
143. Wang J, Sun K, Cheng T, Jiang B, Deng C, Zhao Y, Liu D, Mu Y, Tan M, Wang X, et al. Deep high-resolution representation learning for visual recognition. IEEE Trans Pattern Anal Mach Intell. 2020. https://doi.org/10.1109/TPAMI.2020.2983686.
144. Cheng B, Xiao B, Wang J, Shi H, Huang TS, Zhang L. Higherhrnet: scale-aware representation learning for bottom-up human pose estimation. In: CVPR 2020; 2020. https://www.microsoft.com/en-us/research/publication/higherhrnet-scale-aware-representation-learning-for-bottom-up-human-pose-estimation/.
145. Karimi H, Derr T, Tang J. Characterizing the decision boundary of deep neural networks; 2019. arXiv preprint arXiv:1912.11460.
146. Li Y, Ding L, Gao X. On the decision boundary of deep neural networks; 2018. arXiv preprint arXiv:1808.05385.
147. Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks? In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2014. p. 3320–8.
148. Tan C, Sun F, Kong T, Zhang W, Yang C, Liu C. A survey on deep transfer learning. In: International conference on artificial neural networks. Springer; 2018. p. 270–9.
149. Weiss K, Khoshgoftaar TM, Wang D. A survey of transfer learning. J Big Data. 2016;3(1):9.
150. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data. 2019;6(1):60.
151. Wang F, Wang H, Wang H, Li G, Situ G. Learning from simulation: an end-to-end deep-learning approach for computational ghost imaging. Opt Express. 2019;27(18):25560–72.
152. Pan W. A survey of transfer learning for collaborative recommendation with auxiliary data. Neurocomputing. 2016;177:447–53.
153. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. IEEE; 2009. p. 248–55.
154. Cook D, Feuz KD, Krishnan NC. Transfer learning for activity recognition: a survey. Knowl Inf Syst. 2013;36(3):537–56.
155. Cao X, Wang Z, Yan P, Li X. Transfer learning for pedestrian detection. Neurocomputing. 2013;100:51–7.
156. Raghu M, Zhang C, Kleinberg J, Bengio S. Transfusion: understanding transfer learning for medical imaging. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2019. p. 3347–57.
157. Pham TN, Van Tran L, Dao SVT. Early disease classification of mango leaves using feed-forward neural network and hybrid metaheuristic feature selection. IEEE Access. 2020;8:189960–73.
158. Saleh AM, Hamoud T. Analysis and best parameters selection for person recognition based on gait model using CNN algorithm and image augmentation. J Big Data. 2021;8(1):1–20.
159. Hirahara D, Takaya E, Takahara T, Ueda T. Effects of data count and image scaling on deep learning training. PeerJ Comput Sci. 2020;6:312.

160. Moreno-Barea FJ, Strazzera F, Jerez JM, Urda D, Franco L. Forward noise adjustment scheme for data augmentation. In: 2018 IEEE symposium series on computational intelligence (SSCI). IEEE; 2018. p. 728–34.
161. Dua D, Karra Taniskidou E. Uci machine learning repository. Irvine: University of california. School of Information and Computer Science; 2017. http://archive.ics.uci.edu/ml
162. Johnson JM, Khoshgoftaar TM. Survey on deep learning with class imbalance. J Big Data. 2019;6(1):27.
163. Yang P, Zhang Z, Zhou BB, Zomaya AY. Sample subset optimization for classifying imbalanced biological data. In: Pacific-Asia conference on knowledge discovery and data mining. Springer; 2011. p. 333–44.
164. Yang P, Yoo PD, Fernando J, Zhou BB, Zhang Z, Zomaya AY. Sample subset optimization techniques for imbalanced and ensemble learning problems in bioinformatics applications. IEEE Trans Cybern. 2013;44(3):445–55.
165. Wang S, Sun S, Xu J. Auc-maximized deep convolutional neural fields for sequence labeling 2015. arXiv preprint arXiv:1511.05265.
166. Li Y, Wang S, Umarov R, Xie B, Fan M, Li L, Gao X. Deepre: sequence-based enzyme EC number prediction by deep learning. Bioinformatics. 2018;34(5):760–9.
167. Li Y, Huang C, Ding L, Li Z, Pan Y, Gao X. Deep learning in bioinformatics: introduction, application, and perspective in the big data era. Methods. 2019;166:4–21.
168. Choi E, Bahadori MT, Sun J, Kulas J, Schuetz A, Stewart W. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2016. p. 3504–12.
169. Ching T, Himmelstein DS, Beaulieu-Jones BK, Kalinin AA, Do BT, Way GP, Ferrero E, Agapow PM, Zietz M, Hoffman MM, et al. Opportunities and obstacles for deep learning in biology and medicine. J R Soc Interface. 2018;15(141):20170,387.
170. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods. 2015;12(10):931–4.
171. Pokuri BSS, Ghosal S, Kokate A, Sarkar S, Ganapathysubramanian B. Interpretable deep learning for guided microstructure-property explorations in photovoltaics. NPJ Comput Mater. 2019;5(1):1–11.
172. Ribeiro MT, Singh S, Guestrin C. "Why should I trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining; 2016. p. 1135–44.
173. Wang L, Nie R, Yu Z, Xin R, Zheng C, Zhang Z, Zhang J, Cai J. An interpretable deep-learning architecture of capsule networks for identifying cell-type gene expression programs from single-cell RNA-sequencing data. Nat Mach Intell. 2020;2(11):1–11.
174. Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks; 2017. arXiv preprint arXiv:1703.01365.
175. Platt J, et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Adv Large Margin Classif. 1999;10(3):61–74.
176. Nair T, Precup D, Arnold DL, Arbel T. Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. Med Image Anal. 2020;59:101557.
177. Herzog L, Murina E, Dürr O, Wegener S, Sick B. Integrating uncertainty in deep neural networks for MRI based stroke analysis. Med Image Anal. 2020;65:101790.
178. Pereyra G, Tucker G, Chorowski J, Kaiser Ł, Hinton G. Regularizing neural networks by penalizing confident output distributions; 2017. arXiv preprint arXiv:1701.06548.
179. Naeini MP, Cooper GF, Hauskrecht M. Obtaining well calibrated probabilities using bayesian binning. In: Proceedings of the... AAAI conference on artificial intelligence. AAAI conference on artificial intelligence, vol. 2015. NIH Public Access; 2015. p. 2901.
180. Li M, Sethi IK. Confidence-based classifier design. Pattern Recogn. 2006;39(7):1230–40.
181. Zadrozny B, Elkan C. Obtaining calibrated probability estimates from decision trees and Naive Bayesian classifiers. In: ICML, vol. 1, Citeseer; 2001. p. 609–16.
182. Steinwart I. Consistency of support vector machines and other regularized kernel classifiers. IEEE Trans Inf Theory. 2005;51(1):128–42.
183. Lee K, Lee K, Shin J, Lee H. Overcoming catastrophic forgetting with unlabeled data in the wild. In: Proceedings of the IEEE international conference on computer vision; 2019. p. 312–21.
184. Shmelkov K, Schmid C, Alahari K. Incremental learning of object detectors without catastrophic forgetting. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 3400–09.
185. Zenke F, Gerstner W, Ganguli S. The temporal paradox of Hebbian learning and homeostatic plasticity. Curr Opin Neurobiol. 2017;43:166–76.
186. Andersen N, Krauth N, Nabavi S. Hebbian plasticity in vivo: relevance and induction. Curr Opin Neurobiol. 2017;45:188–92.
187. Zheng R, Chakraborti S. A phase ii nonparametric adaptive exponentially weighted moving average control chart. Qual Eng. 2016;28(4):476–90.
188. Rebuffi SA, Kolesnikov A, Sperl G, Lampert CH. ICARL: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 2001–10.
189. Hinton GE, Plaut DC. Using fast weights to deblur old memories. In: Proceedings of the ninth annual conference of the cognitive science society; 1987. p. 177–86.
190. Parisi GI, Kemker R, Part JL, Kanan C, Wermter S. Continual lifelong learning with neural networks: a review. Neural Netw. 2019;113:54–71.
191. Soltoggio A, Stanley KO, Risi S. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. Neural Netw. 2018;108:48–67.
192. Parisi GI, Tani J, Weber C, Wermter S. Lifelong learning of human actions with deep neural network self-organization. Neural Netw. 2017;96:137–49.
193. Cheng Y, Wang D, Zhou P, Zhang T. Model compression and acceleration for deep neural networks: the principles, progress, and challenges. IEEE Signal Process Mag. 2018;35(1):126–36.

194. Wiedemann S, Kirchhoffer H, Matlage S, Haase P, Marban A, Marinč T, Neumann D, Nguyen T, Schwarz H, Wiegand T, et al. Deepcabac: a universal compression algorithm for deep neural networks. IEEE J Sel Topics Signal Process. 2020;14(4):700–14.
195. Mehta N, Pandit A. Concurrence of big data analytics and healthcare: a systematic review. Int J Med Inform. 2018;114:57–65.
196. Esteva A, Robicquet A, Ramsundar B, Kuleshov V, DePristo M, Chou K, Cui C, Corrado G, Thrun S, Dean J. A guide to deep learning in healthcare. Nat Med. 2019;25(1):24–9.
197. Shawahna A, Sait SM, El-Maleh A. Fpga-based accelerators of deep learning networks for learning and classification: a review. IEEE Access. 2018;7:7823–59.
198. Min Z. Public welfare organization management system based on FPGA and deep learning. Microprocess Microsyst. 2020;80:103333.
199. Al-Shamma O, Fadhel MA, Hameed RA, Alzubaidi L, Zhang J. Boosting convolutional neural networks performance based on fpga accelerator. In: International conference on intelligent systems design and applications. Springer; 2018. p. 509–17.
200. Han S, Mao H, Dally WJ. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding; 2015. arXiv preprint arXiv:1510.00149.
201. Chen Z, Zhang L, Cao Z, Guo J. Distilling the knowledge from handcrafted features for human activity recognition. IEEE Trans Ind Inform. 2018;14(10):4334–42.
202. Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network; 2015. arXiv preprint arXiv:1503.02531.
203. Lenssen JE, Fey M, Libuschewski P. Group equivariant capsule networks. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2018. p. 8844–53.
204. Denton EL, Zaremba W, Bruna J, LeCun Y, Fergus R. Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2014. p. 1269–77.
205. Xu Q, Zhang M, Gu Z, Pan G. Overfitting remedy by sparsifying regularization on fully-connected layers of CNNs. Neurocomputing. 2019;328:69–74.
206. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning requires rethinking generalization. Commun ACM. 2018;64(3):107–15.
207. Xu X, Jiang X, Ma C, Du P, Li X, Lv S, Yu L, Ni Q, Chen Y, Su J, et al. A deep learning system to screen novel coronavirus disease 2019 pneumonia. Engineering. 2020;6(10):1122–9.
208. Sharma K, Alsadoon A, Prasad P, Al-Dala'in T, Nguyen TQV, Pham DTH. A novel solution of using deep learning for left ventricle detection: enhanced feature extraction. Comput Methods Programs Biomed. 2020;197:105751.
209. Zhang G, Wang C, Xu B, Grosse R. Three mechanisms of weight decay regularization; 2018. arXiv preprint arXiv:1810.12281.
210. Laurent C, Pereyra G, Brakel P, Zhang Y, Bengio Y. Batch normalized recurrent neural networks. In: 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE; 2016. p. 2657–61.
211. Salamon J, Bello JP. Deep convolutional neural networks and data augmentation for environmental sound classification. IEEE Signal Process Lett. 2017;24(3):279–83.
212. Wang X, Qin Y, Wang Y, Xiang S, Chen H. ReLTanh: an activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis. Neurocomputing. 2019;363:88–98.
213. Tan HH, Lim KH. Vanishing gradient mitigation with deep learning neural network optimization. In: 2019 7th international conference on smart computing & communications (ICSCC). IEEE; 2019. p. 1–4.
214. MacDonald G, Godbout A, Gillcash B, Cairns S. Volume-preserving neural networks: a solution to the vanishing gradient problem; 2019. arXiv preprint arXiv:1911.09576.
215. Mittal S, Vaishay S. A survey of techniques for optimizing deep learning on GPUs. J Syst Arch. 2019;99:101635.
216. Kanai S, Fujiwara Y, Iwamura S. Preventing gradient explosions in gated recurrent units. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2017. p. 435–44.
217. Hanin B. Which neural net architectures give rise to exploding and vanishing gradients? In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2018. p. 582–91.
218. Ribeiro AH, Tiels K, Aguirre LA, Schön T. Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness. In: International conference on artificial intelligence and statistics, PMLR; 2020. p. 2370–80.
219. D'Amour A, Heller K, Moldovan D, Adlam B, Alipanahi B, Beutel A, Chen C, Deaton J, Eisenstein J, Hoffman MD, et al. Underspecification presents challenges for credibility in modern machine learning; 2020. arXiv preprint arXiv:2011.03395.
220. Chea P, Mandell JC. Current applications and future directions of deep learning in musculoskeletal radiology. Skelet Radiol. 2020;49(2):1–15.
221. Wu X, Sahoo D, Hoi SC. Recent advances in deep learning for object detection. Neurocomputing. 2020;396:39–64.
222. Kuutti S, Bowden R, Jin Y, Barber P, Fallah S. A survey of deep learning applications to autonomous vehicle control. IEEE Trans Intell Transp Syst. 2020;22:712–33.
223. Yolcu G, Oztel I, Kazan S, Oz C, Bunyak F. Deep learning-based face analysis system for monitoring customer interest. J Ambient Intell Humaniz Comput. 2020;11(1):237–48.
224. Jiao L, Zhang F, Liu F, Yang S, Li L, Feng Z, Qu R. A survey of deep learning-based object detection. IEEE Access. 2019;7:128837–68.
225. Muhammad K, Khan S, Del Ser J, de Albuquerque VHC. Deep learning for multigrade brain tumor classification in smart healthcare systems: a prospective survey. IEEE Trans Neural Netw Learn Syst. 2020;32:507–22.
226. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, Van Der Laak JA, Van Ginneken B, Sánchez CI. A survey on deep learning in medical image analysis. Med Image Anal. 2017;42:60–88.
227. Mukherjee D, Mondal R, Singh PK, Sarkar R, Bhattacharjee D. Ensemconvnet: a deep learning approach for human activity recognition using smartphone sensors for healthcare applications. Multimed Tools Appl. 2020;79(41):31663–90.

228. Zeleznik R, Foldyna B, Eslami P, Weiss J, Alexander I, Taron J, Parmar C, Alvi RM, Banerji D, Uno M, et al. Deep convolutional neural networks to predict cardiovascular risk from computed tomography. Nature Commun. 2021;12(1):1–9.
229. Wang J, Liu Q, Xie H, Yang Z, Zhou H. Boosted efficientnet: detection of lymph node metastases in breast cancer using convolutional neural networks. Cancers. 2021;13(4):661.
230. Yu H, Yang LT, Zhang Q, Armstrong D, Deen MJ. Convolutional neural networks for medical image analysis: state-of-the-art, comparisons, improvement and perspectives. Neurocomputing. 2021. https://doi.org/10.1016/j.neucom.2020.04.157.
231. Bharati S, Podder P, Mondal MRH. Hybrid deep learning for detecting lung diseases from X-ray images. Inform Med Unlocked. 2020;20:100391.
232. Dong Y, Pan Y, Zhang J, Xu W. Learning to read chest X-ray images from 16000+ examples using CNN. In: 2017 IEEE/ACM international conference on connected health: applications, systems and engineering technologies (CHASE). IEEE; 2017. p. 51–7.
233. Rajkomar A, Lingam S, Taylor AG, Blum M, Mongan J. High-throughput classification of radiographs using deep convolutional neural networks. J Digit Imaging. 2017;30(1):95–101.
234. Rajpurkar P, Irvin J, Zhu K, Yang B, Mehta H, Duan T, Ding D, Bagul A, Langlotz C, Shpanskaya K, et al. Chexnet: radiologist-level pneumonia detection on chest X-rays with deep learning; 2017. arXiv preprint arXiv:1711.05225.
235. Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 2097–106.
236. Zuo W, Zhou F, Li Z, Wang L. Multi-resolution CNN and knowledge transfer for candidate classification in lung nodule detection. IEEE Access. 2019;7:32510–21.
237. Shen W, Zhou M, Yang F, Yang C, Tian J. Multi-scale convolutional neural networks for lung nodule classification. In: International conference on information processing in medical imaging. Springer; 2015. p. 588–99.
238. Li R, Zhang W, Suk HI, Wang L, Li J, Shen D, Ji S. Deep learning based imaging data completion for improved brain disease diagnosis. In: International conference on medical image computing and computer-assisted intervention. Springer; 2014. p. 305–12.
239. Wen J, Thibeau-Sutre E, Diaz-Melo M, Samper-González J, Routier A, Bottani S, Dormont D, Durrleman S, Burgos N, Colliot O, et al. Convolutional neural networks for classification of Alzheimer's disease: overview and reproducible evaluation. Med Image Anal. 2020;63:101694.
240. Mehmood A, Maqsood M, Bashir M, Shuyuan Y. A deep siamese convolution neural network for multi-class classification of Alzheimer disease. Brain Sci. 2020;10(2):84.
241. Hosseini-Asl E, Ghazal M, Mahmoud A, Aslantas A, Shalaby A, Casanova M, Barnes G, Gimel'farb G, Keynton R, El-Baz A. Alzheimer's disease diagnostics by a 3d deeply supervised adaptable convolutional network. Front Biosci. 2018;23:584–96.
242. Korolev S, Safiullin A, Belyaev M, Dodonova Y. Residual and plain convolutional neural networks for 3D brain MRI classification. In: 2017 IEEE 14th international symposium on biomedical imaging (ISBI 2017). IEEE; 2017. p. 835–8.
243. Alzubaidi L, Fadhel MA, Oleiwi SR, Al-Shamma O, Zhang J. DFU_QUTNet: diabetic foot ulcer classification using novel deep convolutional neural network. Multimed Tools Appl. 2020;79(21):15655–77.
244. Goyal M, Reeves ND, Davison AK, Rajbhandari S, Spragg J, Yap MH. Dfunet: convolutional neural networks for diabetic foot ulcer classification. IEEE Trans Emerg Topics Comput Intell. 2018;4(5):728–39.
245. Yap MH., Hachiuma R, Alavi A, Brungel R, Goyal M, Zhu H, Cassidy B, Ruckert J, Olshansky M, Huang X, et al. Deep learning in diabetic foot ulcers detection: a comprehensive evaluation; 2020. arXiv preprint arXiv:2010.03341.
246. Tulloch J, Zamani R, Akrami M. Machine learning in the prevention, diagnosis and management of diabetic foot ulcers: a systematic review. IEEE Access. 2020;8:198977–9000.
247. Fadhel MA, Al-Shamma O, Alzubaidi L, Oleiwi SR. Real-time sickle cell anemia diagnosis based hardware accelerator. In: International conference on new trends in information and communications technology applications, Springer; 2020. p. 189–99.
248. Debelee TG, Kebede SR, Schwenker F, Shewarega ZM. Deep learning in selected cancers' image analysis—a survey. J Imaging. 2020;6(11):121.
249. Khan S, Islam N, Jan Z, Din IU, Rodrigues JJC. A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. Pattern Recogn Lett. 2019;125:1–6.
250. Alzubaidi L, Hasan RI, Awad FH, Fadhel MA, Alshamma O, Zhang J. Multi-class breast cancer classification by a novel two-branch deep convolutional neural network architecture. In: 2019 12th international conference on developments in eSystems engineering (DeSE). IEEE; 2019. p. 268–73.
251. Roy K, Banik D, Bhattacharjee D, Nasipuri M. Patch-based system for classification of breast histology images using deep learning. Comput Med Imaging Gr. 2019;71:90–103.
252. Hameed Z, Zahia S, Garcia-Zapirain B, Javier Aguirre J, María Vanegas A. Breast cancer histopathology image classification using an ensemble of deep learning models. Sensors. 2020;20(16):4373.
253. Hosny KM, Kassem MA, Foaud MM. Skin cancer classification using deep learning and transfer learning. In: 2018 9th Cairo international biomedical engineering conference (CIBEC). IEEE; 2018. p. 90–3.
254. Dorj UO, Lee KK, Choi JY, Lee M. The skin cancer classification using deep convolutional neural network. Multimed Tools Appl. 2018;77(8):9909–24.
255. Kassem MA, Hosny KM, Fouad MM. Skin lesions classification into eight classes for ISIC 2019 using deep convolutional neural network and transfer learning. IEEE Access. 2020;8:114822–32.
256. Heidari M, Mirniaharikandehei S, Khuzani AZ, Danala G, Qiu Y, Zheng B. Improving the performance of CNN to predict the likelihood of COVID-19 using chest X-ray images with preprocessing algorithms. Int J Med Inform. 2020;144:104284.
257. Al-Timemy AH, Khushaba RN, Mosa ZM, Escudero J. An efficient mixture of deep and machine learning models for COVID-19 and tuberculosis detection using X-ray images in resource limited settings 2020. arXiv preprint arXiv:2007.08223.

258. Abraham B, Nair MS. Computer-aided detection of COVID-19 from X-ray images using multi-CNN and Bayesnet classifier. Biocybern Biomed Eng. 2020;40(4):1436–45.
259. Nour M, Cömert Z, Polat K. A novel medical diagnosis model for COVID-19 infection detection based on deep features and Bayesian optimization. Appl Soft Comput. 2020;97:106580.
260. Mallio CA, Napolitano A, Castiello G, Giordano FM, D'Alessio P, Iozzino M, Sun Y, Angeletti S, Russano M, Santini D, et al. Deep learning algorithm trained with COVID-19 pneumonia also identifies immune checkpoint inhibitor therapy-related pneumonitis. Cancers. 2021;13(4):652.
261. Fourcade A, Khonsari R. Deep learning in medical image analysis: a third eye for doctors. J Stomatol Oral Maxillofac Surg. 2019;120(4):279–88.
262. Guo Z, Li X, Huang H, Guo N, Li Q. Deep learning-based image segmentation on multimodal medical imaging. IEEE Trans Radiat Plasma Med Sci. 2019;3(2):162–9.
263. Thakur N, Yoon H, Chong Y. Current trends of artificial intelligence for colorectal cancer pathology image analysis: a systematic review. Cancers. 2020;12(7):1884.
264. Lundervold AS, Lundervold A. An overview of deep learning in medical imaging focusing on MRI. Zeitschrift für Medizinische Physik. 2019;29(2):102–27.
265. Yadav SS, Jadhav SM. Deep convolutional neural network based medical image classification for disease diagnosis. J Big Data. 2019;6(1):113.
266. Nehme E, Freedman D, Gordon R, Ferdman B, Weiss LE, Alalouf O, Naor T, Orange R, Michaeli T, Shechtman Y. Deep-STORM3D: dense 3D localization microscopy and PSF design by deep learning. Nat Methods. 2020;17(7):734–40.
267. Zulkifley MA, Abdani SR, Zulkifley NH. Pterygium-Net: a deep learning approach to pterygium detection and localization. Multimed Tools Appl. 2019;78(24):34563–84.
268. Sirazitdinov I, Kholiavchenko M, Mustafaev T, Yixuan Y, Kuleev R, Ibragimov B. Deep neural network ensemble for pneumonia localization from a large-scale chest X-ray database. Comput Electr Eng. 2019;78:388–99.
269. Zhao W, Shen L, Han B, Yang Y, Cheng K, Toesca DA, Koong AC, Chang DT, Xing L. Markerless pancreatic tumor target localization enabled by deep learning. Int J Radiat Oncol Biol Phys. 2019;105(2):432–9.
270. Roth HR, Lee CT, Shin HC, Seff A, Kim L, Yao J, Lu L, Summers RM. Anatomy-specific classification of medical images using deep convolutional nets. In: 2015 IEEE 12th international symposium on biomedical imaging (ISBI). IEEE; 2015. p. 101–4.
271. Shin HC, Orton MR, Collins DJ, Doran SJ, Leach MO. Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data. IEEE Trans Pattern Anal Mach Intell. 2012;35(8):1930–43.
272. Li Z, Dong M, Wen S, Hu X, Zhou P, Zeng Z. CLU-CNNs: object detection for medical images. Neurocomputing. 2019;350:53–9.
273. Gao J, Jiang Q, Zhou B, Chen D. Convolutional neural networks for computer-aided detection or diagnosis in medical image analysis: an overview. Math Biosci Eng. 2019;16(6):6536.
274. Lumini A, Nanni L. Review fair comparison of skin detection approaches on publicly available datasets. Expert Syst Appl. 2020. https://doi.org/10.1016/j.eswa.2020.113677.
275. Chouhan V, Singh SK, Khamparia A, Gupta D, Tiwari P, Moreira C, Damaševičius R, De Albuquerque VHC. A novel transfer learning based approach for pneumonia detection in chest X-ray images. Appl Sci. 2020;10(2):559.
276. Apostolopoulos ID, Mpesiana TA. COVID-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. Phys Eng Sci Med. 2020;43(2):635–40.
277. Mahmud T, Rahman MA, Fattah SA. CovXNet: a multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization. Comput Biol Med. 2020;122:103869.
278. Tayarani-N MH. Applications of artificial intelligence in battling against COVID-19: a literature review. Chaos Solitons Fractals. 2020;142:110338.
279. Toraman S, Alakus TB, Turkoglu I. Convolutional capsnet: a novel artificial neural network approach to detect COVID-19 disease from X-ray images using capsule networks. Chaos Solitons Fractals. 2020;140:110122.
280. Dascalu A, David E. Skin cancer detection by deep learning and sound analysis algorithms: a prospective clinical study of an elementary dermoscope. EBioMedicine. 2019;43:107–13.
281. Adegun A, Viriri S. Deep learning techniques for skin lesion analysis and melanoma cancer detection: a survey of state-of-the-art. Artif Intell Rev. 2020;54:1–31.
282. Zhang N, Cai YX, Wang YY, Tian YT, Wang XL, Badami B. Skin cancer diagnosis based on optimized convolutional neural network. Artif Intell Med. 2020;102:101756.
283. Thurnhofer-Hemsi K, Domínguez E. A convolutional neural network framework for accurate skin cancer detection. Neural Process Lett. 2020. https://doi.org/10.1007/s11063-020-10364-y.
284. Jain MS, Massoud TF. Predicting tumour mutational burden from histopathological images using multiscale deep learning. Nat Mach Intell. 2020;2(6):356–62.
285. Lei H, Liu S, Elazab A, Lei B. Attention-guided multi-branch convolutional neural network for mitosis detection from histopathological images. IEEE J Biomed Health Inform. 2020;25(2):358–70.
286. Celik Y, Talo M, Yildirim O, Karabatak M, Acharya UR. Automated invasive ductal carcinoma detection based using deep transfer learning with whole-slide images. Pattern Recogn Lett. 2020;133:232–9.
287. Sebai M, Wang X, Wang T. Maskmitosis: a deep learning framework for fully supervised, weakly supervised, and unsupervised mitosis detection in histopathology images. Med Biol Eng Comput. 2020;58:1603–23.
288. Sebai M, Wang T, Al-Fadhli SA. Partmitosis: a partially supervised deep learning framework for mitosis detection in breast cancer histopathology images. IEEE Access. 2020;8:45133–47.
289. Mahmood T, Arsalan M, Owais M, Lee MB, Park KR. Artificial intelligence-based mitosis detection in breast cancer histopathology images using faster R-CNN and deep CNNs. J Clin Med. 2020;9(3):749.
290. Srinidhi CL, Ciga O, Martel AL. Deep neural network models for computational histopathology: a survey. Med Image Anal. 2020;67:101813.

291. Cireşan DC, Giusti A, Gambardella LM, Schmidhuber J. Mitosis detection in breast cancer histology images with deep neural networks. In: International conference on medical image computing and computer-assisted intervention. Springer; 2013. p. 411–8.
292. Sirinukunwattana K, Raza SEA, Tsang YW, Snead DR, Cree IA, Rajpoot NM. Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. IEEE Trans Med Imaging. 2016;35(5):1196–206.
293. Xu J, Xiang L, Liu Q, Gilmore H, Wu J, Tang J, Madabhushi A. Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. IEEE Trans Med Imaging. 2015;35(1):119–30.
294. Albarqouni S, Baur C, Achilles F, Belagiannis V, Demirci S, Navab N. Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images. IEEE Trans Med Imaging. 2016;35(5):1313–21.
295. Abd-Ellah MK, Awad AI, Khalaf AA, Hamed HF. Two-phase multi-model automatic brain tumour diagnosis system from magnetic resonance images using convolutional neural networks. EURASIP J Image Video Process. 2018;2018(1):97.
296. Thaha MM, Kumar KPM, Murugan B, Dhanasekeran S, Vijayakarthick P, Selvi AS. Brain tumor segmentation using convolutional neural networks in MRI images. J Med Syst. 2019;43(9):294.
297. Talo M, Yildirim O, Baloglu UB, Aydin G, Acharya UR. Convolutional neural networks for multi-class brain disease detection using MRI images. Comput Med Imaging Gr. 2019;78:101673.
298. Gabr RE, Coronado I, Robinson M, Sujit SJ, Datta S, Sun X, Allen WJ, Lublin FD, Wolinsky JS, Narayana PA. Brain and lesion segmentation in multiple sclerosis using fully convolutional neural networks: a large-scale study. Mult Scler J. 2020;26(10):1217–26.
299. Chen S, Ding C, Liu M. Dual-force convolutional neural networks for accurate brain tumor segmentation. Pattern Recogn. 2019;88:90–100.
300. Hu K, Gan Q, Zhang Y, Deng S, Xiao F, Huang W, Cao C, Gao X. Brain tumor segmentation using multi-cascaded convolutional neural networks and conditional random field. IEEE Access. 2019;7:92615–29.
301. Wadhwa A, Bhardwaj A, Verma VS. A review on brain tumor segmentation of MRI images. Magn Reson Imaging. 2019;61:247–59.
302. Akkus Z, Galimzianova A, Hoogi A, Rubin DL, Erickson BJ. Deep learning for brain MRI segmentation: state of the art and future directions. J Digit Imaging. 2017;30(4):449–59.
303. Moeskops P, Viergever MA, Mendrik AM, De Vries LS, Benders MJ, Išgum I. Automatic segmentation of MR brain images with a convolutional neural network. IEEE Trans Med Imaging. 2016;35(5):1252–61.
304. Milletari F, Navab N, Ahmadi SA. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 2016 fourth international conference on 3D vision (3DV). IEEE; 2016. p. 565–71.
305. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. Springer; 2015. p. 234–41.
306. Pereira S, Pinto A, Alves V, Silva CA. Brain tumor segmentation using convolutional neural networks in MRI images. IEEE Trans Med Imaging. 2016;35(5):1240–51.
307. Havaei M, Davy A, Warde-Farley D, Biard A, Courville A, Bengio Y, Pal C, Jodoin PM, Larochelle H. Brain tumor segmentation with deep neural networks. Med Image Anal. 2017;35:18–31.
308. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL. DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans Pattern Anal Mach Intell. 2017;40(4):834–48.
309. Yan Q, Wang B, Gong D, Luo C, Zhao W, Shen J, Shi Q, Jin S, Zhang L, You Z. COVID-19 chest CT image segmentation—a deep convolutional neural network solution; 2020. arXiv preprint arXiv:2004.10987.
310. Wang G, Liu X, Li C, Xu Z, Ruan J, Zhu H, Meng T, Li K, Huang N, Zhang S. A noise-robust framework for automatic segmentation of COVID-19 pneumonia lesions from CT images. IEEE Trans Med Imaging. 2020;39(8):2653–63.
311. Khan SH, Sohail A, Khan A, Lee YS. Classification and region analysis of COVID-19 infection using lung CT images and deep convolutional neural networks; 2020. arXiv preprint arXiv:2009.08864.
312. Shi F, Wang J, Shi J, Wu Z, Wang Q, Tang Z, He K, Shi Y, Shen D. Review of artificial intelligence techniques in imaging data acquisition, segmentation and diagnosis for COVID-19. IEEE Rev Biomed Eng. 2020;14:4–5.
313. Santamaría J, Rivero-Cejudo M, Martos-Fernández M, Roca F. An overview on the latest nature-inspired and metaheuristics-based image registration algorithms. Appl Sci. 2020;10(6):1928.
314. Santamaría J, Cordón O, Damas S. A comparative study of state-of-the-art evolutionary image registration methods for 3D modeling. Comput Vision Image Underst. 2011;115(9):1340–54.
315. Yumer ME, Mitra NJ. Learning semantic deformation flows with 3D convolutional networks. In: European conference on computer vision. Springer; 2016. p. 294–311.
316. Ding L, Feng C. Deepmapping: unsupervised map estimation from multiple point clouds. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2019. p. 8650–9.
317. Mahadevan S. Imagination machines: a new challenge for artificial intelligence. AAAI. 2018;2018:7988–93.
318. Wang L, Fang Y. Unsupervised 3D reconstruction from a single image via adversarial learning; 2017. arXiv preprint arXiv:1711.09312.
319. Hermoza R, Sipiran I. 3D reconstruction of incomplete archaeological objects using a generative adversarial network. In: Proceedings of computer graphics international 2018. Association for Computing Machinery; 2018. p. 5–11.
320. Fu Y, Lei Y, Wang T, Curran WJ, Liu T, Yang X. Deep learning in medical image registration: a review. Phys Med Biol. 2020;65(20):20TR01.
321. Haskins G, Kruger U, Yan P. Deep learning in medical image registration: a survey. Mach Vision Appl. 2020;31(1):8.
322. de Vos BD, Berendsen FF, Viergever MA, Sokooti H, Staring M, Išgum I. A deep learning framework for unsupervised affine and deformable image registration. Med Image Anal. 2019;52:128–43.
323. Yang X, Kwitt R, Styner M, Niethammer M. Quicksilver: fast predictive image registration—a deep learning approach. NeuroImage. 2017;158:378–96.

324. Miao S, Wang ZJ, Liao R. A CNN regression approach for real-time 2D/3D registration. IEEE Trans Med Imaging. 2016;35(5):1352–63.
325. Li P, Pei Y, Guo Y, Ma G, Xu T, Zha H. Non-rigid 2D–3D registration using convolutional autoencoders. In: 2020 IEEE 17th international symposium on biomedical imaging (ISBI). IEEE; 2020. p. 700–4.
326. Zhang J, Yeung SH, Shu Y, He B, Wang W. Efficient memory management for GPU-based deep learning systems; 2019. arXiv preprint arXiv:1903.06631.
327. Zhao H, Han Z, Yang Z, Zhang Q, Yang F, Zhou L, Yang M, Lau FC, Wang Y, Xiong Y, et al. Hived: sharing a {GPU} cluster for deep learning with guarantees. In: 14th {USENIX} symposium on operating systems design and implementation ({OSDI} 20); 2020. p. 515–32.
328. Lin Y, Jiang Z, Gu J, Li W, Dhar S, Ren H, Khailany B, Pan DZ. DREAMPlace: deep learning toolkit-enabled GPU acceleration for modern VLSI placement. IEEE Trans Comput Aided Des Integr Circuits Syst. 2020;40:748–61.
329. Hossain S, Lee DJ. Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices. Sensors. 2019;19(15):3371.
330. Castro FM, Guil N, Marín-Jiménez MJ, Pérez-Serrano J, Ujaldón M. Energy-based tuning of convolutional neural networks on multi-GPUs. Concurr Comput Pract Exp. 2019;31(21):4786.
331. Gschwend D. Zynqnet: an fpga-accelerated embedded convolutional neural network; 2020. arXiv preprint arXiv:2005.06892.
332. Zhang N, Wei X, Chen H, Liu W. FPGA implementation for CNN-based optical remote sensing object detection. Electronics. 2021;10(3):282.
333. Zhao M, Hu C, Wei F, Wang K, Wang C, Jiang Y. Real-time underwater image recognition with FPGA embedded system for convolutional neural network. Sensors. 2019;19(2):350.
334. Liu X, Yang J, Zou C, Chen Q, Yan X, Chen Y, Cai C. Collaborative edge computing with FPGA-based CNN accelerators for energy-efficient and time-aware face tracking system. IEEE Trans Comput Soc Syst. 2021. https://doi.org/10.1109/TCSS.2021.3059318.
335. Hossin M, Sulaiman M. A review on evaluation metrics for data classification evaluations. Int J Data Min Knowl Manag Process. 2015;5(2):1.
336. Provost F, Domingos P. Tree induction for probability-based ranking. Mach Learn. 2003;52(3):199–215.
337. Rakotomamonyj A. Optimizing area under roc with SVMS. In: Proceedings of the European conference on artificial intelligence workshop on ROC curve and artificial intelligence (ROCAI 2004), 2004. p. 71–80.
338. Mingote V, Miguel A, Ortega A, Lleida E. Optimization of the area under the roc curve using neural network supervectors for text-dependent speaker verification. Comput Speech Lang. 2020;63:101078.
339. Fawcett T. An introduction to roc analysis. Pattern Recogn Lett. 2006;27(8):861–74.
340. Huang J, Ling CX. Using AUC and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng. 2005;17(3):299–310.
341. Hand DJ, Till RJ. A simple generalisation of the area under the ROC curve for multiple class classification problems. Mach Learn. 2001;45(2):171–86.
342. Masoudnia S, Mersa O, Araabi BN, Vahabie AH, Sadeghi MA, Ahmadabadi MN. Multi-representational learning for offline signature verification using multi-loss snapshot ensemble of CNNs. Expert Syst Appl. 2019;133:317–30.
343. Coupé P, Mansencal B, Clément M, Giraud R, de Senneville BD, Ta VT, Lepetit V, Manjon JV. Assemblynet: a large ensemble of CNNs for 3D whole brain MRI segmentation. NeuroImage. 2020;219:117026.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com